

الله يحيي الموتى

جزوه درس پردازش تکاملی

استاد محترم

دکتر محمدمهری عبادزاده

گردآوری

زهرا قنبری گرانی

نیمسال اول ۹۴-۹۳

فهرست

فصل اول: مقدمه ۶	۱
۶..... پیش درآمد	۱-۱
۷..... فصل دوم: تئوری تکامل	۲
۷..... مقدمه	۱-۲
۸..... بهینه سازی	۲-۲
۸..... مسائل محاسبه و غیرمحاسبه	۱-۲-۲
۸..... تکامل	۳-۲
۸..... تکامل چیست؟	۱-۳-۲
۹..... لزوم تکامل	۲-۳-۲
۹..... انتخاب طبیعت	۲-۳-۲
۱۰..... چگونگی تکامل	۴-۳-۲
۱۲..... ویژگی‌های جدید	۲-۳-۵
۱۳..... فصل سوم: تئوری تکامل از دیدگاه میکروسکوپی	۳
۱۳..... مقدمه	۱-۳
۱۳..... زیگوت	۲-۳
۱۴..... سلول	۳-۳
۱۵..... اجزاء سلول	۱-۳-۳
۱۷..... موجودات دیپلوبیدی و هاپلوبیدی	۴-۳
۱۷..... موجودات هاپلوبیدی	۱-۴-۳
۱۸..... موجودات دیپلوبیدی	۲-۴-۳
۱۸..... ویژگی‌ها و نحوه به ارث رسیدن آن‌ها	۵-۳
۲۰	
۲۰..... آزمایش مندل	۱-۵-۳
۲۱..... ژنوتیپ و فونوتیپ	۶-۳
۲۲..... DNA	۷-۳
۲۶..... کُدن	۸-۳
۲۸..... زن	۹-۳
۲۸..... چگونگی تولید پروتئین توسط سلول و ایجاد ویژگی‌های فیزیکی	۱۰-۳
۲۹..... نحوه انجام ترجمه و تولید پروتئین	۳-۱۰-۱
۳۲..... خطاهای	۳-۱۰-۲
۳۳..... نحوه تولید پروتئین از روی RNA	۳-۱۰-۳
۴۰..... انتقال ویژگی‌ها به فرزندان	۱۱-۳
۴۰..... تقسیم‌های سلولی	۱-۱۱-۳
۴۶	
فصل چهارم: تئوری تکامل از دیدگاه ماکروسکوپی	۴
۴۶..... مقدمه	۱-۴
۴۶..... ساختار کلی الگوریتم‌های تکاملی	۲-۴
۴۷..... شناسنامه الگوریتم‌های تکاملی	۱-۲-۴
۵۰..... بازنمایی (کد کردن)	۲-۲-۴
۳۵..... جمعیت اولیه	۴-۲-۳
۳۶..... انتخاب و انواع آن	۴-۲-۴
۵۴..... شرط خاتمه	۵-۲-۴
۵۶..... جهش و بازترکیبی	۶-۲-۴
۶۸..... فصل پنجم: الگوریتم‌های پایه	۵

۶۸.....	مقدمه	۱-۵
۶۸.....	الگوریتم ژنتیک	۲-۵
۶۸.....	شناسنامه GA	۱-۲-۵
۷۰.....	استراتژی‌های تکامل	۳-۵
۷۲.....	رابطه متوسط گیری پنجره‌ای	۱-۳-۵
۷۲.....	شناسنامه الگوریتم ES	۲-۳-۵
۷۳.....	خود تطبیقی	۳-۳-۵
۷۴.....	نرخ یادگیری (τ)	۴-۳-۵
۷۴.....	چگونگی جهش برای به دست آوردن σ و α	۵-۳-۵
۷۵.....	های فوق	
۷۶.....	یک کاربرد عملی	۶-۳-۵
۷۷.....	اثبات قانون $1/5$ موقتیت	۴-۵
۸۲.....	انواع ES و استراتژی‌های آن	۵-۵
ES(1 + 1).....		۱-۵-۵
۸۳.....		
ES(μ ,/ $+\lambda$)		۲-۵-۵
۸۳.....		
ES(μ ,/ $+\lambda$)		۳-۵-۵
۸۳.....		
ES(μ ,/ $+\lambda$)		۴-۵-۵
۸۳.....		
۸۵.....	برنامه نویسی تکاملی (EP)	۵-۶
۸۵.....	ماشین حالت محدود (FSM)	۱-۶-۵
۸۶.....	جهش در EP	۲-۶-۵
۸۷.....	برنامه نویسی ژنتیک (GP)	۷-۵
۸۹.....	عبارت سمبولیک	۱-۷-۵
۹۰.....	بازنمایی در GP	۲-۷-۵
۹۲.....	تولید جمعیت اولیه	۳-۷-۵
۹۳.....	جهش در GP	۵-۷-۴
۹۴.....	بازترکیبی در GP	۵-۷-۵
۹۵.....	انتخاب والدین	۶-۷-۵
۹۶.....	خاتمه الگوریتم	۷-۷-۵
۹۶.....	چاقی در GP	۸-۷-۵
۹۸.....	حجم بالای محاسبات در GP	۹-۷-۵
۹۹.....	فصل ششم: مسائل چند هدفه	۶
۹۹.....	مقدمه	۱-۶
۹۹.....	روش پرتو	۲-۶
۱۰۰.....	غلیه پرتو	۱-۲-۶
۱۰۱.....	الگوریتم‌های صریح و غیر صریح	6-3
۱۰۲.....	مهاجرت	۱-۳-۶
۱۰۳.....	فاصله همسایگی	۲-۳-۶
۱۰۴.....	تشریک بازنده‌گی	6-3-3
۱۰۵.....	روش ازدحام	۴-۳-۶
۱۰۵.....	تفاوت روش‌های تشریک	۵-۳-۶
۱۰۵.....	برازندگی و ازدحام	
۱۰۶.....	الگوریتم PSO استاندارد: نسخه تک گروهی	۴-۶

۱۰۷.....	فلوچارت الگوریتم	6-4-1
۱۰۹.....	فصل هفتم: کنترل قیدها و محاسبه مقدار بهینه پارامترها	۷
۱۰۹.....	مقدمه	۱-۷
۱۰۹.....	کنترل قیدها	۲-۷
۱۱۰.....	مقدار بهینه پارامترها	۳-۷
۱۱۰.....	روش‌های غیربرخط	7-3-1
۱۱۰.....	روش‌های برخط	7-3-2

فصل اول: مقدمه

۱-۱ پیش درآمد

در این درس به الگوریتم‌های تکاملی^۱ (EA) به عنوان روشی برای حل مسائل بهینه سازی غیرمحدب^۲ خواهیم پرداخت.

در بخش مطالب درس ابتدا تئوری تکامل از دو دیدگاه میکروسکوپی و ماکروسکوپی مورد بررسی قرار خواهد گرفت. سپس به الگوریتم‌های تکاملی پایه می‌پردازیم که در این بخش الگوریتم ژنتیک^۳ (GA)، استراتژی‌های تکامل^۴ (ES)، برنامه ریزی تکامل^۵ (EP) و برنامه ریزی ژنتیک^۶ (GP) مورد بحث قرار می‌گیرد.

در ادامه به الگوریتم‌های تکاملی چند جوابی می‌پردازیم. همچنین روش‌هایی را مطرح خواهیم نمود که در آن‌ها محدودیت‌هایی بر الگوریتم‌های تکاملی اعمال شده باشد. در پایان نیز به الگوریتم‌های تکاملی چند هدفی خواهیم پرداخت.

¹ Evolutionary Algorithms

² Non-Convex

³ Genetic Algorithm

⁴ Evolution Strategies

⁵ Evolution Programming

⁶ Genetic Programming

فصل دوم: تئوری تکامل

۱- مقدمه

الگوریتم‌های تکاملی^۱ (EA) با الهام از تئوری تکامل به وجود آمده‌اند. با این حال در الگوریتم‌های تکاملی موجود، تئوری تکامل بسیار ساده شده است و به عبارت دیگر؛ تمامی جزئیات در آن‌ها وجود ندارد.

در این درس ابتدا به این موضوع خواهیم پرداخت که الگوریتم‌های پایه از چه قسمت‌هایی تشکیل شده‌اند و نقش هر قسمت و نیز روش‌های مورد استفاده در آن‌ها چیست. پس از بررسی قسمت‌های مختلف تشکیل دهنده الگوریتم‌های تکاملی، به سراغ الگوریتم‌های تکاملی استاندارد موجود خواهیم رفت و وضعیت آن‌ها را در مقایسه با الگوریتم‌های تکاملی پایه بررسی خواهیم نمود.

۲- بهینه سازی

الگوریتم‌های تکاملی در حقیقت روش‌های بهینه سازی هستند. و بهینه سازی به معنای پیدا کردن کمینه^۲ یکتابع است. باید توجه داشت که در صورت تمایل برای یافتن بیشینه^۳ نیز روال کار تغییر نخواهد کرد و نهایتاً مقدار کمینه به دست آمده به راحتی قابل تغییر به بیشینه است. چند روش برای این کار به شرح زیر هستند:

- معکوس کردن: $\frac{1}{f(x)}$
- منفی کردن: $-f(x)$
- کم کردن از یک مقدار بزرگ: $m - f(x)$

همچنین می‌توانیم علاوه بر هدف کمینه یا بیشینه کردن تابع، محدودیت‌های^۴ (قیدها یا شرایط) را به مساله اعمال نماییم و یا محدوده‌ی فضای جستجو را مشخص کنیم:

$$g(x) = 0$$

$$h(x) < 0$$

$$k(x) > 0$$

$$A < x < B$$

در حالت دیگری از مسائل بهینه سازی، می‌توانیم چند هدف داشته باشیم. در این حالت نیز می‌توان مسأله را با داشتن قیدهایی حل کرد:

$$\min(f_1(x), f_2(x), \dots, f_n(x)) \quad s.t. \begin{cases} g(x) = 0 \\ g(x) \leq 0 \end{cases}$$

¹ Evolutionary Algorithms

² Minimum

³ Maximum

⁴ Constraints

لازم به ذکر است که قیدها عملاً به صورت کوچک‌تر یا مساوی هستند و مابقی قیدها را می‌توان به این دو حالت تبدیل کرد.

مسائل محدب و غیرمحدب

در ریاضی اگر با یک مسأله محدب مواجه باشیم، به این معناست که تابع تنها یک کمینه دارد. در چنین حالتی می‌توان با روش‌های تحلیلی چون روش نزول در امتداد گرادیان مسأله را حل کرد و به جواب رسید.

در حالت غیرمحدب یعنی زمانی که بیش از یک کمینه محلی داشته باشیم، روش گرادیان تنها قادر خواهد بود کمینه محلی را پیدا کند.

توانایی الگوریتم‌های تکاملی در این جا نمایان می‌شود. این الگوریتم‌ها قادر خواهند بود از کمینه‌های محلی^۱ فرار کرده و کمینه‌های سراسری^۲ (اصلی) را پیدا کنند. طبعاً یکی از روش‌های یافتن کمینه‌های عمومی، پیدا کردن تمام کمینه‌های است. با این حال، در الگوریتم‌های تکاملی استاندارد، تنها یکی از کمینه‌ها پیدا می‌شوند. لذا یکی از مسائلی که در این درس بررسی خواهیم کرد، الگوریتم‌های تکاملی چند جوابی است.

۴-۱ تکامل

تکامل چیست؟

پیش از آغاز بحث لازم به توضیح است که در این جا، تعریف علمی تکامل مد نظر است. تئوری تکامل در سال ۱۸۵۹ توسط داروین در کتاب منشاء موجودات به جهان معرفی گردید. از زمانی که زندگی بر روی زمین آغاز شد، میلیونها گونه جدید پدید آمده و از بین رفته‌اند. موجودات زنده امروزی، تنها نمایانگر بخش کوچکی از تمام موجودات زنده‌ای هستند که تا به حال وجود داشته‌اند. موجودات طی فرآیندی مستمر، تحول پیدا می‌کنند که تکامل نامیده می‌شود.

تکامل به معنای افزایش متوسط شایستگی موجودات است. این امر در طی سالیان و برای نسل‌های بعد محقق خواهد شد. به بیان دیگر، در تکامل، بقای نسل موجودات و در عین حال تولید فرزندان شایسته‌تر از والدین را داریم که نهایتاً منجر به افزایش موجودات شایسته و از سوی دیگر، کاهش موجوداتی با شایستگی کم‌تر خواهد شد.

شایستگی به معنای داشتن ویژگی‌های فیزیکی مناسب به منظور تطابق با محیط یا به تعبیر دیگر، استفاده از امکانات محیط است. لزوم توانایی تطابق با محیط نیز به منظور افزایش فرزندان شایسته است. در واقع آن چه اهمیت دارد، داشتن تعداد نسل‌های بیشتر است. به این ترتیب تعداد فرزندان، که می‌توان از آن به گسترش عرضی نسل یاد کرد، از آن جهت حائز اهمیت است که می‌تواند به افزایش تعداد نسل‌ها، گسترش طولی نسل، کمک کند.

¹ Local Minimums

² Global Minimums

به این ترتیب همان گونه که ذکر گردید، هدف نهایی، داشتن نسل‌های بیشتر از یک موجود و انتقال ویژگی‌های ژنتیکی موجود مذکور به آن‌ها خواهد بود.

دو نوع انتقال را می‌توان مابین موجودات تعریف کرد:

- ممتيک^۱: انتقال نرم افزاري است. به تعبيير ديگر، به معنai انتقال آموزه‌های والدين به فرزندان است.
- ژنتيک^۲: ژنتيک در مقابل ممتيک، انتقال ویژگي‌های فيزيكى است.

آن چه در بحث ما مد نظر است، ژنتيک است و نه ممتيک.

لزوم تکامل

تکامل و لزوم آن، به دليل محدوديت منابع مطرح می‌گردد. محدوديت منابع، سبب ايجاد رقابت در استفاده از آن‌ها می‌شود. در اين رقابت دو راه برای موجودات وجود دارد: جنگیدن و یا تغيير منابع (استفاده از منابع ديگر). و در هر دو صورت برای پیروزی در اين رقابت، نيازمند تکامل خواهند بود. موجوداتی که جنگ را برمی‌گزینند برای برتری می‌بايست ویژگي‌های فيزيكى مناسب داشته باشند. لذا نياز به تکامل دارند. نمونه‌اي از اين حالت را می‌توان در ویروس‌ها مشاهده کرد. اين ویروس‌ها در مقابل آنتى بادى‌هایي که برای از بين بردنشان تولید می‌شود دائمآ خود را تکامل می‌دهند؛ چرا که در غير اين صورت از بين خواهند رفت. اساساً می‌توان گفت که هر موجودی در روند تکامل متوقف گردد، از بين خواهد رفت. مگر موجوداتی که دشمن خاصی ندارند و بر سر منابع نيز با موجودات ديگر در رقابت نيسند؛ مانند تک سلولی‌ها که در طول سال‌ها تکامل پيدا نکرده‌اند.

همان طور که اشاره شد، دسته ديگري از موجودات، به جاي نبرد بر سر منابع، به استفاده از منابع ديگر روی می‌آورند. اين دسته نيز نيازمند تکامل و داشتن ویژگي‌های فيزيكى لازم برای استفاده از منابع جديد خواهند بود. از اين دسته می‌توان به زرافه اشاره کرد که برای استفاده از برگ‌های روبيده بر شاخه‌های بالايی درختان، به مرور گردنی بلند پيدا کرده است.

بر اساس آن چه گفته شد، می‌توان نتيجه گرفت که تنوع موجودات، اين است که موجودات بتوانند از منابع گوناگون استفاده کرده و باقی بمانند. در هر حال باید به خاطر داشت که رقابت سبب انتخاب می‌شود.

انتخاب طبیعت

هنگامی که با محدوديت منابع رو به رو می‌شويم، مسئله انتخاب پيش می‌آيد. موجوداتی که شايستگی بيشتر و نيز توانايي بيشتری در كسب منابع دارند، باقی می‌مانند (انتخاب می‌شوند) و موجودات ديگر حذف می‌شوند. اين همان انتخاب طبیعت است که بر اساس شايستگی انجام می‌شود. درواقع قانون طبیعت قانون بقای شايسته ترین هاست.

¹ Memetic
² Genetic

موجودی شایسته‌تر است و باقی می‌ماند که منابع لازم برای ادامه بقای خود و تولید فرزند را در اختیار داشته باشد و همان گونه که پیش‌تر نیز بیان گردید، هر چه تعداد نسل‌هایی که این زنجیره تولید فرزندان ایجاد می‌کنند، بیش‌تر باشد، موجود شایسته‌تر است.

چگونگی تکامل

با تعریفی که در اینجا از تکامل مد نظر است، هیچ موجودی نمی‌تواند خود را تکامل دهد. آن‌چه که تغییر می‌کند شایستگی فرزندان است. اکنون مسأله این است که چگونه فرزندان می‌توانند ویژگی‌های مناسب‌تری نسبت به والدین داشته باشند؟ ساده‌ترین پاسخ این است که اگر فرزند کلیه ویژگی‌های خوب پدر و مادر را بگیرد، شایسته‌تر خواهد بود. اما این نگاه دو مشکل عمده دارد: از نظر طبیعی، هیچ تضمینی وجود ندارد که فرزند فقط ویژگی‌های خوب پدر و مادر را بگیرد. به علاوه، ویژگی‌های موجودات محدود هستند و در صورتی که فرزند فقط ویژگی‌های خوب پدر و مادرش را بگیرد، زمانی این ویژگی‌ها به حد نهایی خود خواهند رسید و تکامل پایان می‌یابد. در واقع به وضعیت "بهتر از این نمی‌شود!" رسیده و متوقف می‌شود و این با ویژگی‌بینهایت بودن تکامل تنافض دارد. این رویه همچنین سبب کاهش قابل توجه تنوع موجودات خواهد شد.

آن‌چه در حقیقت اتفاق می‌افتد، این است که ویژگی‌های والدین به صورت تصادفی به فرزندان انتقال می‌یابد. فرزندان با احتمال برابر ویژگی‌های خوب یا بد را دارا خواهند بود اما انتخاب طبیعت سبب می‌شود متوسط نسل به سمت شایسته‌تر شدن پیش‌رود. به این ترتیب برای تکامل به دو عامل نیاز است:

- تنوع: شامل ویژگی‌های موجود و همچنین ویژگی‌های جدید
- انتخاب: مبتنی بر شایستگی

تنوع به معنای وجود موجوداتی با شایستگی پایین، متوسط و بالا در جمعیت است و انتخاب مبتنی بر شایستگی به معنای برگزیده شدن موجودات شایسته است. در تنوع واگرایی داریم. تولید مثل سبب افزایش کمیت و به تبع آن، تنوع خواهد شد چرا که اشاره کردیم آن‌چه فرزندان از والدین به ارث می‌برند، تنها ویژگی‌های خوب آن‌ها نیست، بلکه مجموعه‌ای از ویژگی‌های خوب و بد است.

مثال: فرض کنید والد ۱ قد بلند و دارای هوش پایین باشد و والد ۲ قد کوتاه و باهوش. فرزندان با احتمال برابر می‌توانند قد بلند و باهوش، قد بلند و با هوش پایین، قد کوتاه و با هوش و یا قد کوتاه و با هوش پایین باشند. بدیهی است که فرزند قد بلند و باهوش، خواهان بیش‌تری خواهد داشت و می‌تواند نسل خود را افزایش دهد. در مقابل فرزند قد کوتاه و با هوش پایین، خواهانی نخواهد داشت. به این ترتیب، تکامل سبب می‌شود که در نسل‌های آتی، تعداد متوسط ویژگی‌های بد کاهش یافته و تعداد متوسط ویژگی‌های خوب افزایش یابد.

گرچه در نگاه اول ممکن است به نظر برسد برای حرکت در مسیر تکامل می‌بایست موجودات ضعیف را حذف کرد، اما در واقع چنین نیست. تنوع، و به بیان روشن‌تر، وجود موجودات با شایستگی پایین در کنار موجوداتی با شایستگی بالا، یکی از ارکان تکامل است و خواهیم دید که بدون وجود آن عملاً روند تکامل نیز خیلی زود متوقف خواهد شد. بنابراین لازمه تکامل این است که تنوع و از طرف دیگر انتخابی بر اساس شایستگی داشته باشیم. تنوع و انتخاب، مفاهیم متناقضی به شمار می‌آیند. تنوع، مستلزم وجود موجوداتی با ویژگی‌های متفاوت است و انتخاب، به معنای وجود موجوداتی با ویژگی‌های بهتر. یکی از دغدغه‌ها در بحث الگوریتم‌های تکاملی، ایجاد تعادل بین این دو است چرا که همان طور که اشاره شد، تکامل به هر دوی این عوامل نیاز دارد و در صورت عدم وجود یکی از آن‌ها، تکامل وجود نخواهد داشت.

تنوع با اکتشاف، جستجوی عمومی و واگرایی معادل است و انتخاب با استخراج، جستجوی محلی و همگرایی. در اثر تولید مثل و افزایش تنوع، با واگرایی رو به رو خواهیم شد. تنوع موجب افزایش کمیت می‌شود در صورتی که انتخاب موجب افزایش کیفیت می‌شود.

۱-۴-۴-۱ اکتشاف و استخراج

مفهوم اکتشاف و استخراج از حوزه معدن وام گرفته شده است و به خوبی گویای مفهوم استخراج و اکتشاف در الگوریتم‌های تکاملی است. فرض کنید بخواهیم در منطقه‌ای طلا استخراج کنیم. برای این کار ابتدا باید معدنی را کشف کنیم که هم از نظر میزان طلا و هم از نظر عیار آن، قابل توجه باشد. به علاوه، نسبت به هزینه احتمالی، ارزش استخراج را داشته باشد. بدیهی است، هر چه بیشتر جستجو کنیم به معادن بهتری دست خواهیم یافت اما در مقابل، هزینه و زمان بیشتری باید صرف شود. بحث جستجوی عمومی و محلی را که متناظر با تنوع و انتخاب است می‌توان مشابه با اکتشاف و استخراج در نظر گرفت. در واقع باید تعیین کنیم که تا چه زمانی دنبال اکتشاف و یافتن معادن باشیم (جستجوی عمومی/ایجاد تنوع) و از چه زمانی استخراج (جستجوی محلی/انتخاب) را آغاز کنیم. طبعاً اقدام به استخراج، پس از یافتن و بررسی همه معادن، منطقی نیست. بلکه می‌بایست مابین اکتشاف و استخراج و مصالحه برقرار شود. چرا که اگر فقط به دنبال یافتن بهترین معدن باشیم، تنها باید هزینه کیم و فرصت استخراج و در پی آن سود بردن از معادن را پیدا نخواهیم کرد. از سوی دیگر، اگر با یافتن اولین معدن استخراج را آغاز کنیم، ممکن است میزان ذخایر و عیار معدن به گونه‌ای نباشد که حتی توان جبران هزینه استخراج را داشته باشد. در برخی الگوریتم‌ها این دو فاز به صورت مجزا انجام می‌گیرند اما در الگوریتم‌های تکاملی این دو فاز همزمان انجام می‌شوند. این که تا چه زمانی جستجوی عمومی انجام دهیم و جستجوی محلی را از چه زمانی شروع کنیم، چالش جدی است که در الگوریتم‌های تکاملی با آن مواجهیم.

ویژگی‌های جدید

در بحث تنوع، امکان ایجاد ویژگی‌های جدیدی که قبلاً وجود نداشته‌اند نیز مطرح است. در غیر این صورت بعد از مدتی ویژگی‌های خوب به سقف خود رسیده و در همان وضعیت باقی خواهد ماند.

در ایجاد تنوع، آن بخش که مربوط به ایجاد ویژگی‌های جدید است با تصادف همراه خواهد بود. در مقابل تنوع، انتخاب را داریم که باعث می‌شود تعداد موجودات کم شوند. در اثر انتخاب، متوسط ویژگی‌های مناسب در جمعیت بالا می‌رود و این را تکامل گویند.

اگر انتخاب صد در صد بر اساس شایستگی باشد، عملاً تنوع از بین خواهد رفت و به همگرایی زودرس می‌رسیم. لذا روش‌هایی که بیش از حد به شایستگی اهمیت می‌دهند، خیلی سریع به کمینه محلی می‌رسند و قادر نخواهند بود مقدار بهینه را پیدا کنند.

۱-۵-۴ جهش

در طبیعت مکانیزمی به نام جهش وجود دارد که خطا محسوب می‌شود و طبیعت سعی در جلوگیری از آن دارد و احتمال رخداد آن بسیار اندک و نزدیک به صفر است. در جهش، ویژگی ایجاد شده در موجود، به صورت تقریباً تصادفی ایجاد می‌شود و از پدر یا مادر گرفته نمی‌شود. روشن است که جهش می‌تواند فرآیند کند تکامل را تسريع کند. به همین منظور، در برخی الگوریتم‌های تکاملی از آن استفاده می‌شود.

۱-۵-۵ انواع تکامل

تکامل به دو صورت در طبیعت به چشم می‌خورد: تکامل داخل نوعی و بین نوعی. در تکامل داخل نوعی ویژگی‌ها عوض نمی‌شوند و ویژگی جدیدی نیز به وجود نمی‌آید، بلکه در عوض ویژگی‌ها تکامل می‌یابند. این نوع تکامل، سقف مشخصی دارد و با رسیدن به تمام ظرفیت خود تمام می‌شود. به عنوان مثال یک انسان هر میزان که تکامل پیدا کند قادر به پرواز نخواهد بود؛ مگر اینکه نوع آن عوض شود و بتواند پرواز کند!

در مقابل، تکامل بین نوعی مرزی ندارد و تا می‌تواند بی‌نهایت ادامه یابد. تکامل در الگوریتم‌های تکاملی موجود، از نوع داخل نوعی است.

فصل سوم: تئوری تکامل از دیدگاه میکروسکوپی

۱-۵ مقدمه

در این فصل به این پرسش خواهیم پرداخت که ویژگی‌های یک موجود زنده که نشان دهنده شایستگی آن موجود است چگونه ایجاد می‌شوند؟

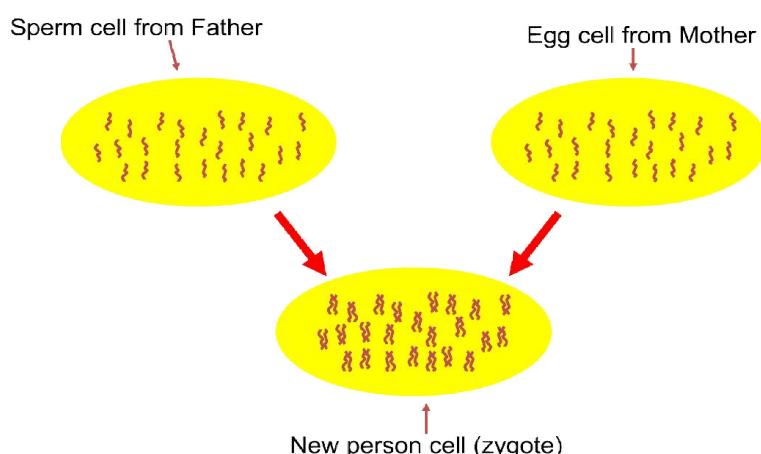
در این خصوص سه مسأله را مورد بررسی قرار می‌دهیم:

- ویژگی‌های فیزیکی موجود زنده چگونه ایجاد می‌شوند؟
- ویژگی‌های فیزیکی در موجودات زنده چگونه منتقل می‌شوند؟
- ویژگی‌های فیزیکی در موجودات زنده چگونه متفاوت می‌شوند؟

برای پاسخ دادن به سوالات بالا، ابتدا لازم است بدانیم موجودات زنده چگونه به وجود می‌آیند. کوچکترین واحد زنده یک موجود زنده، سلول است و اولین سلول تشکیل دهنده موجود زنده زیگوت است. به این دلیل در ادامه به تعریف زیگوت و چگونگی تقسیم آن و تولید موجود زنده خواهیم پرداخت.

۶- زیگوت

زیگوت^۱ اولین و کوچکترین واحد زنده است که یک موجود زنده را ایجاد می‌کند. به بیان دیگر، زیگوت اولین سلول تشکیل دهنده موجود زنده است که مابقی سلول‌ها از آن ایجاد می‌شوند. زیگوت طی مراحلی که در ادامه توضیح خواهیم داد به موجود زنده تبدیل می‌شود. موجودات نر، سلول جنسی‌ای تولید می‌کنند که اسپرم نام دارد. سلول جنسی تولید شده توسط موجودات ماده، تخمک نامیده می‌شود. در اثر لقاد اسپرم و تخمک، زیگوت ایجاد می‌شود که فرزند را تولید می‌کند.



^۱ Zygote

زیگوت با تقسیم سلولی، به دو سلول مانند خود تقسیم می‌شود. آن دو به ۴ سلول تقسیم می‌شوند و این روند با توان‌های ۲ ادامه خواهد یافت. با رسیدن سلول‌ها به تعدادی معین، آن مجموعه را سلول‌های بنیادی جنینی می‌گویند. در حقیقت تولید زیگوت در یک چرخه رخ می‌دهد؛ از لقاح اسپرم و تخمک زیگوت تولید می‌شود و از آن، موجودات زنده. موجودات زنده اسپرم و تخمک تولید می‌کنند و مجدداً زیگوت را به وجود می‌آورند.

تبدیل زیگوت به موجود زنده، شامل مراحل زیر است:

۱- تقسیم سلولی زیگوت و ایجاد سلول‌های بنیادی جنینی

۲- تقسیم سلولی سلول‌های بنیادی جنینی و تولید سلول‌های بنیادی بالغ و سلول‌های تخصصی در واقع آن چه رخ می‌دهد به این ترتیب است: زیگوت، تقسیم سلولی، تک سلولی، پر سلولی.

سلول‌های بنیادی جنینی با تقسیم سلولی به سلول‌های تخصص (مثل سلول‌های عصبی، خون، استخوان، عضله و ...) و همچنین سلول‌های بنیادی بالغ تقسیم می‌شوند. سلول‌های بنیادی بالغ سلول‌هایی هستند که با تقسیم به سلول‌های تخصصی و سلول‌های بالغ تبدیل می‌شوند. باید توجه داشت که بر خلاف سلول‌های بنیادی بالغ، سلول‌های تخصصی تنها می‌توانند به سلول‌های تخصص تبدیل شوند.

پس از مدتی سلول‌های جنینی از بین می‌روند و تنها سلول‌های بنیادی بالغ و تخصص باقی می‌مانند. قابلیت ترمیم در بدن، به این دلیل است که سلول‌های بنیادی بالغ در مغز استخوان وجود دارند و می‌توانند به سلول‌های خون، عضله و ... تبدیل شوند. دلیل عدم ترمیم پذیری و برگشت ناپذیری در قطع نخاع این است که سلول‌های بنیادی بالغ قادر به تولید سلول‌های عصبی نیستند. با این حال، سلول‌های جنینی که در بند ناف وجود دارند این قابلیت را دارند.

۷-۱ سلول

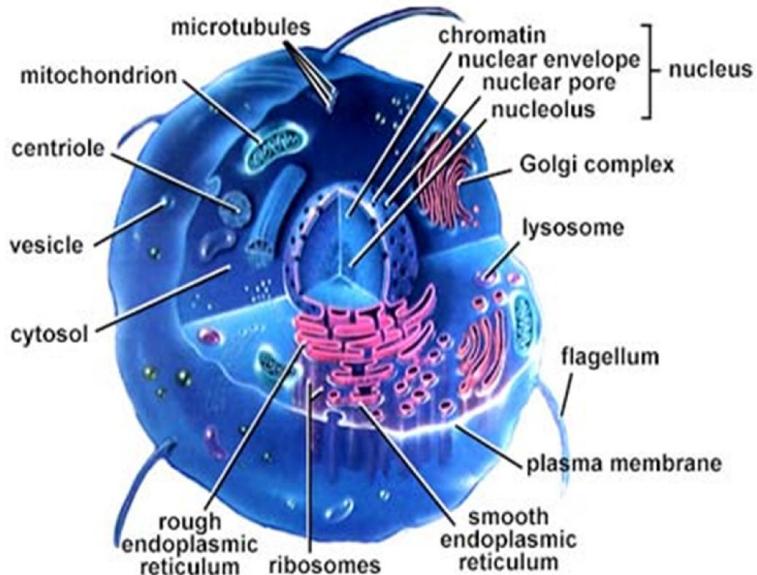
همان گونه که اشاره شد، سلول کوچکترین واحد تشکیل دهنده موجود زنده است. شکل زیر یک سلول بنیادی و اجزای آن را نمایش می‌دهد.

سلول‌ها را می‌توان به سه دسته تقسیم کرد:

- سلول‌های بنیادی جنینی

- سلول‌های بنیادی بالغ

- سلول‌های بنیادی تخصصی



اجزاء سلول

با توجه به این که پرداختن به ساختار سلول و جزئیات آن خارج از حوصله این درس است، به اختصار نکاتی را مرور خواهیم کرد. در کلی ترین حالت می‌توان گفت که سلول از هسته، غشاء و اندامک‌ها تشکیل شده است.

۱-۱-۱-۱ غشاء

محافظ سلول است که اندامک‌ها از آن خارج نشوند. به علاوه وظیفه کنترل ورود و خروج مواد به سلول را نیز بر عهده دارد. اگر غشاء پاره شود، سلول می‌میرد.

۲-۱-۱-۱ هسته

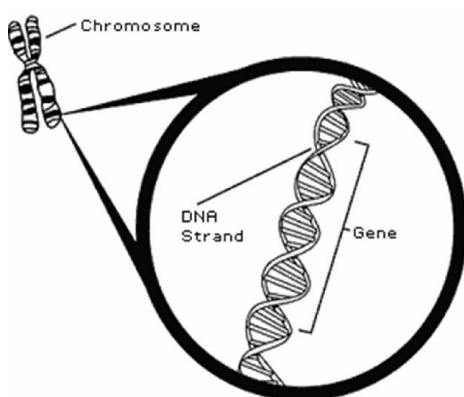
اطلاعات ژنتیکی هر موجود زنده داخل هسته است و هسته همه سلول‌ها یکسان است. خود هسته نیز غشاء دارد. این غشاء از اندامک‌های داخل آن محافظت می‌کند و وظیفه کنترل ورود و خروج مواد را نیز بر عهده دارد. در حقیقت غشاء هسته از اطلاعات ژنتیکی داخل آن محافظت می‌کند و مانع تغییر آن‌ها می‌شود. ویژگی‌های فیزیکی هر موجود زنده در هسته کد می‌شود و مشخص می‌کند که این ویژگی‌ها چه باید باشند. داخل هسته ژنوم وجود دارد که شامل تعدادی کروموزوم است (به عنوان مثال، این تعداد برای انسان ۲۳ جفت و برای درخت بالای صد است).

۱-۷-۱-۲-۱ کروموزوم^۱

اطلاعات ژنتیکی در کروموزوم‌ها ذخیره می‌شود. هر کروموزوم از دو رشته DNA تشکیل شده است. کروموزوم‌ها که حاوی اطلاعات ژنتیکی هستند، داخل هسته قرار دارند. این اطلاعات ثابتند. به زبان ساده می‌توان کروموزوم‌ها را به بسته‌های مواد ژنتیکی تشبیه کرد که درون هسته سلول‌ها ذخیره شده‌اند. و درواقع به شکل مولکول‌های DNA همراه با پروتئین‌ها هستند که به این بسته‌ها فامتن یا کروموزوم گفته می‌شود.

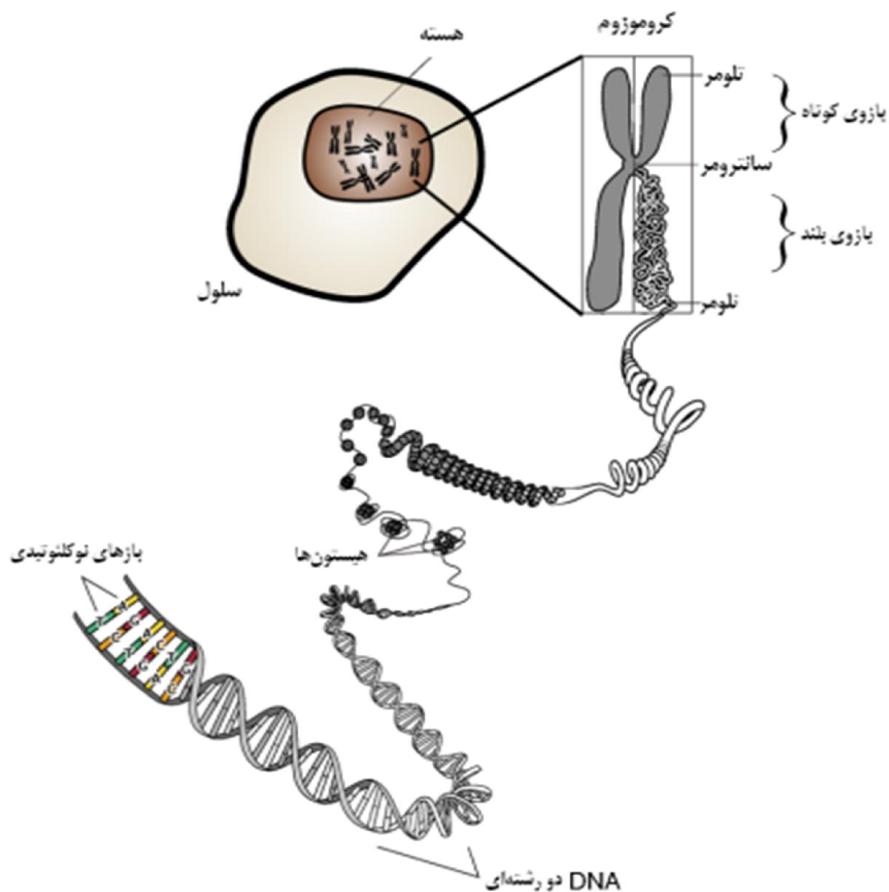
کروموزوم به بخش‌هایی تقسیم می‌شود که هر بخش را یک ژن^۲ می‌نامند. ژن‌ها، نقش کد کردن ویژگی‌ها را بر عهده دارند که در ادامه مفصل‌تر به آن خواهیم پرداخت.

شکل ساده شده کروموزوم در زیر آمده است:



شکل زیر نیز ارتباط سلول، هسته، کروموزوم و DNA را نمایش می‌دهد:

¹ Chromosome
² Gene



۸- موجودات دیپلوبتی و هاپلوبتی

موجودات زنده از نظر وضعیت کروموزوم‌ها به دو دسته تقسیم می‌شوند:

- موجودات دیپلوبتی^۱
- موجودات هاپلوبتی^۲

از آن جا که این دو گروه پایه شکل گیری انواعی از الگوریتم‌های تکاملی هستند، در ادامه به شرح این موجودات و ویژگی‌های آن خواهیم پرداخت.

موجودات هایپلوبتی

در این موجودات، از هر کروموزوم یک عدد وجود دارد؛ مانند تک سلولی‌ها.

¹ Diploid
² Haploid

موجودات دیپلولئیدی

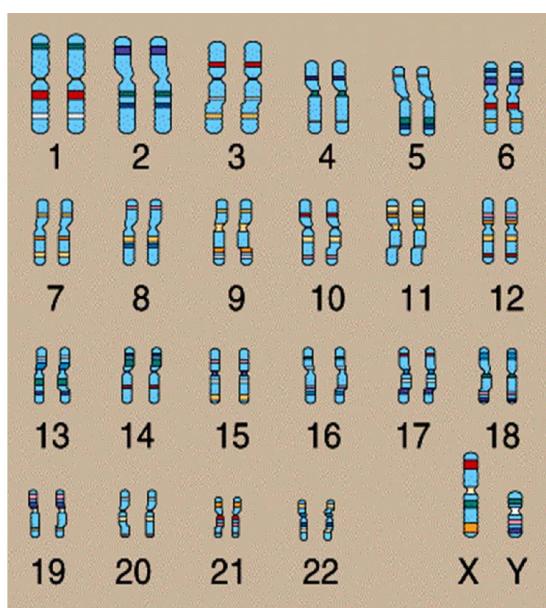
این موجودات از هر کروموزوم یک جفت (دو عدد) دارند. این کروموزوم‌ها از نظر شکل، کدینگ^۱ و جایگاه کدینگ یکسان هستند. به مکان قرار گیری کدها روی کروموزوم لوسی^۲ گفته می‌شود. کدی که در این جایگاه‌ها قرار می‌گیرد، آلل^۳ نام دارد. به بیان دیگر، یک ژن در جایگاه ویژه خود بر روی کروموزوم می‌تواند چینش‌های گوناگونی داشته باشد و به هر یک از این چینش‌ها یا مقادی مختلفی که یک ژن می‌تواند داشته باشد، آلل گفته می‌شود.

به عنوان مثال، فرض کنید در یک گیاه دیپلولئیدی تنها یک ژن مسئول تنظیم رنگ گلبرگ‌ها است و این گیاه می‌تواند دارای گلبرگ‌هایی به رنگ‌های قرمز و سفید باشد. به علاوه رنگ قرمز غالب است. بنابراین مطابق جدول // چهار حالت خواهیم داشت. که تنها در حالتی که هر دو ژنی که بر کروموزوم‌ها قرار دارند سفید باشند، گلبرگ سفید رنگ خواهد بود. حتی در دو حالت قرمز-سفید نیز به دلیل غالب بودن ژن قرمز، رنگ گلبرگ قرمز خواهد بود.

به طور کلی در موجودات دیپلولئیدی، که از هر کروموزوم دو عدد وجود دارد، برای هر ژن دو آلل وجود دارد. اگر مقداری که هر دو ژن مشخص کردۀ‌اند برابر باشد، آن ژن خالص و در غیر این صورت ناخالص خواهد بود. در حالت ژن‌های ناخالص، تعیین نهایی ویژگی از طریق خاصیت غالب و مغلوبی خواهد بود. برای ویژگی‌های گستته از حالت غالب-مغلوبی استفاده می‌شود و برای ویژگی‌های پیوسته، مانند قد، ترکیب مقادیر تعیین کننده خواهد بود.

بر اساس آن چه گفته شد، اگر آلل‌ها یکسان باشند، ژن خالص است و در غیر این صورت، ناخالص. مثلاً اگر در لوسی‌های مربوط به رنگ چشم هر دو آلل مشکی باشند، ژن خالص داریم و اگر یکی آبی و یکی مشکی باشد، ژن ناخالص.

کروموزوم‌های انسان را در شکل زیر مشاهده می‌کنید:

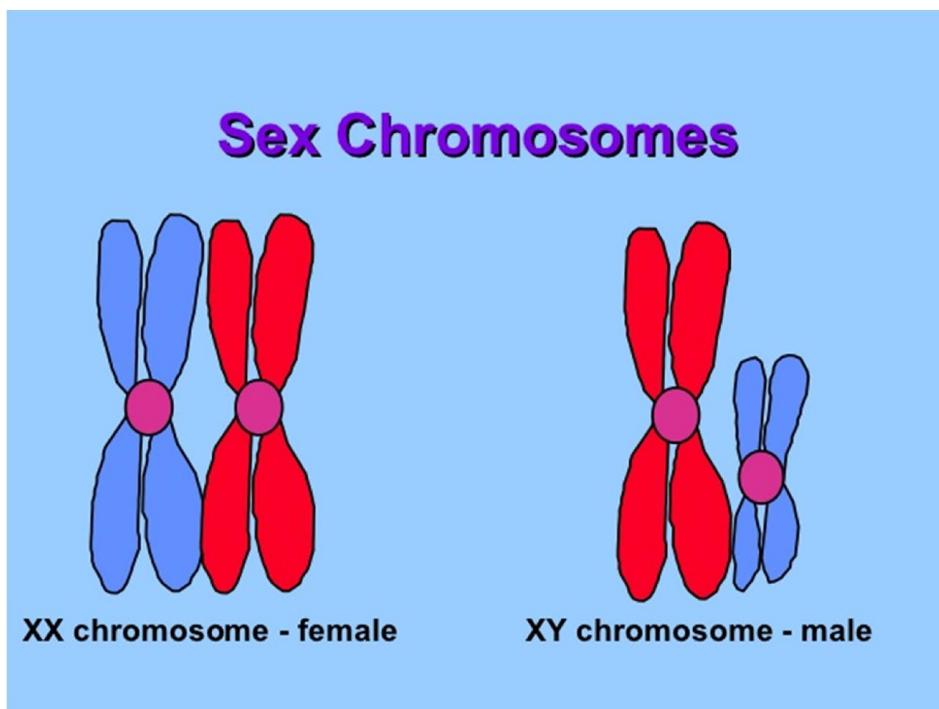


¹ Coding

² Loci

³ Allele

همان طور که گفتیم، موجوداتی که کروموزوم‌های آن‌ها جفت هستند دیپلوبیدی نامیده می‌شوند. در مقابل آن‌ها، موجوداتی که کروموزوم‌های آن‌ها جفت نیستند هاپلوبیدی نام دارند. در انسان ۲۲ کروموزوم اول دیپلوبیدی هستند. کروموزوم ۲۳ م جنسی است و برای زن xx است و همچنان دیپلوبیدی اما برای مرد xy . و در واقع این تنها کروموزومی است که جفت آن مشابه خودش نیست.



در مراحل تولید سلول‌های جنسی، تخمک و اسپرم، این دو کروموزوم از یکدیگر جدا شده و در حین لقاح اسپرم با تخمک، یک نسخه X یا Y از پدر و یک X از مادر به اشتراک گذاشته می‌شود و جنسیت جنین را تعیین می‌کنند. بنابراین تخمک به وجود آمده XX، یک X را از مادر و یک X را از پدر خود می‌گیرد. جدول زیر حالت‌های مختلف تولید فرزند را نشان می‌دهد:

	X	Y
X	XX	XY
X	XX	XY

همان طور که ملاحظه می‌شود، احتمال ایجاد فرزند دختر و پسر یکسان است.

۹-۱ ویژگی‌ها و نحوه به ارت رسیدن آن‌ها

ویژگی‌های فیزیکی در تقسیم بندی کلی به دو دسته زیر تقسیم می‌شوند:

- پیوسته

- گسسته

برای ویژگی‌های گسسته، مانند رنگ چشم، ژن‌های غالب و مغلوب مطرح هستند و برای ویژگی‌های پیوسته مانند قد، ترکیبی از آن‌ها را خواهیم داشت.

آزمایش مندل

در این آزمایش بحث ژن‌های غالب و مغلوب مورد آزمایش قرار گرفت. مندل^۱ دو بذر را که دارای ژن خالص بودند و یکی دارای گلبرگ سفید و دیگری با گلبرگ قرمز، با یکدیگر آمیزش داد. با توجه به غالب بودن رنگ سفید، در نسل اول همگی گل‌ها سفید بودند. اما در حقیقت این گل‌ها ناخالص بودند و به دلیل غالب بودن رنگ سفید، نمود سفید داشتند:

	R	R
W	WR	WR
W	WR	WR

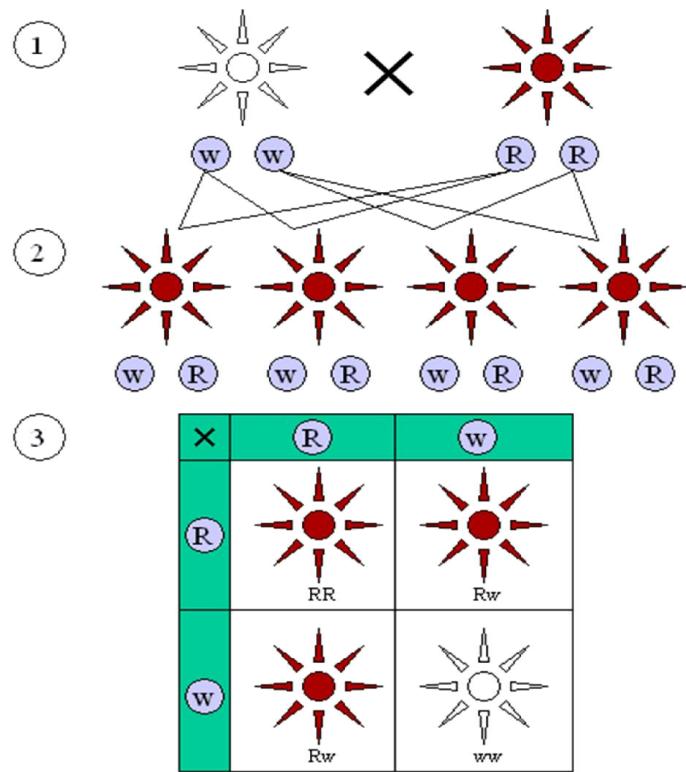
برای تولید نسل دوم، مندل از دانه‌هایی به دست آمده از نسل اول استفاده کرد. به این ترتیب رنگ قرمز در نسل دوم خود را نمایش داد:

	W	R
W	WW	WR
R	WR	RR

همان گونه که مشاهده می‌شود به احتمال ۲۵٪ سفید خالص، به احتمال ۲۵٪ قرمز خالص و احتمال ۵۰٪ سفید ناخالص خواهیم داشت.

به این ترتیب نتیجه می‌شود که یک ویژگی ناخالص می‌تواند چندین نسل نهفته بماند و بعد بروز بیدا کند. به تعبیر دیگر، در موجودات دیپلوفلئیدی، یک ویژگی می‌تواند در طی نسل‌ها در حافظه ژنتیکی باقی بماند و پس از آن ظاهر شود. شکل زیر نیز این آزمایش را نمایش می‌دهد:

^۱ Gregor Mendel



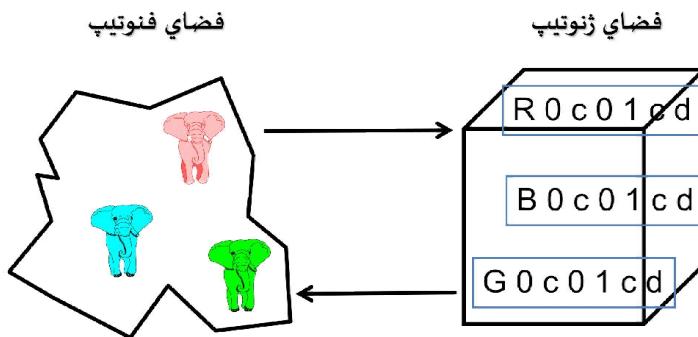
بدیهی است که حالت دیپلولئیدی تنوع بیشتری را ایجاد می‌کند و به همین سبب است که موجودات پرسلوی دیپلولئیدی هستند. سیستم دیپلولئیدی، علاوه بر این که باعث تنوع بیشتر می‌شود، می‌تواند حافظه‌دار باشد و یک ویژگی را چندین نسل در حافظه خود نگه دارد. بر خلاف این وضعیت، در سیستم هاپلولئیدی فقط ژن خالص داریم. با این حال، عمدۀ الگوریتم‌های تکاملی موجود هاپلولئیدی هستند.

۱۰- ژنوتیپ و فونوتیپ

پیش از این گفتیم که اطلاعات ژنتیکی در کروموزوم‌ها ذخیره می‌شوند و کروموزوم‌ها به بخش‌هایی موسوم به ژن تقسیم بندی شده‌اند. در واقع، اطلاعات مربوط به ویژگی‌های فیزیکی در ژن‌ها کد شده‌اند و پس از مراحلی که در ادامه خواهد آمد به خود ویژگی فیزیکی مذکور تبدیل خواهند شد. بر این اساس دو مفهوم ژنوتیپ و فونوتیپ را تعریف می‌کنیم:

ژنوتیپ کد مربوط به ویژگی فیزیکی است که در کروموزوم وجود دارد و مشخص می‌کند مثلاً رنگ چشم باید مشکی باشد.

فونوتیپ به ویژگی‌های فیزیکی مانند مشکی بودن رنگ چشم گفته می‌شود. به عنوان مثال در $A = 1011001$ ، A فونوتیپ و کد سمت راست، ژنوتیپ هستند. شکل زیر نیز فضای ژنوتیپ و فونوتیپ را نمایش می‌دهد:



در طبیعت، ژنوتیپ و فونوتیپ رابطه یک به یک ندارند. به این معنا که یک ژن می‌تواند چندین ویژگی را تحت تأثیر قرار دهد و برعکس. با این حال، در الگوریتم‌های تکاملی بین ژن و ویژگی فیزیکی رابطه یک به یک وجود دارد. اطلاعات لازم برای تولید یک موجود زنده، در DNA آن کد شده است. لذا برای درک بهتر نحوه انجام کدینگ و سپس کد گشایی و تبدیل اطلاعات به ویژگی‌های فیزیکی، به ساختار DNA خواهیم پرداخت.

DNA ۱۱-۱

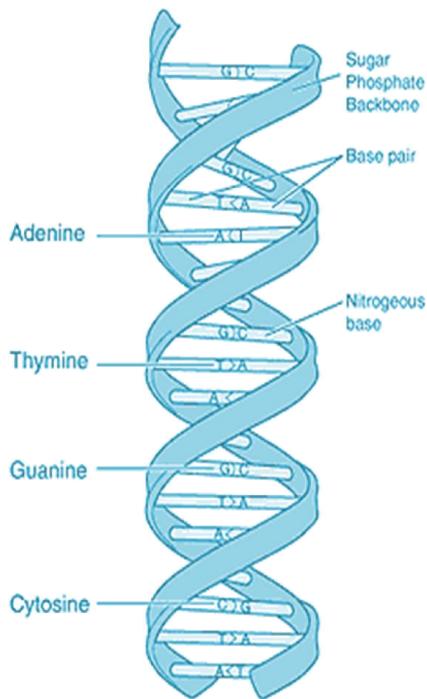
نخستین بار در سال ۱۹۵۳^۳، واتسن^۱ و کریک^۲ ساختمان شیمیایی DNA (دی اکسی ریبونوکلئیک اسید^۴) را کشف کرده و به خاطر آن جایزه نوبل گرفتند. دی اکسی ریبووز^۳ نام قندی است که در ساختمان شیمیایی DNA وجود دارد. نوکلئیک به معنای داخل هسته بودن است و اسید نیز خاصیت اسیدی را نمایش می‌دهد. DNA پلیمری است که از زنجیرهای از مونومرها تشکیل شده است. مونومر DNA از سه قسمت تشکیل شده است: باز، قند و فسفات. که شکل نمادین آن در زیر آمده است:

¹ Watson

² Crick

³ Deoxyribonucleic acid

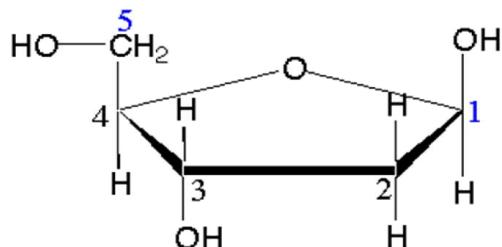
⁴ Deoxyribose



اندامک‌ها داخل سلول معمولاً چهار شکل دارند: خطی، دو بعدی، سه و چهار بعدی. در حالت عادی این اندامک‌ها در حالت چهار بعدی خود هستند.

DNA در حالت خطی به شکل یک نرده‌بان است. اگر این نرده‌بان را از وسط بیپچانیم، دو بعدی می‌شود. اگر آن را به صورت حلقوی بیپچانیم سه بعدی می‌شود و با پیچش بعدی، به شکل چهار بعدی می‌شود. سانتومر محلی است که دو رشته به یکدیگر متصل می‌شوند.

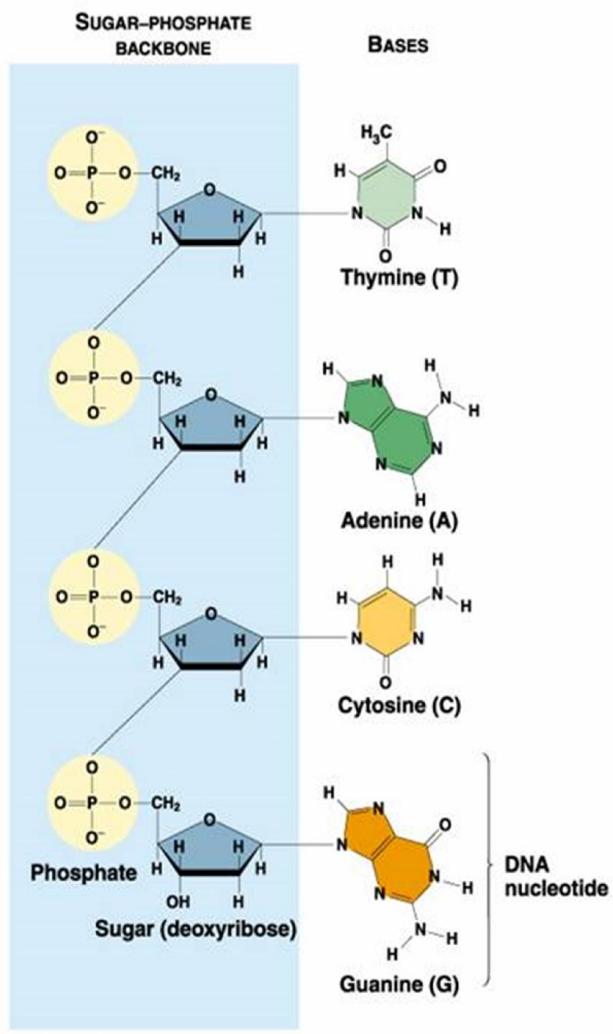
نرده‌بان فوق‌الذکر که در شکل مشاهده می‌کنید، در واقع پلیمر گفته شده برای DNA است و از مونومرهای تشکیل شده که هر یک شامل قند، فسفات و باز هستند. قند، هم به باز و هم به گروه فسفات متصل است. شکل زیر گروه قندی را نمایش می‌دهد:



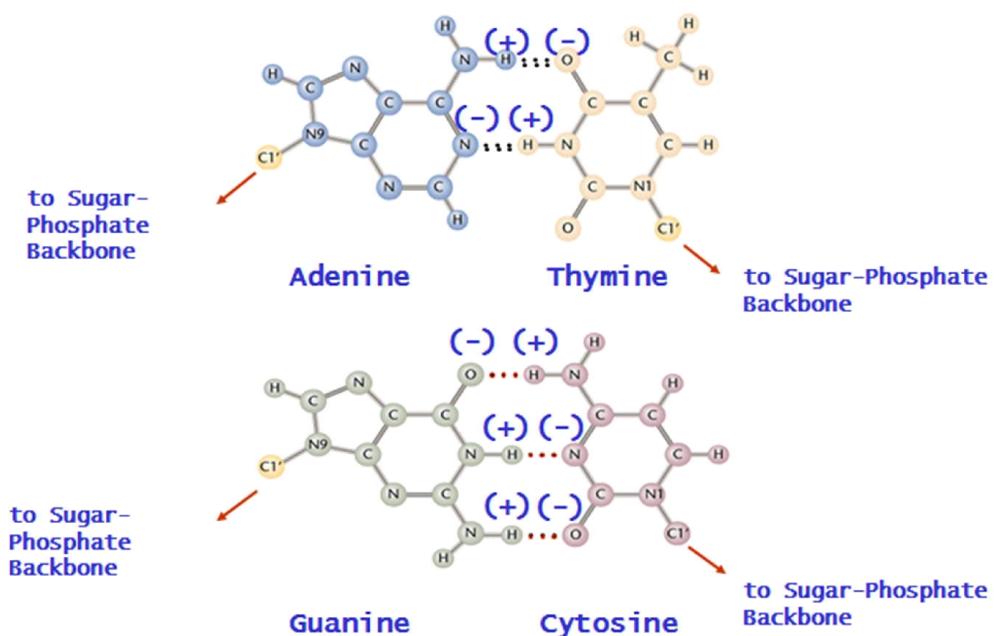
آن چه اهمیت دارد بازها هستند:

- بازهای پورین: شامل آدنین (A) و گوانین (G)
- بازهای پیریمیدین: شامل سینوز (C)، تیمین (T) و یوراسیل (U)

البای ژنتیک A، G، C و T است. که آن‌ها را در شکل زیر مشاهده می‌کنید:



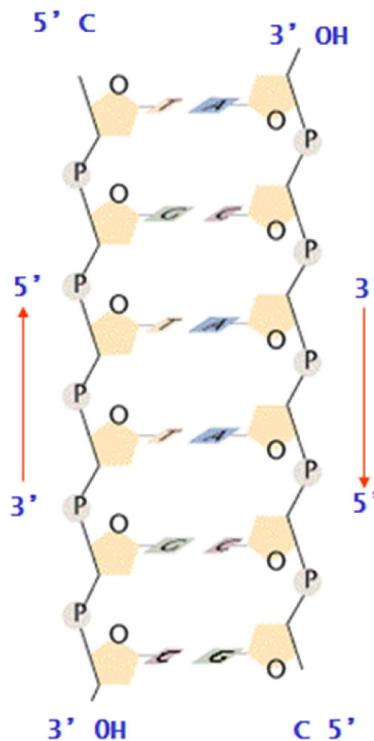
پیوندهای نیتروژنی به صورت A-T و G-C هستند که در شکل زیر مشاهده می‌کنید. بازهای مذکور دو به دو مکمل هستند: T و A مکمل یکدیگرند و C و G مکمل هم.



این بازها، همان طور که گفته شد، در واقع حروف الفبای ژنتیک موجودات زنده و از جمله انسان هستند و آن را کد می‌کنند. بازهای آدنین و تیمین در دو نقطه و بازهای گوانین و سیتوزین در سه نقطه پیوند شیمیایی برقرار می‌کنند. شکل نردهبانی DNA از جفت شدن این بازها ایجاد می‌شود. اتصال طولی مابین گروه‌های فوسفات و قند است. اتصال‌های عرضی، از به هم پیوستن باز و قند ایجاد می‌شوند.

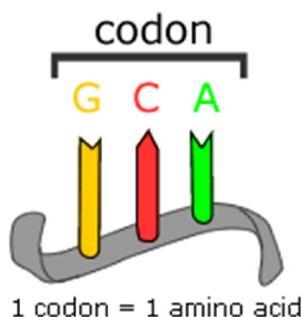
قند دی اکسی ریبوز ۵ کربن دارد که از ۱ تا ۵ شماره گذاری می‌شوند. این قند از محل کربن شماره ۳ به فسفات پایینی و از محل کربن شماره ۵، به بالایی متصل می‌شود. DNA ها جهت دار هستند و جهت آن‌ها با شماره کربن‌ها مشخص می‌شود. جهت، از کربن ۳ به سمت کربن ۵ است. درواقع یک سر DNA از کربن ۳ و سر دیگر از کربن ۵ تشکیل شده است.

در واقع، اگر یک سمت از این نردهبان، ژنی را کد کند (بازهایش مشخص باشند)، در سمت مقابل اطلاعات جدیدی نداریم و صرفاً شامل بازهای مکمل نیمه اول خواهد بود.



۱۲-۱ گُدن

هر مجموعه سه حرفی از حروف الفبای ژنتیکی که در بالا معرفی شدند، یک کدن^۱ را تشکیل می‌دهند و هر کدن یک آمینواسید را کد می‌کند.



همان گونه که اشاره شد، در کدن، سه جایگاه برای حروف وجود دارد. به این ترتیب با وجود چهار حرف در الفبای ژنتیک، تعداد حالت‌های ممکن برای کدن‌ها، 4^3 است. با این حال، تنها ۲۱ آمینواسید وجود دارد. دلیل این امر این است که برخی از حالات مذکور به دلیل جلوگیری از جهش و خطا حذف شده یا یکسان تلقی می‌شوند که در ادامه بیشتر به آن خواهیم پرداخت.

جدول زیر آمینواسیدهای معادل هر یک از کدن‌ها را نمایش می‌دهد:

¹ Codon

		Second letter				
		U	C	A	G	
First letter	U	UUU } Phe UUC UUA } Leu UUG }	UCU } Ser UCC UCA UCG }	UAU } Tyr UAC UAA Stop UAG Stop }	UGU } Cys UGC UGA Stop UGG Trp }	U C A G
	C	CUU } Leu CUC CUA CUG }	CCU } Pro CCC CCA CCG }	CAU } His CAC CAA } Gln CAG }	CGU } Arg CGC CGA CGG }	U C A G
	A	AUU } Ile AUC AUA } Met AUG }	ACU } Thr ACC ACA ACG }	AAU } Asn AAC AAA } Lys AAG }	AGU } Ser AGC AGA } Arg AGG }	U C A G
	G	GUU } Val GUC GUA GUG }	GCU } Ala GCC GCA GCG }	GAU } Asp GAC GAA } Glu GAG }	GGU } Gly GGC GGA GGG }	U C A G
Third letter						

جهش در طبیعت خطا محسوب می‌شود و در چند مرحله از رخداد آن جلوگیری می‌شود. یک راه، وجود حالت‌های اضافی^۱ برای آمینواسیدها است که در بالا گفته شد. به این ترتیب که موارد مشابه که احتمال اشتباہ شدن با یکدیگر را دارند، یک حالت در نظر گرفته می‌شوند.

کدون شروع، دارای کد AUG و کدون‌های پایان، دارای کدهای UAA، UAG و UGA می‌باشد. همان طور که قبلاً اشاره شد، زنجیره‌ای از آمینواسیدها پروتئین‌ها را تشکیل می‌دهند. آمینواسیدها همان کدون‌های بین کدون شروع و پایان هستند.

به نیمی از نردبان DNA، ریبونوکلئیک اسید^۲ (RNA) گفته می‌شود. در RNA به جای باز T که در DNA به کار می‌رود، باز U وجود دارد. RNA از روی DNA کپی می‌شود و اگر در روند کپی خطایی رخ دهد می‌بایست اصلاح شود.

RNA مشابه DNA است با این تفاوت که:

- زنجیره شکر-فسفات آن یک اکسیژن اضافی دارد.
- به جای باز تیمین (T) باز یوراسیل (U) دارد.
- از DNA ناپایدارتر است و طبعاً ساده‌تر از بین می‌رود.

¹ redundancy

² Ribonucleic acid

۱۳-۱ ژن

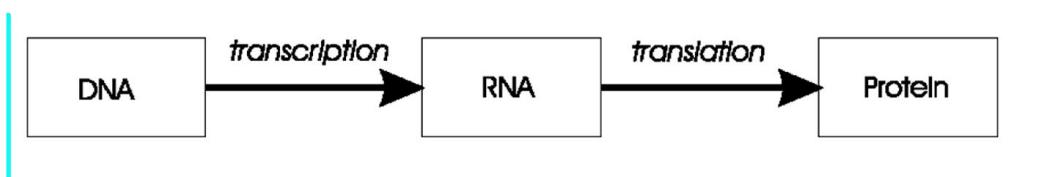
از کنار هم قرار گرفتن تعدادی کدن به وجود می‌آید. ابتدا و انتهای ژن همواره با کدن آغاز و پایان مشخص می‌گردد.

ژن، زنجیره‌ای از آمینواسیدهایی که کنار هم قرار می‌گیرند را مشخص می‌کند. زنجیره آمینواسیدها پروتئین‌ها را تشکیل می‌دهند که ویژگی‌های فیزیکی موجود زنده را می‌سازند. سلول در واقع کارخانه تولید پروتئین است. پروتئین نیز مانند DNA چهار شکل خطی، دو، سه و چهار بعدی دارد. پروتئین‌ها روی اسکلت‌های RNA سوار شده و ویژگی‌های فیزیکی را می‌سازند. پروتئین‌های مختلفی که باید در یک موجود وجود داشته باشند توسط ژن‌ها کد می‌شوند.

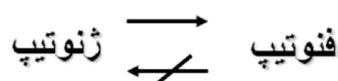
در آمینواسیدها ۲۱ کد می‌تواند وجود داشته باشد (با احتساب کد شروع و بدون کد پایان). لذا، با توجه به این که محدودیتی در کدهای یک ژن نداریم، تعداد بسیار زیادی پروتئین می‌توانیم داشته باشیم.

۱۴-۱ چگونگی تولید پروتئین توسط سلول و ایجاد ویژگی‌های فیزیکی

DNA نقشه هستی است. تمام موجودات عالم دارای ژن‌ها و کدهای یکسانی هستند. وجه تمایز آن‌ها، این است که چه پروتئینی در هر کدام باشد. نوع این پروتئین را DNA تعیین می‌کند. درواقع سلول وظیفه پروتئین سازی را دارد. همه قسمت‌های DNA ژن نیست و بعضی از قسمت‌های آن کنترلی است. در ادامه به مراحل تولید پروتئین و در نتیجه ویژگی‌های فیزیکی، بر اساس اطلاعات DNA خواهیم پرداخت. در واقع، هدف بررسی چگونگی انتقال ژن‌ها به فرزندان است. سلول از روی ژن فعال، پروتئین مربوطه را ساخته و ویژگی‌های فیزیکی متناظر را تولید می‌کند. شکل زیر روند مربوطه را به صورت کلی نمایش می‌دهد.



لامارک معتقد بود که این روند می‌تواند برگشت پذیر باشد، اما پیشرفت علم نشان داد که فرآیند فوق یک سویه است:



چند لایه حفاظتی برای جلوگیری از تغییر اطلاعات DNA وجود دارد. در انتقال DNA به فرزندان نیز، جز تغییرات خاص، نباید تغییری در DNA رخ دهد. به منظور حفاظت، DNA داخل هسته قرار دارد و هیچ عملیات شیمیایی درون هسته رخ نمی‌دهد. بنابراین باید از ژن فعال درون هسته یک کپی ساخته شود و به درون سلول بیاید. سپس از روی آن، پروتئین‌هایی برای خود سلول یا بیرون آن ساخته می‌شود.

دیوکسی ریبوز، قند DNA، جایگاه خالی برای فعالیت شیمیابی ندارد و به این ترتیب نمی‌تواند فعالیت شیمیابی داشته باشد. اما برخلاف آن، ریبوز تمایل بالایی به فعالیت شیمیابی دارد. به این ترتیب، از روی DNA فعال در هسته، RNA یی تولید می‌شود که کپی قسمتی از DNA است. این RNA وارد سلول شده، ترجمه می‌شود و پروتئین را می‌سازد. لازم به ذکر است که امکان عبور از غشاء هسته را ندارد، اما RNA کپی شده می‌تواند از آن عبور کرده و وارد فضای سلول شود.

همان طور که گفته شد، ژن از تعدادی کدن تشکیل شده است که مایبن کدن آغاز و پایان قرار دارند. RNA از کدن آغاز شروع کرده و تا کدن پایان را کپی می‌کند.

باید توجه داشت که فرآیند تبدیل ژنتیپ به فونوتیپ یک طرفه است. به عنوان مثال اگر کسی با ورزش عضلات خود را تقویت کند، این ویژگی وارد ژن او نشده و قابلیت انتقال به فرزندانش را نخواهد داشت.

نحوه انجام ترجمه و تولید پروتئین

در این بخش ابتدا به انواع تقسیم‌های سلولی خواهیم پرداخت. دو نوع تقسیم سلولی وجود دارد:

- تقسیم میوز (جنسی)
- تقسیم میتوز (عادی)

تقسیم سلولی جنسی، مربوط به سلول‌های جنسی است که در موجود نر اسپرم و در موجود ماده تخمک را تولید می‌کند.

زیگوت با تقسیم سلولی، ابتدا سلول‌های بنیادی جنینی و سپس سلول‌های بنیادی بالغ و تخصصی را تولید می‌کند. دو دسته اخیر به صورت مداوم با تقسیم سلولی تجدید می‌شوند.

در هر دو تقسیم فوق، مساله مهم این است که ساخت کپی (همانند سازی از روی DNA) باید به گونه‌ای انجام گیرد که در اطلاعات ژنتیکی تغییر نکند و به تعبیر دیگر سلول دختر، همان اطلاعات سلول مادر را داشته باشد.

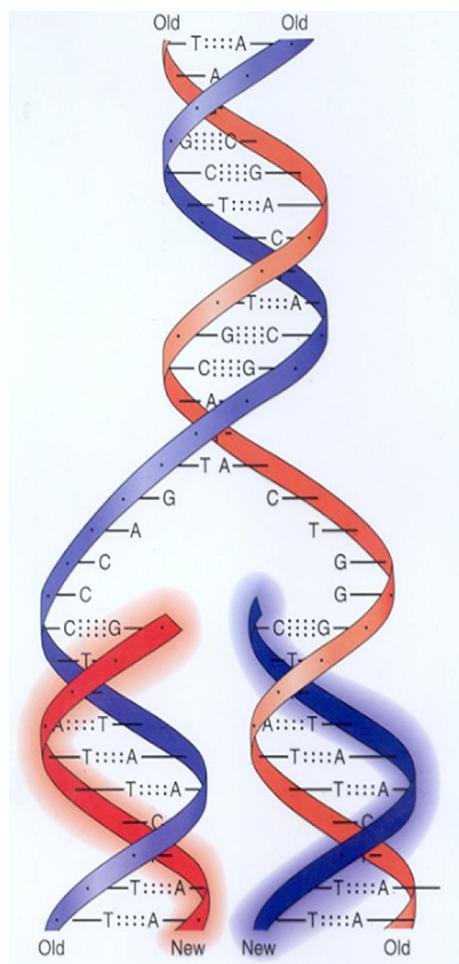
در سلول پروتئین مایعی به نام آنزیم وجود دارد که نقش کاتالیزوری دارد. کاتالیزور باعث تسريع واکنش شیمیابی شده و در عین حال خود در آن واکنش شرکت نمی‌کند. برای انجام عملیات همانند سازی شش آنزیم لازم است. نحوه نامگذاری آنزیمهای آنها به این صورت است که نام آنها از دو بخش تشکیل شده است؛ بخش اول عملیاتی که انجام می‌دهند را نشان می‌دهد و بخش دوم واژه "آز" است.

برای همانند سازی DNA، ابتدا لازم است که ساختار نرdbانی آن مانند یک زیپ از هم باز شود. این کار توسط آنزیمی به نام هلیکاز^۱ انجام می‌شود. پیش‌تر گفتیم که عملیات از ۳ به ۵ است. پلیمر کردن در این جهت توسط

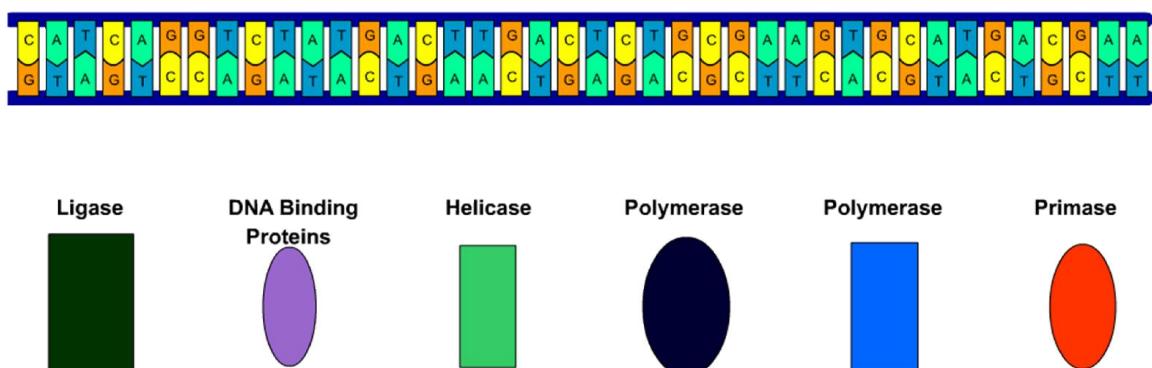
^۱ Helicase

آنزیم پلیمراز انجام می‌شود. برای این آنزیم، پریماز^۱ نشان دهنده نقطه شروع است. کار این آنزیم تولید مونومر مکمل و پلیمر کردن و پیش رفتن روی نیمه باز شده DNA است.

برای پلیمر شدن قسمت مقابل، لازم است DNA به مقدار کافی باز شود. در این زمان، این بخش نیز در جهت ۳ به ۵ خود (که خلاف جهت بخش مقابل است) پلیمر می‌شود.

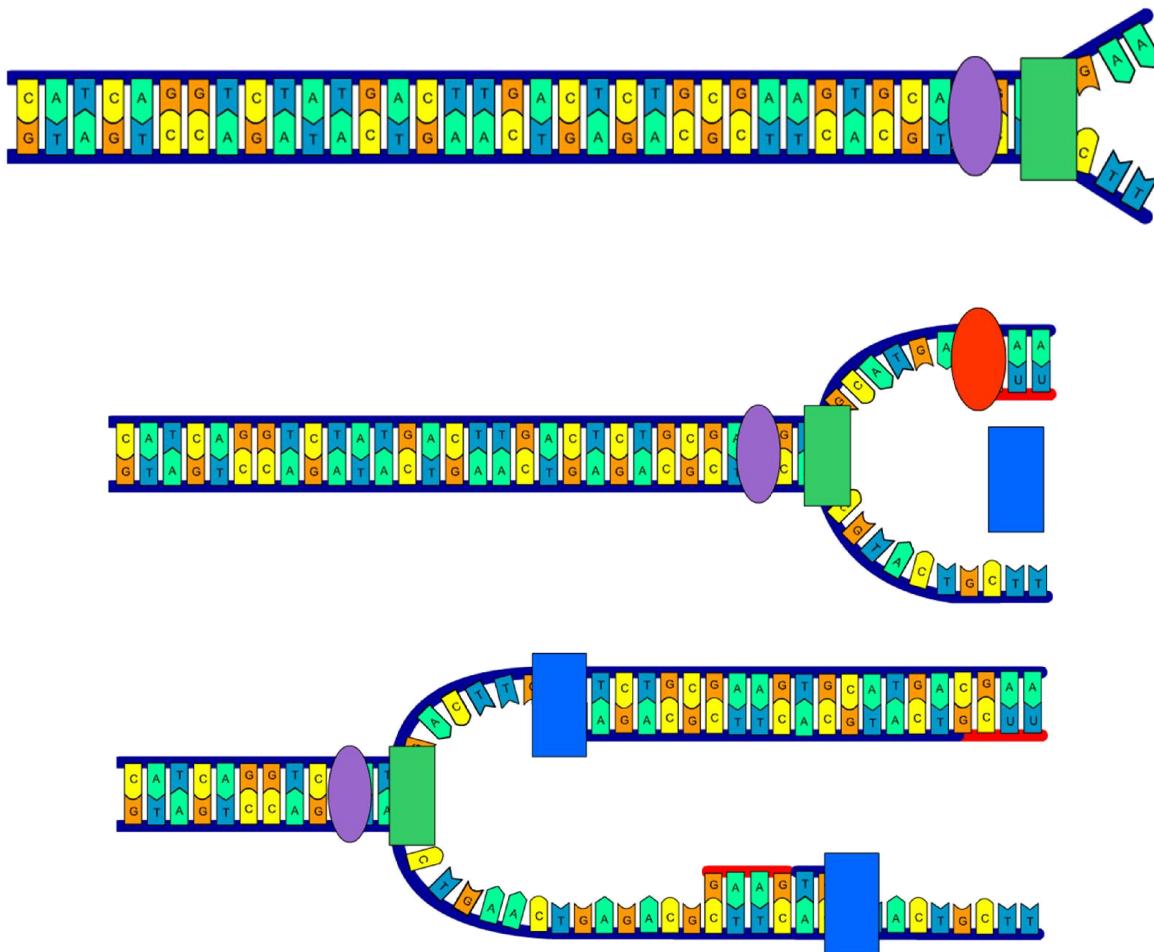


مراحل همانند سازی DNA در مجموعه شکل‌های زیر نمایش داده شده است. در شکل زیر رشته DNA و آنزیم‌ها و نمادهایی که برای آن‌ها در نظر گرفته شده است نمایش داده شده است:

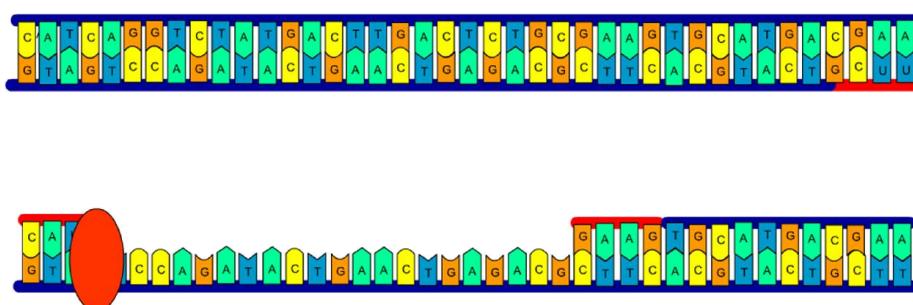


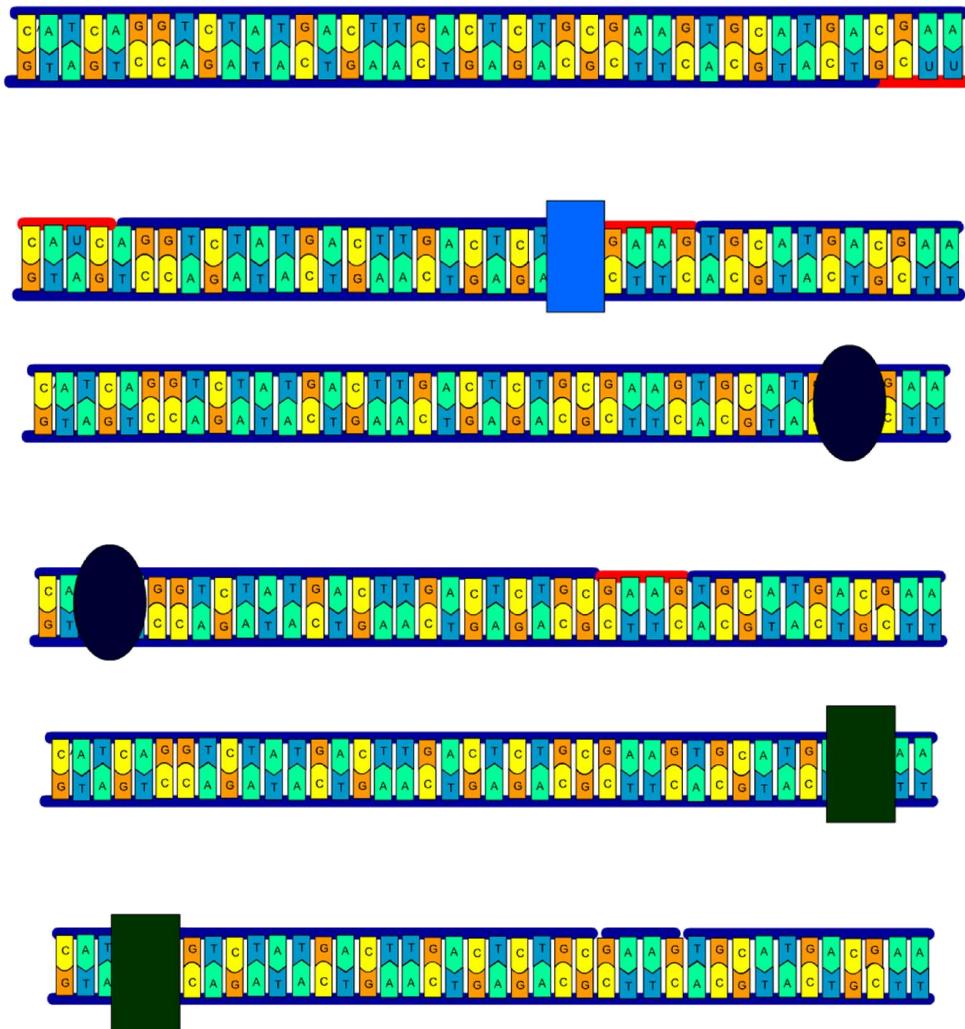
¹ perimase

در ادامه مراحل باز شدن رشته، آنزیمهای دخیل در آن و همانند سازی آن را که در بالا توضیح داده شد می‌بینیم:



همان طور که مشاهده می‌شود، با توجه به جهت دار بودن همانند سازی، لازم است رشته تا جایی باز شود تا امکان همانند سازی رشته پایینی که در خلاف جهت رشته بالایی صورت می‌گیرد، فراهم شود.





خطاهای

اشاره شد که در DNA، U داریم اما در RNA، به جای آن T داریم. علاوه بر این مورد که باید اصلاح شود، ممکن است چهش‌هایی داشته باشیم. لذا این جا آنزیم دیگری مجدداً همه موارد را کنترل کرده و در صورت لزوم اصلاح می‌کند. پیش‌تر گفتیم که یک روش جلوگیری از رخداد خطا redundancy است. روش فعلی نیز در کنار روش اول بروز خطا را به حداقل می‌رساند.

بعد از آن چه گفته شد، دو کاملاً مشابه با بازهای مکمل خواهیم داشت. باید توجه داشت که ممکن است روشن شدن یک ژن، سبب خاموش شدن ژن‌های دیگر شود. در سرطان آن جا که ژن باید تقسیم سلولی را متوقف کند، نمی‌کند.

نحوه تولید پروتئین از روی RNA

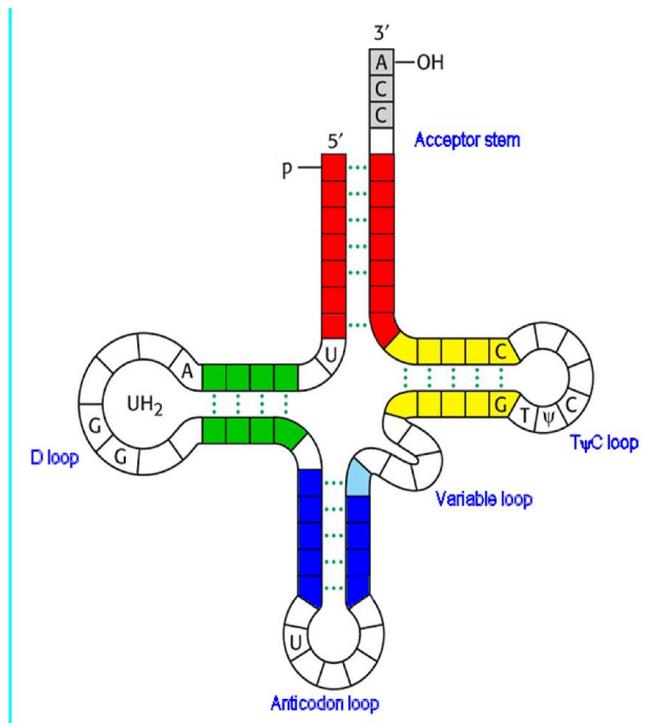
در واقع کد ژنتیکی در RNA وجود دارد و DNA از قرار گرفتن مکمل RNA در کنار آن ایجاد می‌شود. سیگما فاکتور تشخیص می‌دهد که پیش برنده^۱ از کجا آغاز می‌شود و از آن جا DNA باز شده و مکمل‌هایشان ساخته می‌شود. در انتهای ژن تعداد زیادی A داریم که در مقابل آن U قرار می‌گیرد و چون در RNA به جای T، U داریم لازم است برای تبدیل به DNA، این‌ها اصلاح شوند.

مکمل DNA به صورت یک طرفه روی RNA کپی می‌شود و از هسته جدا شده و برای پروتئین سازی وارد سلول می‌شود. به این RNA که حامل کد ژنتیکی است حامل^۲ (mRNA) گفته می‌شود.

برای رونویسی از روی RNA، آنزیم RNA-پلیمراز لازم است. این آنزیم می‌تواند پیش برنده (شروع ژن) را شناسایی کرده، روی آن می‌نشیند، رشته را باز می‌کند و پلیمر یک طرفه (نصف نردبان) را با مکمل کردن تولید می‌کند.

توجه داشته باشید که در همانند سازی از روی DNA، آنزیم DNA-پلیمراز^۳ داشتیم و نردبان دو طرفه ساخته می‌شد.

RND مدل^۴: مانند mRNA از روی ژن درست می‌شود. در حالت یک بعدی، مثل نردبان یک طرفه است. به علت وجود بازهای مکمل، در حالت دو بعدی به صورت زیر است:



¹ promoter

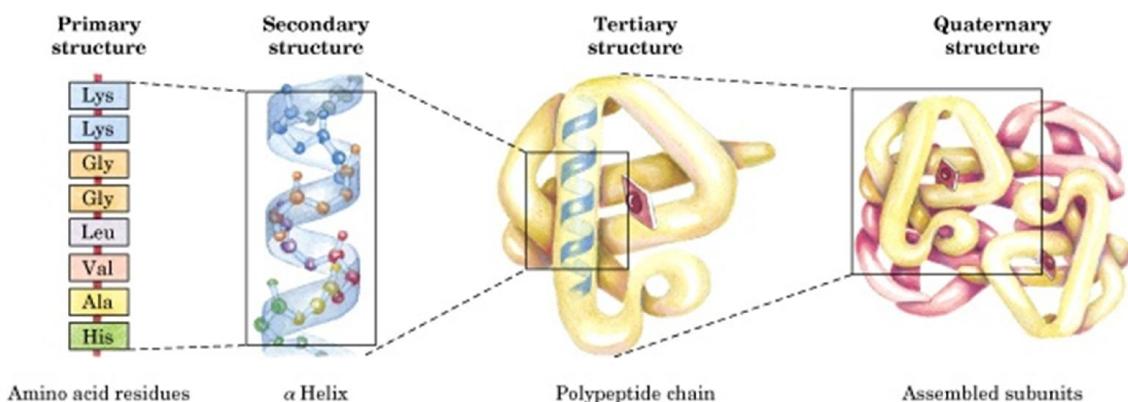
² messenger

³ polymerase

⁴ transfer

حلقه‌هایی که دارد، آنتی کدن‌های خاصی را دارند و به این ترتیب توان اتصال و حمل موارد لازم را خواهد داشت.
شکل سه بعدی آن، مانند شبدر خواهد بود.

پروتئین‌های تولیدی هم می‌توانند یک، دو، سه یا چهار بعدی باشند که در شکل زیر مشاهده می‌کنید:



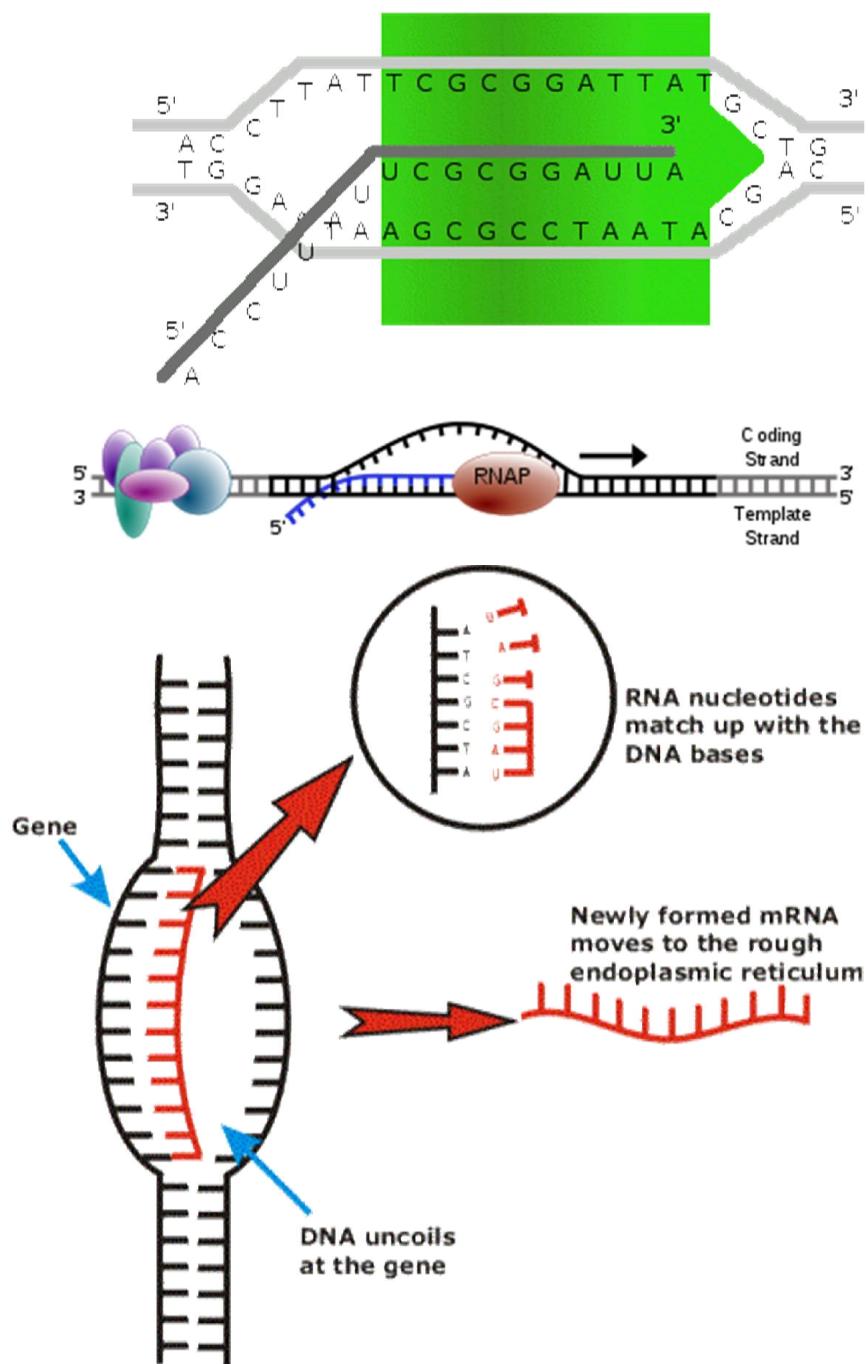
rRNA: در حالت یک بعدی نردهان یک طرفه و در حالت سه و چهار بعدی، اسکلت ریبوزوم را می‌سازد؛ که پروتئین‌هایی که روی آن قرار می‌گیرند ریبوزوم را می‌سازند. ریبوزوم خود نیز در پروتئین سازی به کار می‌رود.
mRNA کد ژن را می‌آورد. تعدادی tRNA وجود دارند که بر اساس آنتی کدی که دارند آمینواسیدها را حمل می‌کنند.

rRNA از دو قسمت تشکیل شده است که روی هم قرار دارند و کار پروتئین سازی را بر عهده دارند.
در ریبوزوم سه جایگاه داریم که به سایتها^۱ E، P و A موسومند. اگر آنتی کدن با کدی که آن جا قرار دارد مکمل شود، سلول RNA و یا پروتئین تولید می‌کند. مواد اولیه مورد نیاز، مونومرهایی از جنس باز، فسفات و قند هستند.

تعداد ژن‌ها روی DNA خیلی زیاد است. بنابراین بسته به این که کدام ژن روشن (فعال) است، پروتئین مربوطه ساخته می‌شود.

به طور خلاصه تولید پروتئین طی سه مرحله اصلی انجام می‌شود که در ادامه خواهد آمد.
مرحله اول، کپی برداری RNA از روی ژن DNA است که به آن رونویسی گفته می‌شود و در هسته سلول به صورت زیر صورت می‌گیرد:

^۱ Site



ابتدا RNA-پلیمراز محل شروع ژن را تشخیص می‌دهد. سپس، از محل تعیین شده، DNA مانند یک زیپ باز می‌شود. نوکلئیدهای مکمل از سمت ۵ به ۳ ایجاد می‌شوند. به این ترتیب، زیپ بسته می‌شود و این کار تا رسیدن به کدون پایان ادامه می‌یابد و mRNA ساخته می‌شود. قواعد حاکم بر فرآیند رونویسی نیز همان قوانین جفت شدن بازه است که در همانندسازی DNA به کار می‌رود. تنها تفاوت این است که در مقابل دئوكسی نوکلئوتید A (آدنین‌دار) در DNA، ریبونوکلئوتید U

(یوراسیل‌دار) در RNA قرار می‌گیرد. RNA-پلیمراز mRNA و DNA جدید ساخته شده، پس از رونویسی جایگاه پایان رونویسی، از یکدیگر جدا می‌شوند و مولکول mRNA برای مرحله بعد یعنی ترجمه، آزاد می‌شود. در اینجا، چون رونویسی از روی بخش ژن DNA انجام شده است، حاصل mRNA است. روند رونویسی tRNA نیز به همین ترتیب است با این تفاوت که برای هر کدام از بخش‌های خاصی از DNA رونویسی می‌شود.

در مرحله دوم mRNA از هسته خارج می‌شود. مرحله سوم ساخت پروتئین از روی mRNA است که ترجمه نام دارد. ترجمه فرآیندی است که در آن اطلاعات موجود در RNA برای ساخت پروتئین‌ها مورد استفاده قرار می‌گیرد.

برای برخی از ژن‌ها محصول نهایی همان RNA ای است که در مراحل قبل تولید شده است، اما برای بسیاری مانند mRNA‌ها این پایان کار نیست. mRNA‌ها همراه خود کد یک یا چند توالی پروتئین را حمل می‌کنند. در هر توالی mRNA اطلاعات ژنتیکی به وسیله کدون‌هایی که پشت سر هم قرار دارند ذخیره شده است. در فضای سیتوپلاسم برای هر کدام از این کدون‌ها یک tRNA با یک کدون مکمل وجود دارد که همراه خود آمینو اسید مشخصی را حمل می‌کند. سپس این آمینواسیدها به وسیله یک ریبوزوم و همچنین ترتیب کدون‌ها بر روی mRNA به هم متصل شده و توالی پروتئینی به دست می‌آید. از روی هر رشته mRNA می‌توان تعداد زیادی توالی پروتئین ساخت.

در خصوص ترجمه، مسائل زیر وجود دارند.

در ترجمه، توالی نوکلئوتیدی mRNA به توالی آمینواسیدی در رشته پلی پپتیدی ترجمه می‌شود. ترجمه یک فرآیند آنزیمی است و نیاز به صرف انرژی دارد و هر آنزیم نوعی پروتئین است که کار سلول در هر لحظه را تعیین می‌کند. فرآیند ترجمه در سیتوپلاسم سلول‌ها رخ می‌دهد. برای انجام عمل ترجمه در سلول به mRNA، ریبوزوم، tRNA، اسیدهای آمینه، آنزیم و عوامل پروتئینی و همچنین انرژی نیاز است.

پروتئین سازی در ریبوزوم‌ها صورت می‌گیرد که از tRNA و پروتئین تشکیل شده است و tRNA اسیدهای آمینه را به ریبوزوم حمل می‌کند و mRNA اطلاعات را به ریبوزوم حمل می‌کند. (در پروتئین سازی هر سه نوع RNA شرکت دارند)

برای هر اسید آمینه حداقل یک نوع tRNA وجود دارد (یعنی بعضی اسیدهای آمینه توسط بیش از یک نوع tRNA حمل می‌شوند).

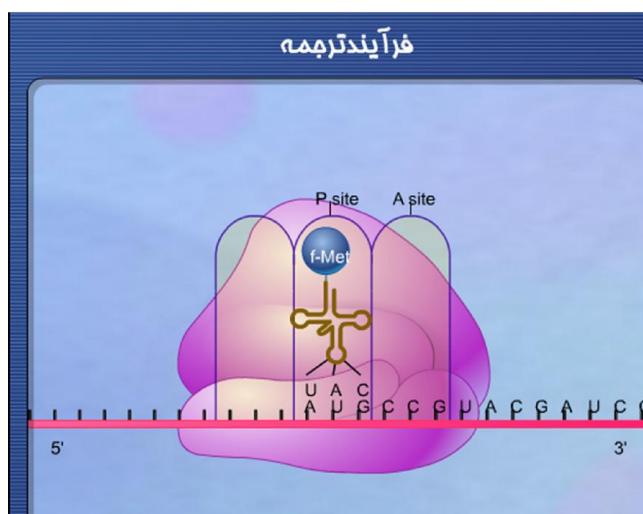
ریبوزوم دارای دو زیر واحد کوچک و بزرگ است که زیر واحد کوچک آغازگر فرآیند ترجمه است و خود دارای دو جایگاه به نام‌های A برای آمینواسید و P برای پلی پپتید در حال ساخت است. شناسایی mRNA به عهده زیر واحد کوچک است. زیر واحد بزرگ دارای کانالهایی است که اسیدهای آمینه‌ها را در دسته‌هایی که مخصوص آنها هستند بگذارد. آنها از یکیواردشده‌هورشت‌پلی‌پپتیدیتازه‌ساختازدیگری خارج می‌شوند.

ترجمه دارای سه مرحله‌ی آغاز، ادامه و پایانی است.

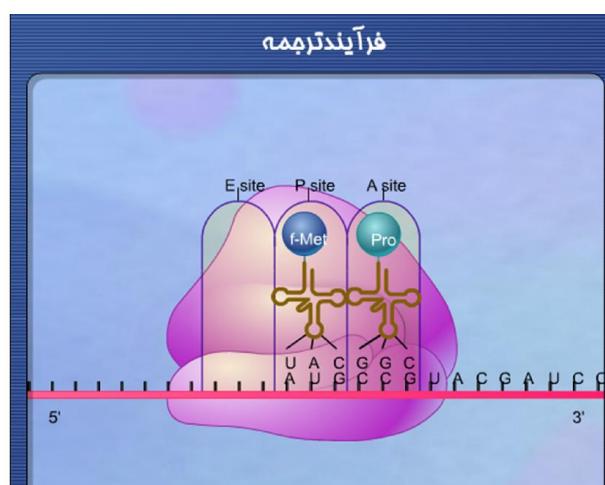
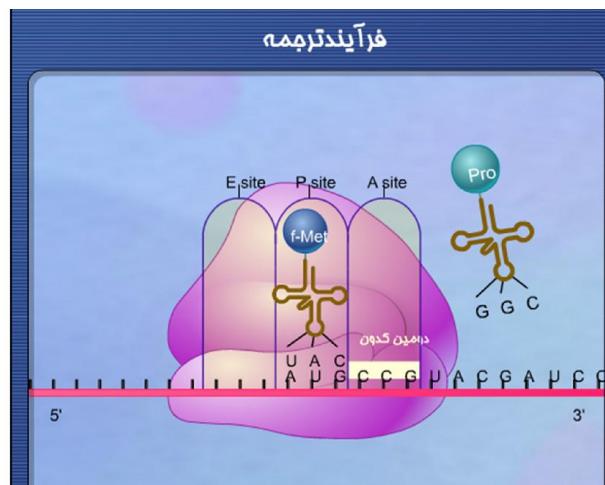
در آغاز ترجمه، دو زیر واحد کوچک و بزرگ ریبوزوم از یکدیگر جدا هستند. بخش کوچک ریبوزوم با شناسایی کدون آغاز(AUG)، بر روی mRNA به آن متصل می‌شود. پس از اتصال صحیح جزء کوچک ریبوزوم، اولین اسید آمینه که متیونین نام دارد از سیتوپلاسم فرا خوانده می‌شود و در جایگاه P ریبوزوم قرار می‌گیرد و با کدون آغاز (AUG) رابطه مکملی برقرار می‌کند.



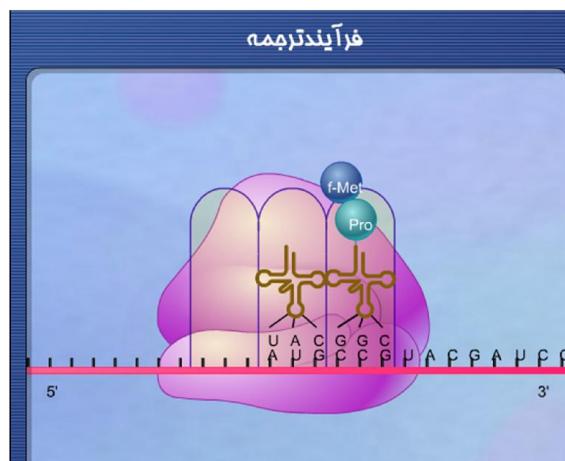
سپس زیر واحد بزرگ ریبوزوم به زیر واحد کوچک متصل می‌شود.



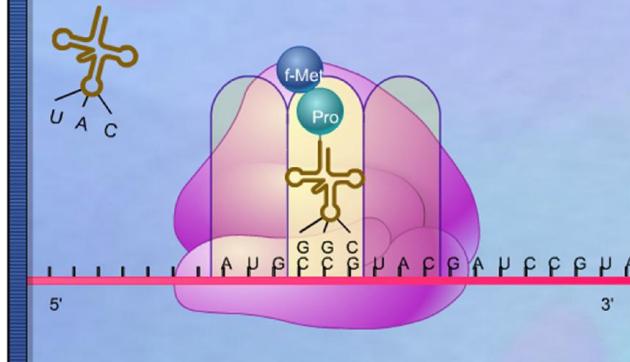
در مرحله‌ی ادامه tRNA_i بعدی در جایگاه A قرار می‌گیرد و دومین اسید آمینه با اولین اسید آمینه پیوند پیتیدی برقرار می‌کند.



سپس tRNA_i موجود در جایگاه P آزاد شده و جایگاه P خالی می شود و tRNA_i جایگاه A به همراه رشته i پلی پپتیدی به اندازه i یک کدون جابجا شده و جایگاه P را اشغال می کندو جایگاه A خالی می شود.

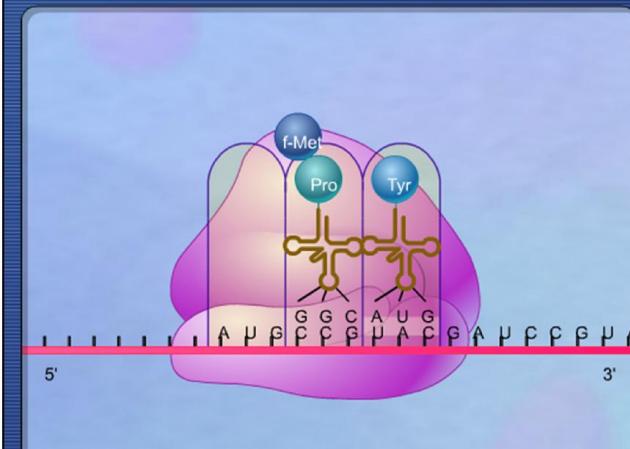


فرآیند ترجمه

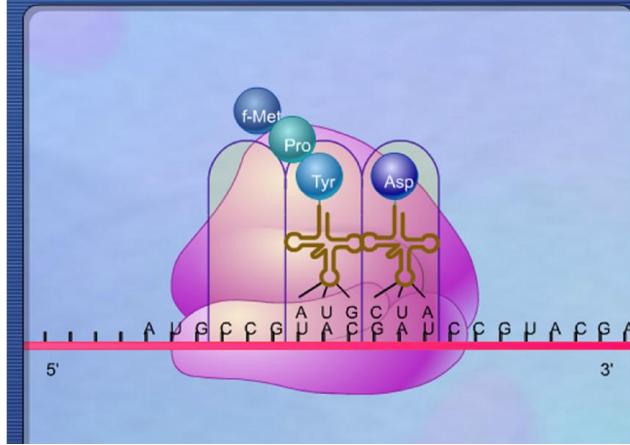


و tRNA سوم در جایگاه A قرار می‌گیرد.

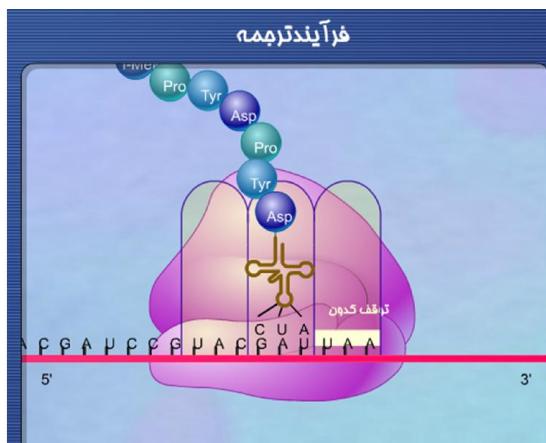
فرآیند ترجمه



فرآیند ترجمه



این روند تا رسیدن به کدون پایان ادامه دارد:



در مرحله‌ی پایان با قرار گرفتن یکی از کدون‌های پایان در جایگاه A ریبوزوم، عامل پایان ترجمه در جایگاه A قرار می‌گیرد و زیر واحد‌های ریبوزوم و mRNA و رشته‌ی پلی‌پپتیدی ساخته شده از هم جدا می‌شوند و ترجمه به اتمام می‌رسد.

ترجمه از انتهای^۱ شروع شده در انتهای^۲ خاتمه می‌باید. به این معنی که کدون AUG در انتهای^۳ رشته mRNA و کدون‌های خاتمه در انتهای^۴ قرار می‌گیرند.

از DNA به عنوان کامپیوتر استفاده شده است. به این ترتیب که هر DNA را در یک لوله آزمایش قرار داده‌اند و از آن‌ها برای انجام پردازش‌های موازی استفاده کرده‌اند. علاوه بر کامپیوتر سلولی^۲ و کامپیوتر غشایی^۳ نیز در تحقیقات مورد بررسی و استفاده هستند.

۱۵- انتقال ویژگی‌ها به فرزندان

گفتیم که ویژگی‌های فیزیکی به واسطه پروتئین سازی از روی کدها ایجاد می‌شوند. پیش‌تر نیز اشاره شد که لامارک پیش از داروین اعتقاد داشت که اثر فونوتیپ و ژنوتیپ دو طرفه است. با این حال بعدها ثابت شد که این موضوع صحت ندارد.

تقسیم‌های سلولی

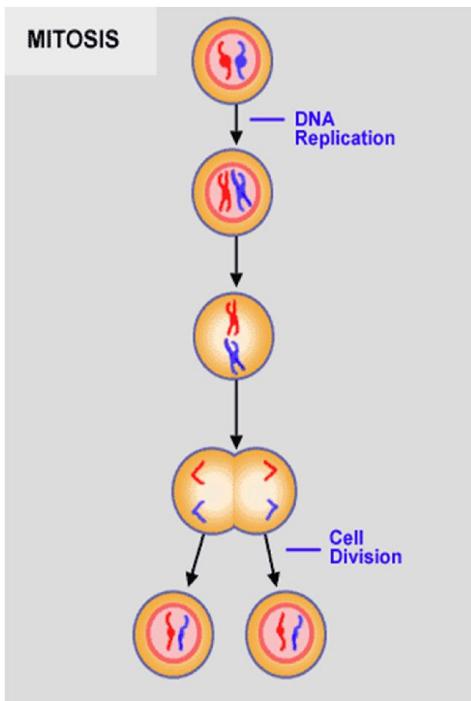
در بخش قبل تقسیم سلولی میتوz و میوز بیان گردید. میتوz شیوه متعارف تولید اندام‌ها در موجود زنده است. در میتوz همانند سازی DNA را داریم. بقیه اندام‌ها نیز به مقدار کافی تولید شده و سلول تبدیل به دو سلول می‌شود.

^۱ DNA-computer

^۲ cell-computer

^۳ membrane-computer

در تقسیم میتوز، در صورت تأثیرگذاری شرایط (مثل آلودگی هوا) امکان جهش و به تبع آن رخدادهایی چون سرطان وجود دارد. تقسیم میتوز در شکل زیر نمایش داده شده است:



تقسیم میوز مختص سلول‌های جنسی است. در تقسیم میوز، بر خلاف میتوز، ۴ سلول دختر خواهیم داشت. در میتوز، بر خلاف همانند سازی معمولی، DNA های سلول‌های تولید شده کاملاً مشابه سلول والد نیستند. در میوز دو مرحله تقسیم سلولی داریم:

- مشابه تقسیم میتوز، سلول به دو سلول با DNA های دیپلوبیدی تبدیل می‌شود.
- در مرحله بعد، تقسیمی انجام می‌شود که دو سلول هاپلوبیدی می‌سازد.

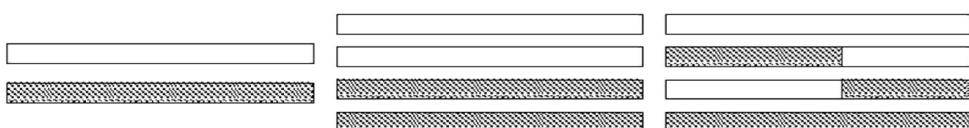
تنها سلول‌های موجودات زنده دیپلوبیدی که هاپلوبیدی هستند، تخمک و اسپرم‌مند؛ چرا که قرار است با ترکیب یکی از هر کدام آن‌ها، یک سلول دیپلوبیدی جدید تولید شود. به سلول به دست آمده از تقسیم میوز، سلول جنینی گفته می‌شود. در نر، هر ۴ اسپرم تولید شده قابل استفاده خواهد بود، در حالی که در ماده تنها یکی از ۴ تخمک تولید شده می‌تواند مورد استفاده قرار گیرد. با این حال، این که کدام یک از آن ۴ سلول تولید شده باقی می‌ماند کاملاً تصادفی خواهد بود.

در فرآیند تولید مثل با دو رخداد مواجه هستیم که سبب تغییرات ژنتیکی می‌شوند: بازترکیبی^۱ و جهش^۲. به تعبیری می‌توان این دو را خطاهایی دانست که در فرآیند تولید مثل رخ می‌دهند.

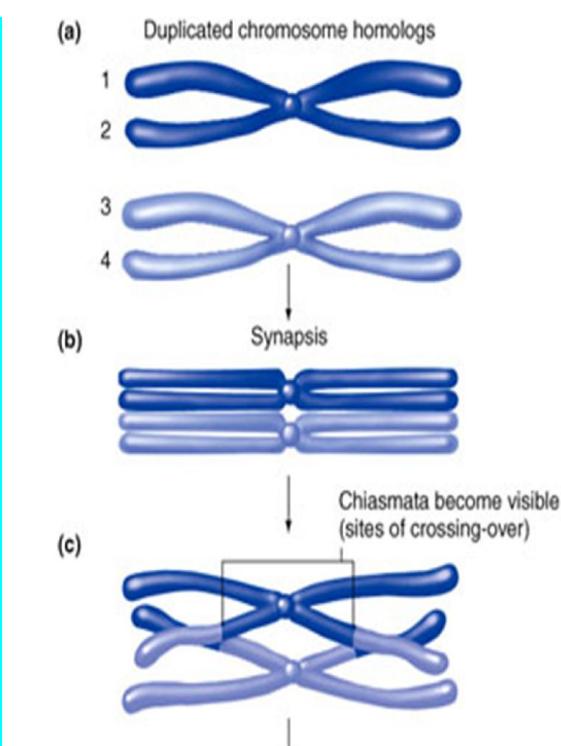
¹ Cross-over
² Mutation

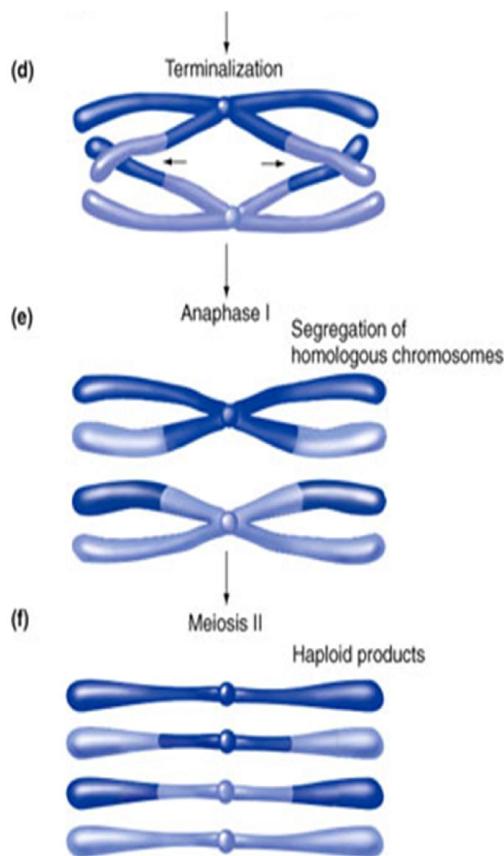
۱-۱-۱-۱ بازترکیبی

در تقسیم میتوز، دو DNA کاملاً همانند سازی می‌شوند و دو سلول کاملاً مشابه تولید می‌شود. به تعبیر دیگر، در این حالت بازترکیبی نخواهیم داشت. علی‌رغم این وضعیت، در تقسیم میوز بازترکیبی رخ می‌دهد. کروموزوم‌های 22 و 22' را در نظر بگیرید. یک فرضیه برای چگونگی رخداد بازترکیبی این است و وقتی دو کروموزوم در هم تنیده شده‌اند، در فرآیند باز و کپی شدن، کروموزوم با رسیدن به نقطه اتصال به کروموزوم دیگر، به جای ادامه مسیر روی خود، روی کروموزوم دیگر مسیر را ادامه می‌دهد. به این ترتیب نهایتاً ۴ کروموزوم به دست خواهد آمد که دو تا از آن‌ها 22 و 22' اولیه خواهند بود و دو تای دیگر، هر یک بخشی از کروموزوم 22 و بخشی از کروموزوم 22' را در خود دارند. این فرآیند را به فرم ساده در شکل زیر ملاحظه می‌کنید:



شکل‌های بعدی نیز به فرمی کامل‌تر بیان کننده بازترکیبی در تقسیم میوز هستند. به این ترتیب برای ۲۳ چفت کروموزوم،²³ ۴ حالت خواهیم داشت. همین اتفاق برای جنس مخالف نیز می‌افتد. به دلیل دیپلوئیدی بودن موجودات می‌توانیم چنین تنوعی را داشته باشیم.





آن چه گفته شد برای موجودات دیپلوبیوئیدی است. که در آن‌ها به ازای هر جفت کروموزوم هر یک از والدین نر و ماده، ۴ کروموزوم ایجاد می‌شود و لقاح می‌تواند مابین یک کروموزوم از مجموعه ۴ تایی والد نر و یک کروموزوم از مجموعه ۴ تایی والد نر رخ دهد. علاوه بر تنوع، سیستم‌های دیپلوبیوئیدی حافظه دار هستند.

در موجودات هاپلوبیوئیدی، برای تولید هر کروموزوم فرزند، یک کروموزوم والد نر و یک کروموزوم والد ماده ترکیب می‌شوند و در کروموزوم حاصل، بخشی متعلق به کروموزوم والد نر و بخشی متعلق به کروموزوم والد ماده است.

۱-۱۵-۲- جهش و تولید ویژگی جدید

اگر جا به جایی، به جای این که در پایان کد اتفاق بیفتند، پیش از پایان آن رخ دهد، می‌تواند منجر به تولید ویژگی جدیدی شود که پیش‌تر وجود نداشته است. باید دقت داشت که اگر بازترکیبی بین ژن رخ دهد، آلل جدیدی ایجاد نمی‌شود اما اگر در میان آلل جایه جایی داشته باشیم، می‌توانیم ویژگی جدید داشته باشیم.

همواره در حین تقسیم سلولی، چه میتوز و چه میوز، جهش (اختلاف‌های کوچکی) رخ می‌دهد. پیش‌تر اشاره شد که در همانند سازی در دو مرحله با این امر مقابله می‌شود. به علاوه، برای تغییر ماهیت فیزیکی پروتئین، تغییر چند آمینواسید لازم است و تغییر یک آمینواسید، چندان مشکل ساز نخواهد بود.

در پایان بحث، توجه به چند نکته ضروری است:

احتمال این که جهش مخرب باشد تقریباً صد درصد است و طبیعت جلوی آن را می‌گیرد.

در بازترکیبی جایگاه ژن‌ها عوض نمی‌شود، تنها مقدار آن‌هاست که تغییر می‌کند. یعنی کد رنگ چشم با کد دیگری از رنگ چشم عوض می‌شود و نه با کد مربوط به قد.

این سیستم هیچ وقت کروموزوم جدیدی تولید نمی‌کند. به بیان دیگر تکامل بین گونه‌ای ندارد و تنها تکامل درون گونه‌ای را سبب می‌شود.

۱-۱۵-۳- تولید سلول‌های بنیادی جنینی

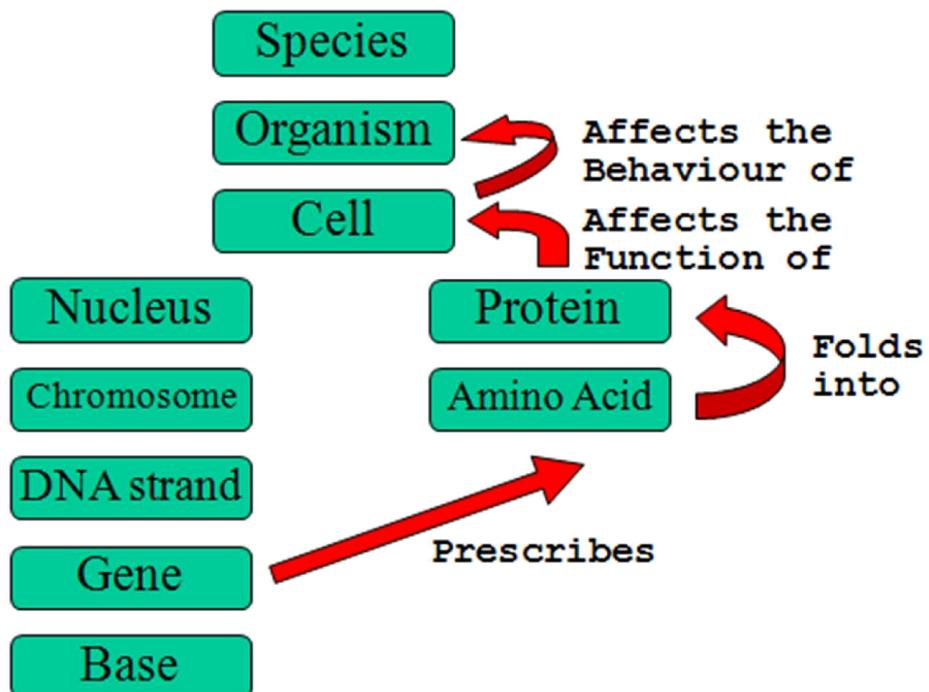
اگر مجدداً بحث خود را روی یک جفت کروموزوم ۲۲ و ۲۲' متمرکز کنیم، اگر به جای گرفتن اسپرم و تخمک از دو موجود، از ۴ حالت به وجود آمده برای سلول جنسی یک موجود واحد، کروموزوم‌های ۲۲ و ۲۲' را برداشته و با هم لقاح دهیم، موجود حاصل با موجود اولیه یکسان خواهد بود. دقت داشته باشید که برای انسان ۲۳ جفت کروموزوم وجود دارد و تا به اینجا در مورد یک جفت آن صحبت می‌کردیم. لازم است از ۴^{۲۳} حالت موجود، خالص‌ها را برداشته و با هم ترکیب کنیم.

این همانند سازی برای حیوانات انجام می‌گیرد و همراه با دستکاری‌های ژنتیکی است که به عنوان مثال در شیر بزرگوتی انسولین وجود داشته باشد. با این حال، زیگوت انسانی تولید شده به این صورت، تنها چند ساعت زنده می‌ماند.

برای زنده ماندن زیگوت شرایطی لازم است. اگر اسپرم و تخمک از دو گونه مختلف با یکدیگر ترکیب شوند یا لقاح صورت نمی‌گیرد و یا زیگوت تولید شده زنده نمی‌ماند. در واقع در موجود باید ساختار ژنتیکی و جایگاه‌های ژن مشابه داشته باشند. و یکی از دلایلی که سرعت تکامل خیلی پایین است همین امر است، چرا که دو موجود برای داشتن قابلیت لقاح، باید خیلی نزدیک باشند.

دیدیم که بخشی از ویژگی‌های فیزیکی فرزند از پدر و بخشی از آن‌ها از مادر می‌آیند. به علاوه، ویژگی‌هایی که از هر یک از والدین مثلاً پدر به فرزند می‌رسد نیز الزاماً خالص نیست، بلکه می‌تواند ترکیبی از دو کروموزوم او باشد. با این حال، فرزند نمی‌تواند ویژگی‌ای خارج از مجموعه ویژگی‌های والدین خود داشته باشد. بر همین اساس است که با آزمایش DNA می‌توان تعیین کرد فردی فرزند کسی هست یا نه.

برای جمع‌بندی، شکل زیر فرآیند تولید ژن از باز و بعد ترکیب ژن‌ها و ساخت DNA و پس از آن تولید کروموزوم از DNA را نشان می‌دهد. کروموزوم‌ها در هسته سلول قرار دارند و هر عضو موجود زنده شامل تعدادی سلول است. به علاوه، مجموعه اعضا یک موجود زنده را که متعلق به گونه‌ای خاص است ایجاد می‌کند. همچنین در سوی دیگر این نمودار می‌بینیم که بر اساس ژن‌ها آمینواسیدها تولید شده و با استفاده از آن‌ها پروتئین‌ها ساخته می‌شوند و پروتئین‌های سلول را خواهند ساخت. به این ترتیب در واقع یک چرخه داریم.



فصل چهارم: تئوری تکامل از دیدگاه ماکروسکوپی

۱۶- مقدمه

در فصل قبل تئوری تکامل از منظر میکروسکوپی مورد بررسی قرار گرفت. در این فصل از دیدگاه ماکروسکوپی به تکامل خواهیم پرداخت.

در تکامل، هر نسل نسبت به نسل قبل خود، به سمت بهبود پیش می‌رود به گونه‌ای که فرزندان ویژگی‌های شایسته‌تری نسبت به والدین داشته باشند و این شایستگی در توان تطابق بالاتر با محیط معنا می‌شود. هدف ما این است که از ایده تکامل برای حل مسائل بھینه سازی استفاده کنیم. لذا اگر جمعیتی داشته باشیم و آن را در هر نسل تکامل دهیم، بر این اساس می‌توانیم الگوریتمی برای حل مساله بھینه سازی داشته باشیم.

در طبیعت جمعیتی از کروموزوم‌ها وجود دارند. ما می‌بایست بین کروموزوم‌های طبیعی و کروموزوم‌های مسأله خود تناظر برقرار کنیم. به این ترتیب، پارامترهایی که مقدار تابع را تغییر می‌دهند معادل ژن‌ها در طبیعت خواهند بود. باید دقیق داشت که تنها پارامترهایی که تغییرشان سبب تغییر در تابع هدف می‌شود مد نظر هستند و نه هر پارامتری.

x_1	x_2	...	x_n
-------	-------	-----	-------

شكل بالا یک کروموزوم را نمایش می‌دهد. که x_1 تا x_n ژن‌های آن هستند. هدف بھینه کردن $f(x_1, \dots, x_n)$ است. به علاوه، کدینگی داریم که ارتباط کروموزوم و مسأله را روشن می‌کند.

۱۷- ساختار کلی الگوریتم‌های تکاملی

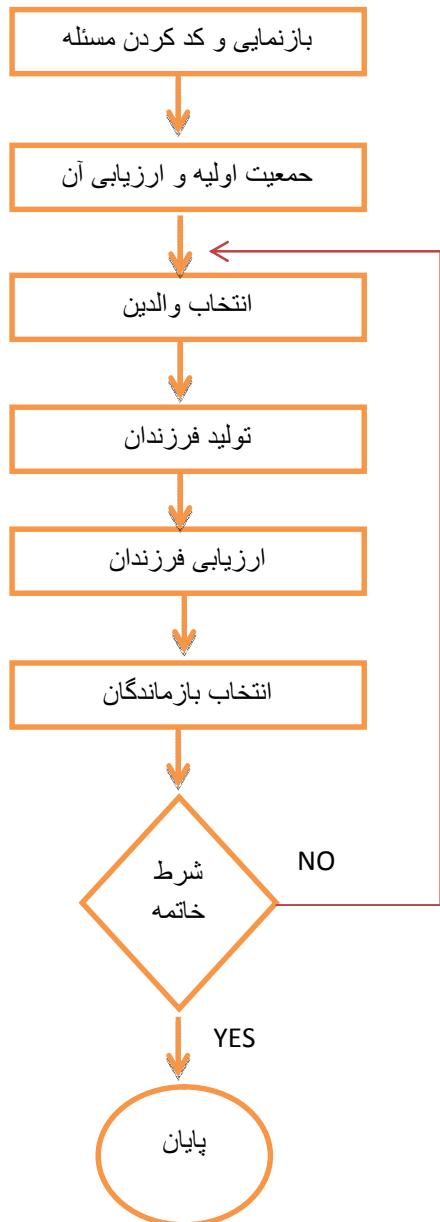
در فصل قبل دیدیم که تکامل نیازمند دو فاکتور تنوع و انتخاب است. تنوع از طریق تولید مثل ایجاد می‌شود که شامل بازترکیبی و جهش است. به علاوه، در طبیعت همه موجودات از یک نوع نمی‌توانند تولید مثل کنند، بنابراین نیاز به انتخاب والدین است.

یک الگوریتم تکاملی دارای مراحل اصلی زیر است:

- ۱- کد کردن مسأله (بازنمایی)
- ۲- جمعیت اولیه و ارزیابی آن
- ۳- انتخاب والدین
- ۴- تولید مثل (به دو روش بازترکیبی و جهش)
- ۵- ارزیابی فرزندان (از نظر شایستگی)
- ۶- انتخاب بازماندگان

-۷ شرط خاتمه، در غیر این صورت برگشت به ۳

فرم بلاک دیاگرامی الگوریتم به صورت زیر است:



شناسنامه الگوریتم‌های تکاملی

همان گونه که پیش‌تر اشاره شد، الگوریتم‌های تکاملی شناسنامه‌ای به شکل زیر با ۹ آیتم دارند.

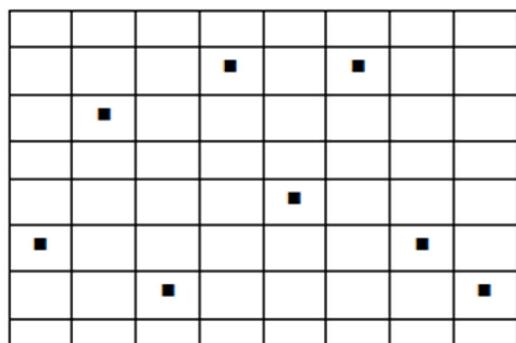
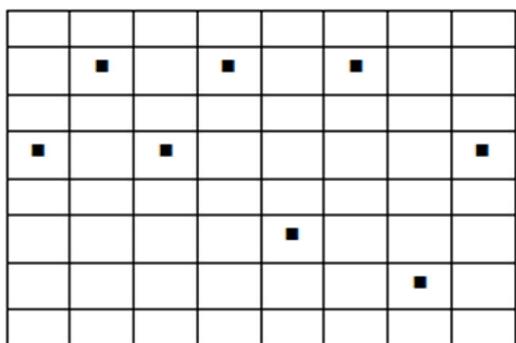
نامگذاری، بر اساس این که در هر یک از این ۹ مرحله چه کاری انجام می‌شود صورت می‌گیرد.

بازنمایی
جمعیت اولیه
ارزیابی
انتخاب والدین
بازترکیبی
جهش
انتخاب بازماندگان
شرط خاتمه
ویژگی خاص

ایده کلی در این روش‌ها این است که جمعیتی از کروموزوم‌ها را تکامل داده و در هر نسل بهبود بخشیم تا به پاسخ برسیم. اما منطق این که اساساً چرا امیدواریم این شیوه منجر به پاسخ شود، تکامل در طبیعت است.

مثال: فرض کنید هدف حل مسأله ۸ وزیر است:

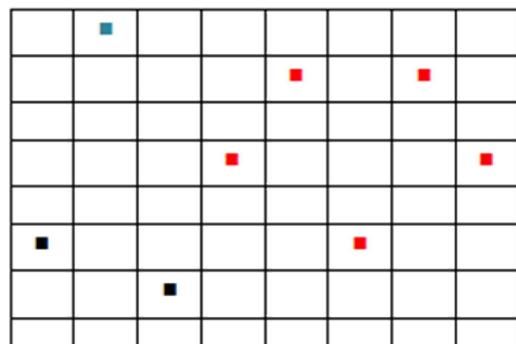
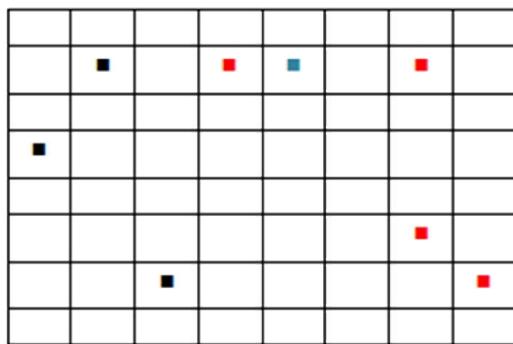
دو موجود به صورت تصادفی تولید می‌کنیم:



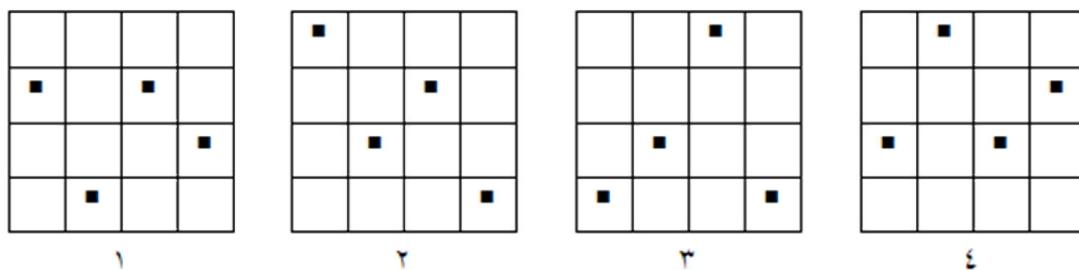
فرض کنید هر ستون بیان گر یک ژن است و می‌تواند تنها یک ۱ (وزیر) داشته باشد. که نقطه‌های سیاه به معنای وجود وزیر در خانه مذکور هستند.

نحوه ارزیابی در این مسأله بر اساس تعداد گارد شدن هر وزیر است (تعداد وزیرهایی که در سطر، ستون و مسیرهای

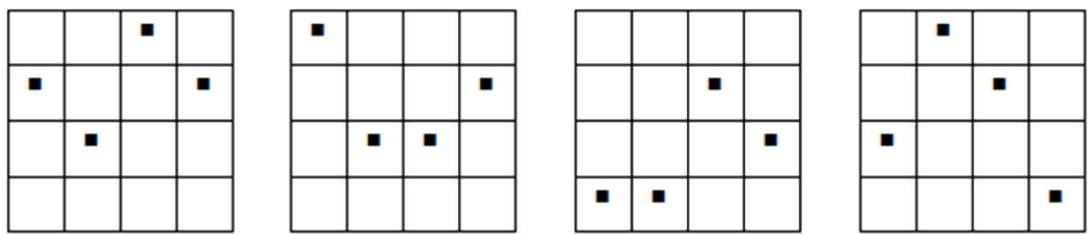
قطري مشترك با آن وجود دارند). مجموع اين تعداد (جريمه‌ها) يك عدد به هر موجود نسبت مي‌دهد.
اگر ترکيب را بعد از ستون سوم سمت راست در نظر بگيريم، مي‌توانيم دو فرزند توليد کنيم. همچنان مي‌توانيم
جهش (به معنای عوض شدن جای وزیر در هر ستون) را نيز با احتمال $1/6$ در نظر گرفته و اعمال نماييم. حاصل به
صورت زير خواهد بود (جهش‌ها با رنگ آبي مشخص شده‌اند):
پس از چك کردن شايستگي، اين دو برای انتقال به نسل بعد مورد بررسی قرار مي‌گيرند.



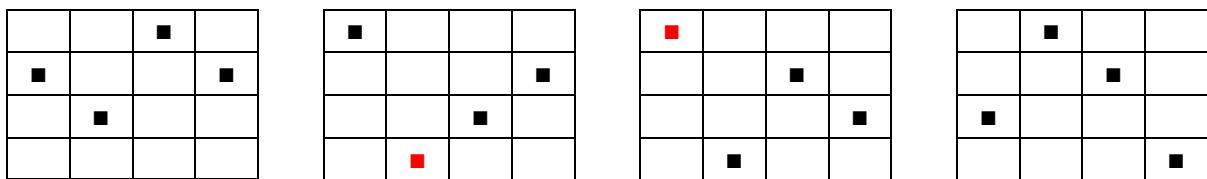
در ادامه الگوريتم گفته شده در بالا را روی مسئله ۴ وزير اعمال و بررسی خواهيم کرد. فرض کنيد تعداد جمعيت، M ،
برابر ۴ و تعداد فرزندان، λ نيز برابر ۴ باشد. در اين مسئله به ازاي هر گارد شدن يك جريمه در نظر مي‌گيريم و
هدف کمينه کردن اين جريمه است. بدويهي است که با صفر شدن به جواب رسيده‌اييم. جمعيت اوليه را به صورت زير
در نظر بگيريد:



برای هر ۴ موجود جمعيت اوليه، جريمه برابر ۴ خواهد بود.
همينجا باید توجه داشت کهتابع شايستگي تعریف شده خوب، تابعی است که قادر باشد مابین حالت‌های مختلف
تمایز قائل شود. به تعبیر ديگر در اينجا تابع نمی‌تواند بین موجودات مختلف تمایز قائل شود، پس خوب نیست.
در گام بعد انتخاب والدين را داريم. در انتخاب والدين دو موضوع مطرح است؛ اين که کدام موجودات توليد مثل کنند
و اين که کدام موجود با کدام موجود توليد مثل را انجام دهد. مثلاً فرض کنيد موجودات ۱ و ۳ با هم، و هم ۲ و ۴ با
هم فرزند توليد کنند. در اينجا ترکيب را از يك نقطه انجام مي‌دهيم.



فرض کنید که دو موجود میانی دستخوش جهش شوند و مجموعه نهایی فرزندان به صورت زیر تولید شود:



تابع شایستگی فرزندان به ترتیب از راست به چپ برابر است با ۶، ۲، ۸ و ۲. دقت داشته باشید که فقط جمعیت اولیه و فرزندان را خواهیم دید و نتایج مراحل میانی حائز اهمیت نیستند. اگر جمعیت اولیه را شامل ۶ موجود و مراحل را به صورت زیر در نظر بگیریم خواهیم داشت:

جمعیت اولیه:

1	2	3	4	5	6
---	---	---	---	---	---

والدین:

1	3	2	1	3	4
---	---	---	---	---	---

بازترکیبی:

1'	3''	2	1	3'	4'
----	-----	---	---	----	----

فرزندان:

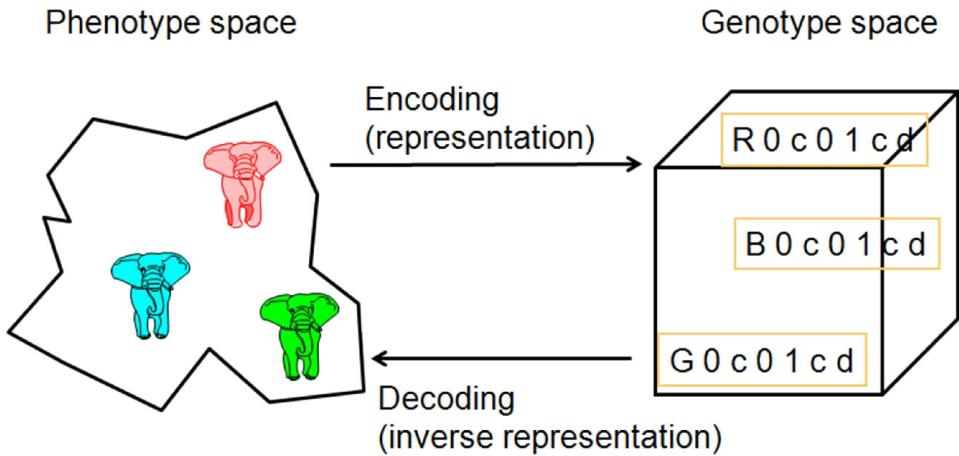
1'	3'''	2'	1	3''	4'
----	------	----	---	-----	----

جهش‌ها با رنگ قرمز مشخص شده‌اند. همان‌گونه که اشاره شد دو مرحله والدین و بازترکیبی اهمیتی ندارند. در ادامه هر یک از بخش‌های الگوریتم را به تفصیل مورد بررسی قرار خواهیم داد:

بازنمایی^۱ (کد کردن)

پیش‌تر در بحث ژنوتیپ و فونوتیپ، با مفهوم کد کردن آشنا شدیم.

¹ Representation



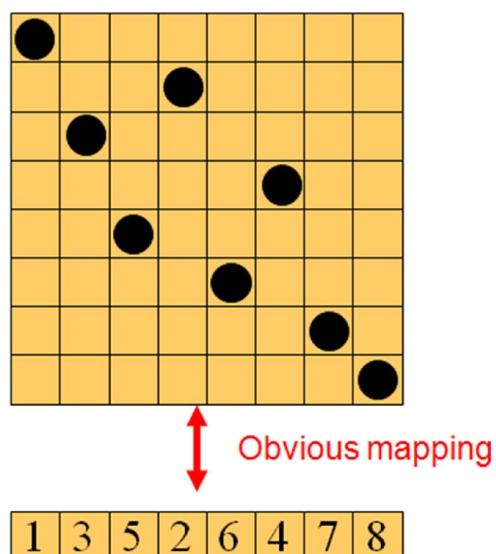
در بازنمایی، هدف کد کردن اطلاعات است. بازنمایی مطلوب باید دارای چهار ویژگی باشد:

- تمام راه حل‌های ممکن را کد کند.

بدیهی است که اگر بازنمایی ما قادر به کد کردن برخی از راه حل‌ها نباشد، توان یافتن پاسخ از میان آن‌ها را نیز خواهد داشت.

- حتی المقدور فضای جستجو را کوچک کند.

به عنوان مثال در مساله ۸ وزیر، ۸ ژن را در نظر می‌گیریم که هر یک می‌تواند مقادیر ۱ تا ۶۴ را بگیرد. به این ترتیب فضای حالت 64^8 است. اما اگر ۸ ژن داشته باشیم و هر یک معادل یک ستون باشد و عدد داخل ژن معادل یک سطر باشد، با توجه به این که در هر سطر و ستون نمی‌توان بیش از یک وزیر داشت، مساله تبدیل به یک مساله جایگشت با فضای حالت 8 خواهد شد که به مرتب کوچک‌تر است.



- بازنمایی باید به گونه‌ای باشد که تابع شایستگی برای حالت‌های مختلف متفاوت بوده و قادر به تمایز آن‌ها باشد.
- باید امکان بازترکیبی و جهش مناسب را فراهم آورد.

دقت داشته باشید که حذف redundancy ها می‌تواند منجر به این شود که نتوانیم بازترکیبی و جهش‌های خوبی تعریف کنیم. به عنوان مثال در حالت زیر، نمی‌توانیم بازترکیبی یک نقطه‌ای انجام دهیم.

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

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

با فرض این که ترکیب بعد از ستون سوم از سمت چپ انجام شود، نتایج به شرح زیر خواهند بود که قابل پذیرش نیستند:

7	2	8	8	7	1	3	6
---	---	---	---	---	---	---	---

5	2	4	4	1	5	3	6
---	---	---	---	---	---	---	---

۱-۲-۱۷- روش‌های بازنمایی (کد کردن)

- باینری

در مثال استخراج و اکتشاف معادن، جستجوی عمومی و محلی را توضیح دادیم. در الگوریتم‌های تکاملی هدف این است که هر دو جستجو را به طور همزمان داشته باشیم. معمولاً^۱ یکی از عملگرهای جهش و بازترکیبی را جستجوی محلی در نظر می‌گیرند و دیگری را جستجوی عمومی.

در بازنمایی باینری، تغییر بیت کم ارزش معادل جستجوی محلی است و تغییر بیت پر ارزش معادل جستجوی عمومی است. بدیهی است که هم جهش و هم بازترکیبی می‌توانند هر دو مجموعه بیت‌های با ارزش کم و زیاد را تغییر دهند. لذا هیچ یک را نمی‌توان معادل جستجوی محلی یا عمومی دانست. و این یکی از عیوب‌های بازنمایی باینری است. برای اصلاح آن از gray code استفاده می‌شود.

سایر روش‌های بازنمایی عبارتند از:

- کد گری^۱
- جایگشت
- اعداد صحیح
- اعداد حقیقی
- درخت

¹ Gray code

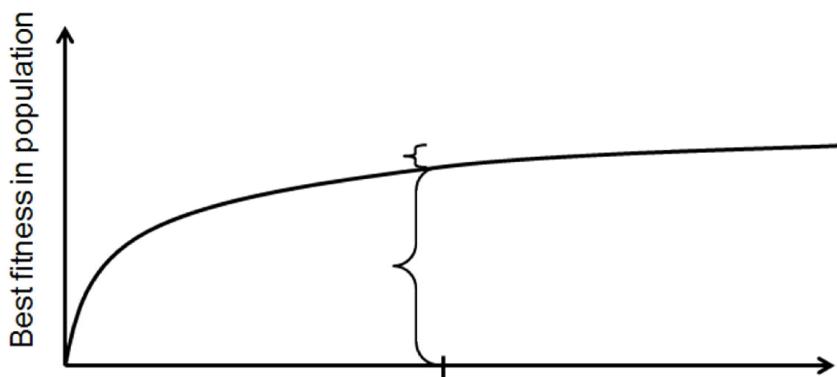
- گراف

- گروه

جمعیت اولیه

جمعیت اولیه به صورت تصادفی و یکنواخت تولید می‌شود.

اگر جمعیت اولیه توسط یک هیورستیک تولید شود و دارای شایستگی بالایی باشد، در صورتی که بدانیم که پاسخ بهینه را می‌دهد اساساً نیازی به الگوریتم تکاملی نیست. اما اگر فقط شایستگی را بالا ببرد، این ریسک وجود دارد که ما را به سمت کمینه محلی ببرد. به این ترتیب الگوریتم می‌باشد ابتدا از کمینه محلی مذکور خارج شده و سپس کار خود را انجام دهد. به علاوه منحنی‌های الگوریتم‌های تکاملی به صورت زیر هستند:



به تعبیر دیگر، این الگوریتم‌ها در نسل‌های اولیه خیلی سریع پیش می‌روند. گفته می‌شود، ۲۵٪ زمان را صرف رسیدن به ۹۰٪ پاسخ می‌کنند و ۷۵٪ زمان را برای ۱۰٪ باقی مانده. به این ترتیب، حتی بدون داشتن الگوریتم هیورستیک، خود الگوریتم تکاملی در زمان کوتاهی به شایستگی‌ای آن خواهد رسید. و گذشته از آن ریسک گیر کردن در می‌نرم محلی از بین خواهد رفت.

با این حال مسائلی وجود دارند که در آن‌ها از جمعیت‌های خاصی استفاده می‌شود. مثلاً در مساله دنبال کردن^۱ هوایپما گرچه برای یافتن هوایپما در صفحه لازم است جمعیت تصادفی یکنواخت داشته باشیم، اما پس از آن با علم به این که هوایپما کجا قرار دارد، در لحظه بعدی تنها می‌تواند در مجموعه نقاطی در همان حوالی باشد. لذا نیازی به پوشش مجدد کل فضا نخواهد بود.

¹ Tracking

انتخاب و انواع آن

موضوع بعدی که به آن خواهیم پرداخت انتخاب است. اگر انتخاب متناسب با شایستگی نباشد اساساً الگوریتم تکاملی خواهیم داشت. دیدیم که در دو مرحله درگیر انتخاب می‌شویم. لزومی ندارد که هر دوی انتخاب‌های مذکور بر اساس شایستگی باشند. اما دست کم یکی از آن‌ها باید با شایستگی در ارتباط باشد.

انتخاب می‌تواند به یکی از طرق زیر باشد:

۱- انتخاب تصادفی یکنواخت

۲- بدون انتخاب

هر یک از دو روش فوق برای انجام انتخاب، می‌تواند در یکی از دو مرحله‌ای که انتخاب مطرح است به کار گرفته شود و نه در هر دوی آن‌ها.

۳- انتخاب بُرشی (شاپرکه سالاری)

در این نوع انتخاب، موجودات بر اساس شایستگی مرتب می‌شوند و از شایستگی بیشتر شروع کرده و به تعداد لازم از آن‌ها بر می‌داریم.

۴- انتخاب متناسب با شایستگی

این شیوه انتخاب می‌تواند به سه صورت SUS، RW و روشی مشابه با SUS انجام گیرد که در ادامه به آن‌ها خواهیم پرداخت.

برای هر موجود یک احتمال تعریف می‌کنیم:

$$p_i = \frac{f_i}{\sum_{j=1}^{\text{تعداد موجودات}} f_j}$$

همچنین باید داشته باشیم:

$$\sum p_i = (\sum f_i) / (\sum f_j) = 1$$

این جا دو تعداد مطرح است:

- موجوداتی که داریم

- موجوداتی که قصد داریم از بین شان انتخاب کنیم.

معمولًاً تعداد فرزندان و والدین برابر هم و برابر λ است.

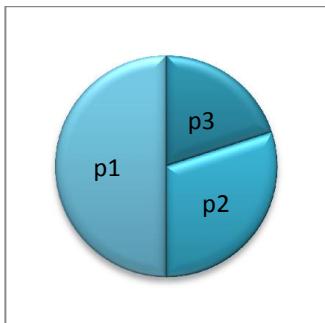
$$p_i = \frac{1}{\sum_j \left(\frac{1}{c_j} \right)}$$

در واقع آن چه مطلوب است این است که f را حداکثر و c را حداقل کنیم.

۱-۴-۱۷- روش رولت ویل^۱ (RW)

ایده این روش از اماکن بازی آمده است. در این مکان‌ها، افراد ژتون‌هایشان را روی صفحه‌ای دایره‌ای شکل قرار می‌دهند و اگر شاخص روی ژتون مورد نظر بایستد فرد برنده می‌شود. بدیهی است که تعداد ژتون‌های بیشتر، به معنای داشتن شанс بالاتر خواهد بود.

به عنوان مثال فرض کنید سه موجود با شایستگی‌های ۲۰، ۳۰ و ۵۰ داریم:



به این ترتیب:

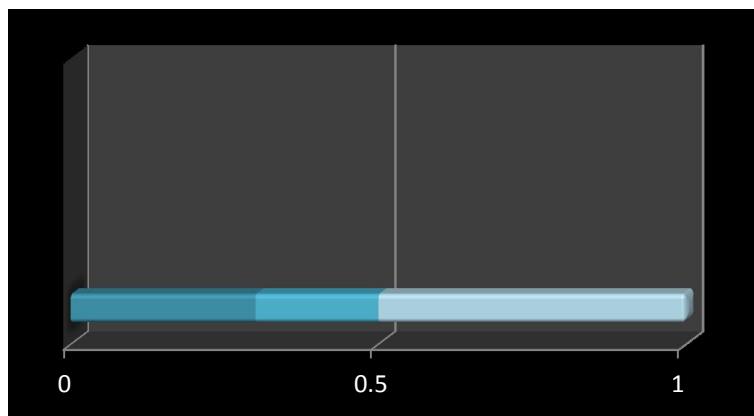
$$p_1 = \frac{50}{50 + 30 + 20} = 0.5$$

$$p_2 = \frac{30}{50 + 30 + 20} = 0.3$$

$$p_3 = \frac{20}{50 + 30 + 20} = 0.2$$

در این حالت به تصادف عددی بین ۰ تا ۳۶۰ انتخاب می‌شود و در هر ناحیه که قرار گیرد احتمال مربوط به آن را خواهیم داشت.

در روش‌های کامپیوتری معادل این دایره یک خط ۰ تا ۱ را در نظر می‌گیریم.

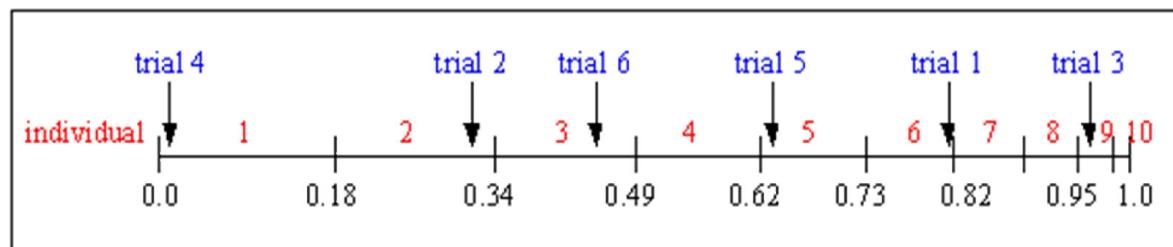


به این ترتیب شанс انتخاب موجودی که شایستگی بیشتر دارد، بیشتر است.

^۱ Rolet Wheel

مثال دیگری از روش RW: فرض کنید ۱۱ فرد مطابق جدول زیر داریم که میزان شایستگی آنها و نیز احتمال انتخابشان مشخص شده است. اگر مانند مثال قبل کل بازه را فضای $0 \dots 1$ در نظر بگیریم، شکل احتمال مربوط به انتخاب هر یک از افراد را نمایش می‌دهد.

شماره فرد	۱۱	۱۰	۹	۸	۷	۶	۵	۴	۳	۲	۱	میزان شایستگی
احتمال انتخاب شدن	۰.۰	۰.۰۲	۰.۰۳	۰.۰۶	۰.۰۷	۰.۰۹	۰.۱۱	۰.۱۳	۰.۱۵	۰.۱۶	۰.۱۸	



آیا این روش، انتخاب بر اساس شایستگی را محقق می‌کند؟ پاسخ به این سؤال در همه حال مثبت نیست. همان طور که در شکل بالا نیز می‌بینید، ممکن است در هر آزمایش، به رغم تفاوت‌هایی که در احتمال‌ها وجود دارد، پاسخ در هر یک از بازه‌های مشخص شده فوق قرار گیرد. برای روشن‌تر شدن موضوع، پرتاپ تاس را در نظر بگیرید. در پرتاپ تاس به تعداد کم نمی‌توان درباره این که چه عددی می‌آید اظهار نظر کرد. در مقابل وقتی تعداد پرتاپ‌ها زیاد باشد با دقت بالایی می‌توان گفت که احتمال آمدن هر عدد، یک ششم خواهد بود. در اینجا نیز با وضعیتی مشابه روبرو هستیم. وقتی تعداد موجودات کم باشد، یکنواختی از بین می‌رود.

۱-۱۷-۴-۲ روش نمونه برداری عمومی تصادفی^۱ (SUS)

اگر بخواهیم در تعداد کم هم یکنواختی برقرار باشد چه باید کرد؟ روش SUS و RW در اساس کاملاً مشابه هستند با این تفاوت که SUS ضعف عدم یکنواختی RW را در تعداد اندک پوشش می‌دهد.

در این روش، احتمال نسبت داده شده به هر فرد برابر است با:

$$P_i = \frac{e^{\frac{f_i}{T}}}{\sum e^{\frac{f_i}{T}}}$$

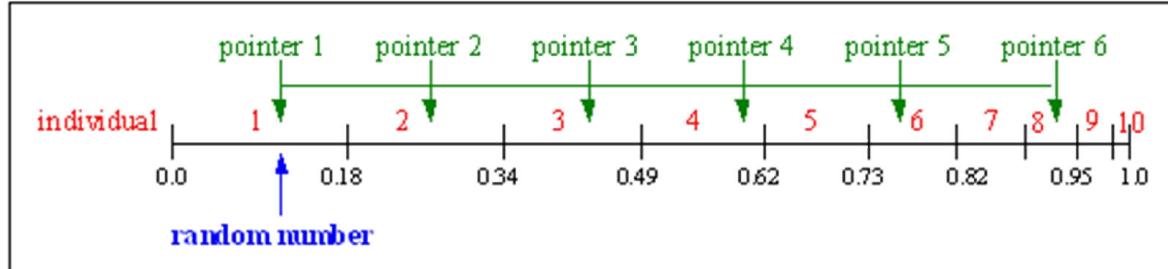
که T مقدار ثابتی است که به عنوان پارامتر کنترلی این روش در نظر گرفته می‌شود.

در SUS، احتمال‌های نسبت داده شده به افراد مانند RW است. برای تولید N عدد در بازه $[0, 1]$ عددی تصادفی در بازه $[0, \frac{1}{N}]$ تولید شده و اعداد بعدی هر یک به فاصله $1/N$ از عدد قبلی انتخاب می‌شوند. به عبارت دیگر، اگر تعداد موجودات را N در نظر بگیریم، در اینجا خط کشی به طول $1/N - 1$ وجود دارد. روی این خط کش به

¹ Stochastic Universal Sampling

فواصل $1/N$ ، $1/N$ جدا می‌کنیم. به این ترتیب تعداد N فلش روی آن خواهیم داشت. یک عدد تصادفی یکنواخت بین ۰ و $1/N$ تولید می‌کنیم. این عدد شروع خط کش دوم را روی خط کش اول مشخص می‌کند.

به عنوان مثال اگر $N = 6$ در نظر بگیریم، $\frac{1}{N} = 1/6 \approx 0.167$ خواهد بود. شکل زیر چگونگی انتخاب به روش SUS را برای این حالت و با مقادیر گفته شده در مثال قبل، نمایش می‌دهد:



دقت داشته باشید که در SUS، p_i ها به صورت تصادفی چیده می‌شوند. اگر نخواهیم این گونه باشد، خط کش را به تعداد موجوداتی که می‌خواهیم داشته باشیم تقسیم می‌کنیم؛ مثلاً ۳. و این کار را با در نظر گرفتن قطعات مساوی انجام می‌دهیم. یک عدد تصادفی یکنواخت انتخاب می‌کنیم. در هر ناحیه قرار گرفت، موجود مربوط به آن ناحیه را انتخاب می‌کنیم.

امید ریاضی انتخاب موجود به صورت زیر خواهد بود:

$$E[n_i] = N \cdot P_i$$

که N تعداد موجوداتی است که انتخاب می‌کنیم و p_i احتمال مربوط به آن موجود است.

مثال: اگر $N = 3$ و $p_i = 0.5$ باشد $E[n_i] = 1.5$ خواهد بود. یعنی بین ۱ تا ۲ موجود انتخاب می‌شود.

در SUS پراکندگی روی تمام خطکش یکنواخت است و به این ترتیب مشکل RW را در تعداد کم حل می‌کند چرا که در تعداد کم، در RW ممکن است همه در یک ناحیه متمرکز شده و یکنواختی مورد نظر حاصل نشود. باید توجه داشت که روش‌های RW و روش سوم از نظر اساس ریاضی هیچ تفاوتی ندارند و روابط گفته شده در بالا برای هر سه برقرار است. به علاوه، پاسخ روش‌های دوم و سوم هم یکسان هستند. با این حال پیاده سازی روش سوم ساده‌تر است.

برای داشتن ایده‌ای از تعداد نسل‌ها، در الگوریتم‌های تکاملی، این تعداد بین ۱۰۰۰ تا ۱۰۰۰۰ است و تعداد جمعیت حداقل ۱۰۰ است. همچنین تعداد ارزیابی‌ها به طور معمول ۱۰۰ تا ۲۰۰ هزار است.

۱-۴-۳-۲-۱۷-۱ تعریف چند اصطلاح

۱-۴-۳-۲-۱۷-۱ فشار انتخاب^۱ (SP)

برابر است با احتمال انتخاب بهترین فرد جامعه در مقایسه با میانگین احتمال انتخاب کلیه موجودات. به تعبیری بیان گر تعداد کپی‌هایی است که از موجود شایسته انتخاب می‌شود (فرقی ندارد برای والدین یا بازماندگان). اگر p_b امید ریاضی تعداد کپی‌هایی که از هر موجود انتخاب می‌شوند، باشند SP به صورت زیر تعریف می‌شود:

$$SP = Np_b$$

که N می‌تواند در انتخاب والدین λ و در انتخاب بازماندگان μ باشد.

۱-۴-۳-۲-۱۷-۱ شیوع

شیوع^۲ امید ریاضی تعداد کپی‌هایی است که از هر موجود انتخاب می‌شود:

$$spread = Np_i = E[n_i]$$

در حقیقت فشار انتخاب، شیوع بهترین موجود است.

۱-۴-۳-۲-۱۷-۱ بایاس

بایاس^۳ اختلاف احتمال انتخاب موجود و شایستگی نرمال شده است و معمولاً^۴ به صورت قدر مطلق بیان می‌گردد:

$$bias = p_i - \frac{f_i}{\sum_j f_j}$$

اگر انتخاب‌ها متناسب با شایستگی باشند، بایاس صفر است. هر چه بایاس از صفر دورتر باشد، از تناسب با شایستگی دورتر می‌شویم. اگر بایاس صفر باشد، انتخاب دقیقاً متناسب با شایستگی است و:

$$p_i = \frac{f_i}{\sum_j f_j}$$

۱-۴-۳-۲-۱۷-۱ شدت انتخاب^۵ (SI)

اگر جمعیتی داشته باشیم که شایستگی آن را با توزیع نرمال (متوسط صفر و واریانس ۱) و توزیع گوسی تولید کرده باشیم، و روی آن روش انتخاب را اعمال نمائیم و متوسط شایستگی موجودات انتخاب شده را به دست آوریم و چندین بار تکرار کرده و از آن‌ها متوسط بگیریم (یعنی امید ریاضی را محاسبه نماییم)، SI به دست می‌آید.

¹ Selection Pressure

² Spreed

³ Bais

⁴ Selection Intensity

SI بالا به معنای سرعت همگرایی بالاتر است. یعنی روش انتخاب مورد بررسی، شایستگی را به سرعت بالا می‌برد. بر عکس اگر SI پایین باشد به این معناست که روش انتخاب مورد بحث تنوع را حفظ می‌کند.

برای حفظ تنوع باید $SI = 0$ باقی بماند. اگر بخواهیم همگرا شود نیاز است که $0 > SI$ باشد. همچنین $0 < SI$ به معنای کم شدن شایستگی است.

۱-۴-۳-۵ واریانس انتخاب^۱ (SV)

فرض کنید جمعیتی مشابه آن چه در بالا گفته شد داریم. روش انتخاب را اعمال کرده و واریانس شایستگی‌ها را حساب می‌کنیم. این کار را چندین بار تکرار کرده و از نتایج متوسط می‌گیریم. به این ترتیب SV به دست خواهد آمد.

بالاتر به این معناست که روش انتخاب تنوع بالاتری دارد. دو معیار SV و SI بر عکس هم هستند و بیانگر این هستند که روش انتخاب دارای همگرایی است یا تنوع.

۱-۴-۳-۶ کاهش تنوع^۲ (LOD)

این معیار بیان‌گر احتمال متناظر با موجوداتی است که انتخاب نمی‌شوند. اگر تعداد کل موجودات N و تعداد موجودات انتخاب شده μ باشند می‌توانیم دو شیوه انتخاب در نظر بگیریم؛ انتخاب از میان بازماندگان و از میان موجودات انتخاب شده. در حالت انتخاب از بین بازماندگان:

$$LOD = (N - \mu)/N$$

و در حالت انتخاب از μ :

$$LOD = (\mu - \lambda)/\mu$$

توجه داشته باشید که دو معیار SV و LOD بیان‌گر یک واقعیت هستند. با این حال، هر چه LOD بیشتر باشد، SV کمتر است و بالعکس.

اکنون به بررسی معیارهای گفته شده در انتخاب متناسب با شایستگی می‌پردازیم:

$$sp = \frac{Nf_b}{\sum f_j}, bias = 0, spread = N\left(\frac{f_i}{\sum f_j}\right)$$

جمعیت همیشه ثابت و برابر μ نگه داشته می‌شود.

در رابطه گفته شده برای بازماندگان:

$$LOD = \begin{cases} \frac{N - \mu}{\lambda} & \mu, \lambda \text{ model} \\ \frac{\lambda}{\mu + \lambda} & \mu + \lambda \text{ model} \end{cases}$$

¹ Selection Variance

² Lost of Diversity

انتخاب متناسب با شایستگی دو عیب دارد. مورد اول در شرایطی است که در جمعیت موجودی داشته باشیم که شایستگی آن خیلی بیش از متوسط و در عین حال خیلی کمتر از شایستگی واقعی باشد یعنی:

$$f_{max} \gg f_b \gg \bar{f}$$

به این ترتیب اگر $p_i = \frac{f_i}{\sum f_j}$ ، یعنی انتخاب بر اساس شایستگی باشد و موجودی با شرایط فوق الذکر داشته باشیم، sp بالا رفته و کپی‌های موجود برتر بعد از یکی دو نسل تمام موجودات را فرا خواهند گرفت. در این صورت تنوع از بین رفته و الگوریتم متوقف می‌شود.

به عنوان مثال فرض کنید شایستگی موجودات به شرح زیر باشد:

$$90, 3, 2, 1, 0.5, 1.5, 0.75, 1.25, 0.25, 0.25$$

همچنین فرض کنید شایستگی واقعی ۲۰۰ باشد. در این صورت:

$$p_b = \frac{90}{100} = 0.9, p_2 = 0.03, p_3 = 0.02, \dots$$

اکنون اگر از روش سوم برای انتخاب استفاده کنیم، ۹ تا از موجود اول و یکی از سایر موجودات انتخاب خواهد شد و در نسل بعد همه موجودات، کپی موجود اول خواهند بود.

اشکال دوم در شرایطی پیش می‌آید که موجودات شایستگی مشابه داشته باشند. در این حالت، شیوع و sp برابر ۱ می‌شود. به عبارت دیگر، همان موجودات در نسل بعد تکرار خواهند شد. این جمعیت تنوع پایینی دارد. تکرار شدن جمعیت در نسل بعد را ساکن شدن جمعیت گویند.

فشار انتخاب مشکل همگرایی زودرس را ایجاد می‌کند و اگر بتوانیم آن را کنترل نماییم قادر خواهیم بود همگرایی زودرس را نیز کنترل کنیم.

به منظور جلوگیری از متوقف شدن تکامل، می‌بایست از ساکن شدن جمعیت و نیز همگرایی زودرس جلوگیری کنیم. در واقع نباید اجازه دهیم جمعیت به جایی برسد که چنین انفاقاتی رخ دهند.

اگر \bar{f} را متوسط شایستگی در نظر بگیریم؛ در همگرایی زودرس f_b فاصله زیادی با \bar{f} دارد و در حالت ساکن شدن جمعیت، بسیار نزدیک به آن است.

شكل

راه حل این است که با اعمال تبدیلی، در هر دو حالت، فاصله معقولی مابین f_b و \bar{f} ایجاد کنیم. با فرض استفاده از یک تبدیل خطی، \bar{f} را به \bar{f}' و f_b را به $\beta\bar{f}$ تبدیل می‌کنیم.

بر اساس فرم کلی معادله، محاسبه به صورت زیر است:

$$y = \frac{y_2 - y_1}{x_2 - x_1} (x - x_1) + y_1$$

$$\begin{cases} x = f \\ y = f' \end{cases} \quad \begin{cases} x_1 = \bar{f} \\ y_1 = \bar{f} \end{cases}$$

به این ترتیب خواهیم داشت:

$$f' = \frac{\beta\bar{f} - \bar{f}}{f_b - \bar{f}} (f - \bar{f}) + \bar{f}$$

با تبدیل خطی فوق، از رخداد همگرایی زودرس و سکون جمعیت جلوگیری می‌کنیم. به جای در نظر گرفتن شاستگی اصلی، تبدیلی خطی از آن را در نظر می‌گیریم. به این ترتیب روشی دیگر به مجموعه قبلی افزوده خواهد شد:

۵- انتخاب متناسب با شایستگی مقیاس شده خطی

در این روش ابتدا f به \bar{f} تبدیل می‌شود و به این ترتیب خواهیم داشت: $p_i = f'_i / \sum f'_i$. سپس می‌توانیم از یکی از روش‌های SUS ، RW و یا روش سوم گفته شده استفاده کنیم. گرچه روش فوق درگیر با شایستگی است، با این حال انتخاب صرفاً بر اساس شایستگی را از بین می‌برد. روش دومی که می‌توان در انتخاب متناسب با شایستگی از آن استفاده کرد در نظر گرفتن $p_i = af_i + b$ است. پارامترهای a و b را طوری انتخاب می‌کنیم که فشار انتخاب را در اختیار داشته باشیم.

$$\begin{cases} \sum p_i = 1 \\ SP = NP_b = N(af_b + b) \\ \frac{SP}{N} = af_b + b \\ \frac{a \sum f_i}{N} + \frac{bN}{N} = \frac{1}{N} \\ a\bar{f} + b = \frac{1}{N} \\ \frac{SP - 1}{N} = a(f_b - \bar{f}) \\ a = \frac{SP - 1}{N} \cdot \frac{1}{f_b - \bar{f}} \end{cases}$$

به این ترتیب خواهیم داشت:

$$b = \frac{1}{N} - \frac{SP - 1}{N} \cdot \frac{\bar{f}}{f_b - \bar{f}}$$

$$p_i = \frac{SP - 1}{N} \cdot \frac{f_i - \bar{f}}{f_b - \bar{f}} + \frac{1}{N}$$

مشابه آن چه پیش‌تر دیدیم، بعد از به دست آوردن p_i می‌توان از یکی از روش‌های SUS ، RW و روش سوم گفته شده استفاده کرد.

با این روش می‌توان فشار انتخاب را مابین ۱ و ۲ کنترل کرد یعنی: $2 < SP < 1$. برای به دست آوردن SP بالاتر از ۲ نیازمند اشتفاده از روش‌های غیرخطی خواهیم بود.

۶- انتخاب متناسب با شایستگی مقیاس شده غیرخطی

یکی از تبدیل‌های غیرخطی که می‌تواند مورد استفاده قرار گیرد به صورت زیر است:

$$P_i = \frac{e^{-\frac{f_i}{T}}}{\sum e^{-\frac{f_j}{T}}}$$

در حالت کلی در نظر می‌گیریم:

$$P_i = ae^{bf_i}$$

باز هم مانند حالت قبل باید داشته باشیم:

$$\begin{cases} \sum p_i = 1 \\ SP = NP_b = Nae^{bf_b} \end{cases}$$

و مشابه حالت قبل مقادیر a و b به دست می‌آیند.

برای رابطه غیرخطی فوق مقادیر به شرح زیر خواهد بود:

$$a = \frac{1}{\sum e^{-\frac{f_j}{T}}}, b = -\frac{1}{T}$$

فشار انتخاب ۱ یعنی حداقل یک کپی از بهترین موجود داشته باشیم. به این ترتیب برای فشار انتخاب ۲، با توان‌های ۲ افزایش یافته و به سرعت همگرا می‌شود. همان‌گونه که گفتیم فشار انتخاب، تنوع و همگرایی را کنترل می‌کند. فشار انتخاب نزدیک یک تنوع بالاتر ایجاد می‌کند و فشار انتخاب نزدیک ۲، همگرایی.

$$\sum P_i = a \sum e^{bf_i} = 1$$

$$\frac{SP}{N} = \frac{1}{\sum e^{bf_i}} e^{bf_b}$$

$$e^{bf_b} = \frac{SP}{N} \sum e^{bf_i}$$

به این ترتیب $a \cdot b = \frac{\ln A}{f_b}$ و خواهیم داشت:

$$P_i = \frac{e^{bf_i}}{\sum e^{bf_i}} = \frac{e^{\ln(\frac{SP}{N}) \cdot \frac{f_i}{f_b}}}{\sum e^{\ln(\frac{SP}{N}) \cdot \frac{f_i}{f_b}}}$$

سایر روش‌های غیرخطی

۷- انتخاب مناسب با شایستگی رتبه بندی شده

برای حل مشکل همگرایی زودرس و سکون، راه حل ساده‌تری وجود دارد. به این منظور موجودات را رتبه بندی^۱ می‌کنیم. بدترین موجود، رتبه ۱ و بهترین موجود، رتبه μ را خواهند داشت.

به این ترتیب، $i = f_i$ و بدترین موجود ۱ و بهترین موجود f_μ .

$$P_i = \frac{2i}{\mu(\mu + 1)}$$

¹ rank

نقطه قوت روش رتبه بندی شده این است که نیازی به محاسبه مقدار دقیق شایستگی ها نخواهد بود. صرف مرتب کردن کفايت می کند. این مرتب کردن، بدون امتياز دهی، به آن چه در طبیعت رخ می دهد نزدیک تر است. لازم به ذکر است که غیر از سه روش اول معرفی شده برای انتخاب، در سایر روش ها، از جمله روش فوق، ابتدا احتمال P_i را به دست می آوریم و پس از آن از RW، SUS و یا روش سوم گفته شده استفاده خواهیم کرد. در رتبه بندی اگر شایستگی مشخص باشد، موجودات بر اساس آن مرتب می شوند و در غیر این صورت توسط متخصص.

-۸- انتخاب مناسب با شایستگی رتبه بندی شده خطی

مجدداً P_i را به فرم کلی زیر در نظر می گیریم:

$$P_i = ai + b$$

$$\sum P_i = 1$$

$$a \sum i + b\mu = 1$$

$$\frac{1}{2}(a\mu(\mu+1)) + b\mu = 1$$

$$\frac{a(\mu+1)}{2} + b = \frac{1}{\mu}$$

$$SP = \mu P_b = \mu(a\mu + b)$$

$$a\mu + b = \frac{SP}{\mu}$$

$$b = \frac{SP}{\mu} - a\mu = \frac{SP}{\mu} - \frac{2(SP-1)}{\mu-1}$$

$$\frac{a(2\mu-\mu-1)}{2} = \frac{SP-1}{\mu}$$

$$a = \frac{SP-1}{\mu} \cdot \frac{2}{\mu-1}$$

$$b = \frac{1}{\mu} - \frac{SP-1}{\mu} \cdot \frac{2}{\mu-1} \cdot \frac{\mu+1}{2}$$

به این ترتیب خواهیم داشت:

$$P_i = \frac{SP-1}{\mu-1} \left(\frac{2i}{\mu} - 2 \right) + \frac{SP}{\mu}$$

$$P_i = 2 \left(\frac{SP-1}{\mu-1} \right) \cdot \frac{i-\mu}{\mu} + \frac{SP}{\mu}$$

همچنین در این حالت می توان حالت غیرخطی نیز داشت. مثلاً:

^۱ - روش تورنومنت

^۱ Tournoment

روش تورنومنت دو ورژن دارد. تورنومنت^۱ Q یکی از بهترین روش‌های انتخاب است که علاوه بر مزایای روش‌هایی که تاکنون معرفی شدند، ویژگی‌های دیگری نیز دارد. در این روش Q موجود را به صورت تصادفی یکنواخت از جمیعت انتخاب می‌کنیم (با یا بدون جایگذاری). سپس از بین Q موجود مذکور بهترین را انتخاب می‌کنیم. مقادیر Q به صورت $5 < Q < 2$ انتخاب می‌شوند. معمولاً از $2 = Q$ استفاده می‌شود که به آن تورنومنت باینری^۲ گفته می‌شود.

روش تورنومنت، علاوه بر این که مشکل همگرایی زودرس و سکون جمیعت و محاسبه امتیاز را ندارد، حتی موجودات را مرتب^۳ نمی‌کند و به این ترتیب باز هم هزینه محاسباتی پایین‌تری دارد. در روش رتبه بندی که پیش از این معرفی شد، لازم بود بین μ موجود عمل مرتب کردن را انجام دهیم. اما روش حاضر تنها از بین Q موجود انتخاب شده (که تعدادشان ۲ تا ۵ است)، بهترین موجود را انتخاب می‌کند. مزیت قابل توجه این روش، توانایی فرار از کمینه‌های محلی است.

در نسخه دوم این روش، به ازای هر موجود، Q موجود به صورت تصادفی یکنواخت تولید می‌شود. و محاسبه می‌شود که موجود مذکور از چند تا از Q‌هایی که انتخاب کرده‌ایم بهتر است. این کار برای تمام موجودات انجام می‌شود و سپس موجوداتی را هم لازم داریم از بین آن‌ها که بالاترین امتیاز را دارند، به ترتیب برمی‌داریم.

۱-۴-۴-۱ مدل‌های انتخاب

دیدیم که در الگوریتم‌های تکاملی در دو بخش انتخاب والدین و بازماندگان انتخاب داریم. در انتخاب والدین انتخاب از بین μ حالت انجام می‌گیرد. در انتخاب بازماندگان دو مجموعه جمیعت جاری و فرزندان را داریم. بنابراین دو مدل زیر را خواهیم داشت:

$$\begin{cases} \lambda, \mu \\ \lambda + \mu \end{cases}$$

در مدل μ, λ ، فقط از بین فرزندان μ موجود برای نسل بعد انتخاب می‌شوند. در مدل دوم بازماندگان (نسل بعد) از $\mu + \lambda$ انتخاب می‌شوند.

۱-۴-۴-۲ ویژگی‌های مدل‌ها

بارزترین ویژگی مدل μ, λ بی‌حافظه بودن یا به تعبیری فراموشکاری آن است که سبب افزایش تنوع و کاهش سرعت همگرایی می‌شود. این روش اساساً تضمینی برای همگرایی ندارد. البته بی‌حافظه بودن، مزیت توانایی فرار از کمینه‌ها و بیشینه‌های محلی را برای این روش ایجاد می‌کند. این روش به علت بی‌حافظه بودن، علاوه بر

¹ Q-tournament

² Binary Tournament

³ sort

کمینه‌های محلی، پارامترهای بد را نیز فراموش می‌کند. همچنین قادر است بهینه‌های متحرک را دنبال کند. مثلاً در ردبایی^۱ هواپیما بهتر از روش $\mu + \lambda$ است.

در مقابل، روش $\mu + \lambda$ حافظه‌دار است. به این ترتیب تنوع کمتر، سرعت همگرایی بالاتر و به علاوه تضمین همگرایی دارد.

توجه به این نکته ضروری است که الگوریتم‌های تکاملی، مکانیسم‌های مختلفی برای فرار از کمینه‌های محلی و به تعییر دیگر تنوع وجود دارد. از جمله، دو مدل گفته شده در بالا، نوع انتخاب، جهش و... با این حال از همه مکانیسم‌ها به طور همزمان استفاده نمی‌شود چرا که استفاده از چندین مکانیسم فرار، سبب خواهد شد به جواب نرسیم. همان گونه که پیش‌تر نیز اشاره شد باید بتوانیم مابین تنوع و همگرایی تعادل برقرار کنیم، در غیراین صورت اساساً تکامل نخواهیم داشت.

۲-۴-۴-۱۷-۱ **حالت‌های خاص**

-۱ اگر $\lambda = 1$ باشد خواهیم داشت:

$$\begin{cases} EA(1+1) \\ EA(1,1) \end{cases}$$

در $(1+1)EA$ ، از بین فرزند و جمعیت، بهترین به نسل بعد خواهد رفت. حالت $(1,1)EA$ را نمی‌توانیم داشته باشیم. چون کاملاً تصادفی خواهد بود و شایستگی در آن نقشی نخواهد داشت.

-۲ اگر فقط $\lambda = 1$ باشد:

$$\begin{cases} EA(1+\lambda) \\ EA(1,\lambda) \end{cases}$$

در $(1+\lambda)EA$ از جمعیت اولیه (1) با جهش، λ فرزند تولید می‌شود. سپس از مجموع این λ فرزند تولید شده و جمعیت اولیه، نسل بعد انتخاب می‌گردد.

-۳ اگر $\lambda = 1$ باشد خواهیم داشت:

$$\begin{cases} EA(\mu+1) \\ EA(\mu,1) \end{cases}$$

حالت $(\mu,1)EA$ ممکن نیست؛ چرا که نمی‌توانیم 1 فرزند داشته باشیم و μ تا از آن انتخاب کنیم.

¹ track

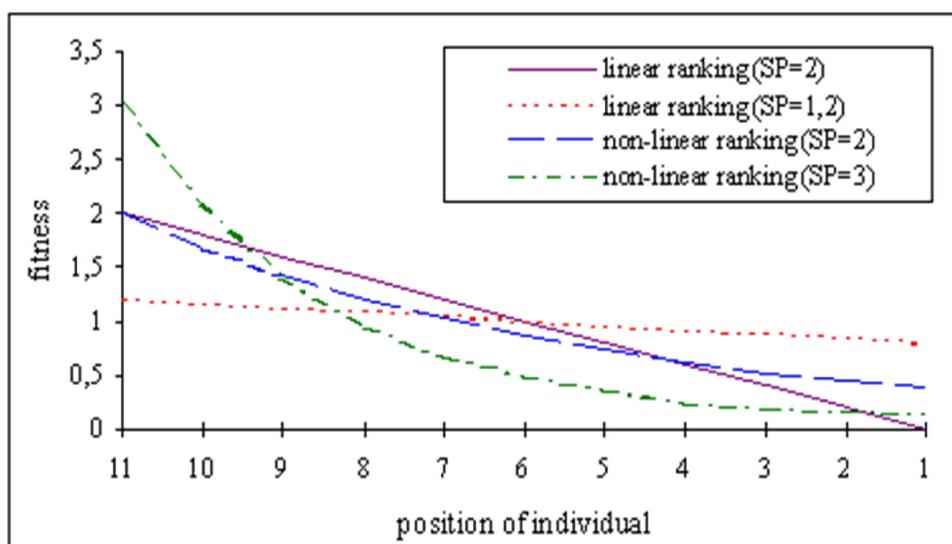
حالت $(\mu + 1)$ را در صورت تولید تصادفی فرزند، حالت پایدار^۱ می‌گویند. اگر تنها فرزند تولید شده جایگزین بدترین موجود شود، آن را جنتور^۲ گوییم. جنتور، حالت پایداری است که در آن یک فرزند تولید شده جایگزین بدترین موجود می‌شود.

-۴- اگر $\mu = \lambda$ باشد؛ μ فرزند تولید کرده و همگی را به نسل بعد می‌بریم. به این حالت نسلی^۳ گفته می‌شود و الگوریتم ژنتیک استاندارد به این صورت است.

در (μ, μ) شایستگی درگیر نیست. انتخاب والدین حتماً باید بر اساس شایستگی باشد. در غیر این صورت الگوریتم تکاملی نخواهیم داشت. در حالت کلی می‌توانیم $EA(\mu, \lambda)$ داشته باشیم که می‌بایست $\mu \geq \lambda$ باشد. در حالت $(\mu + \lambda)$ محدودیت فوق را نداریم. در حالتی که $\mu \leq \lambda$ باشد، $EA(\mu + \lambda)$ را شکاف نسلها^۴ می‌نامند.

۱-۴-۱۷-۵ جمع بندی روش‌های انتخاب

در این بخش ویژگی‌های گفته شده برای روش‌های مختلف را به صورت تصویری جمع بندی می‌کنیم. در شکل زیر روش‌های رتبه بندی خطی و غیرخطی با چند مقدار SP مقایسه شده‌اند.



همان گونه که مشاهده می‌شود به صورت خطی نمی‌توان SP بیش از ۲ داشت چون در این صورت برخی شایستگی‌ها منفی خواهند شد.

¹ steady state

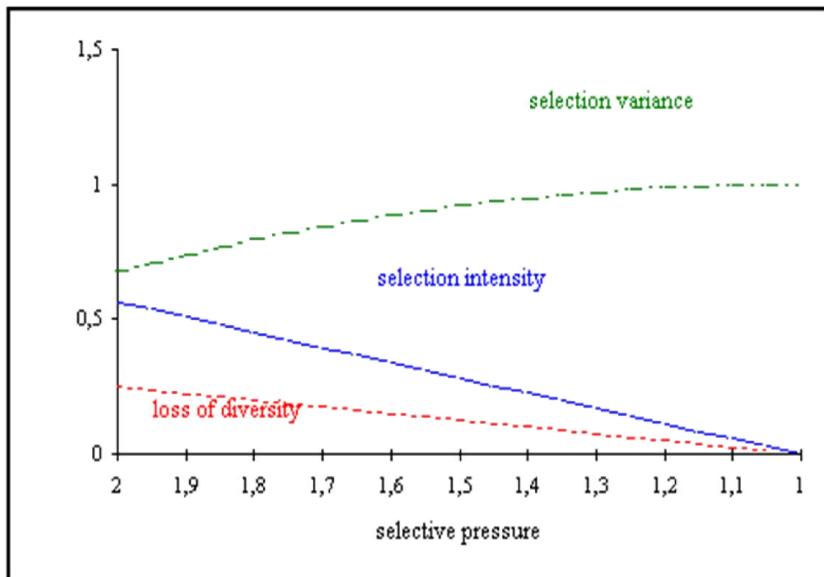
² genitor

³ generational

⁴ generational gap

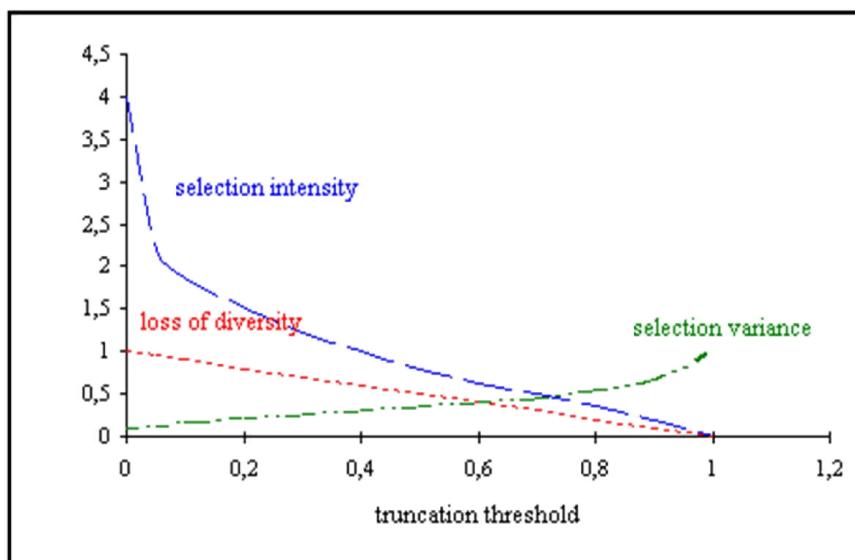
۱-۵-۴-۱۷-۱ برای انتخاب خطی

شکل زیر مقادیر LOD، SI و SV را برای انتخاب خطی نمایش می‌دهد. همان طور که مشاهده می‌شود این مقادیر بر اساس SP نمایش داده شده‌اند. مشاهده می‌شود که مقادیر SI و LOD نسبت به افزایش SP سیر نزولی دارند و بر خلاف این دو، SV با افزایش SP افزایش پیدا می‌کند.



۱-۵-۴-۱۷-۲ برای انتخاب برشی

نومدار زیر سه پارامتر LOD، SI و SV را بر اساس آستانه برش^۱ نمایش می‌دهد. مشاهده می‌شود که SI و LOD نسبت به افزایش پارامتر مذکور روند کاهشی دارند و SV با افزایش آستانه برش، افزایش خواهد یافت.

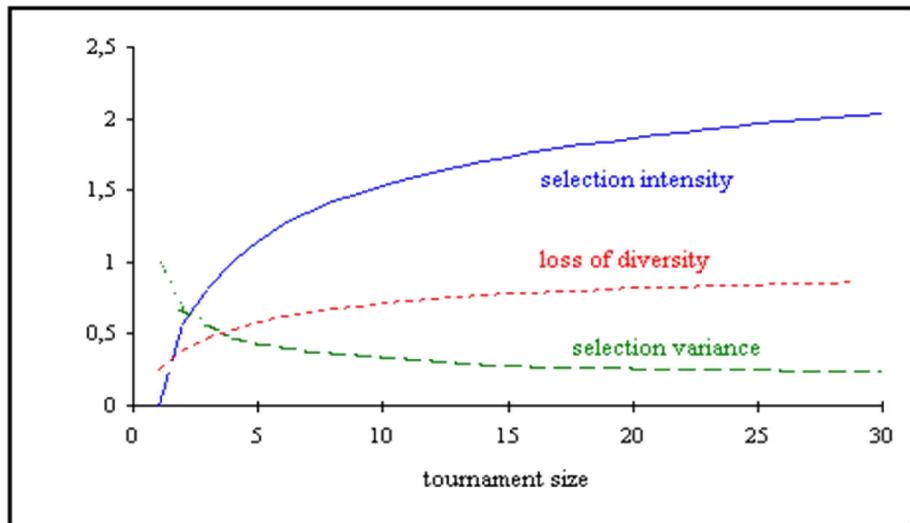


به این معناست که چند درصد از موجودات انتخاب شوند.

^۱ Truncation threshold

۱-۴-۵-۳- برای تورنومنت

شکل زیر تغییرات سه پارامتر SV , SI و LOD را بر حسب اندازه تورنومنت^۱ (Q) نمایش می‌دهد. و همان گونه که مشاهده می‌شود SV با افزایش Q کاهش می‌یابد در حالی که SI و LOD افزایش خواهد داشت.



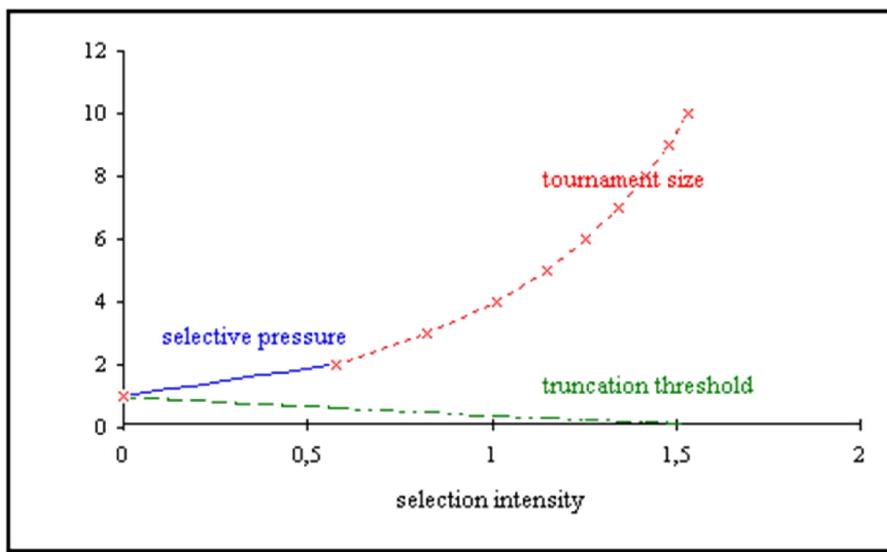
در این حالت مقادیر بالاتر Q (اندازه تورنومنت) معادل تنوع کمتر و سرعت همگرایی بیشتر است.
اگر Q را تمام جمعیت در نظر بگیریم و روند انتخاب بدون جایگذاری باشد، معادل انتخاب برشی خواهد بود.
اگر این کار را با جایگذاری انجام دهیم، یک موجود مدام انتخاب خواهد شد. به بیان دیگر همه موجودات برتر خواهند شد.

علاوه‌نماییم هم سرعت همگرایی و هم تنوع بالایی داشته باشیم. همان گونه که در شکل مشاهده می‌شود، از جایی به بعد شبیب SV کند می‌شود و به تعبیری تأثیر خاصی بر همگرایی ندارد اما در عین حال تنوع را از بین می‌برد. این مقدار حدود ۵ است و بر همین اساس است که Q مابین ۲ و ۵ انتخاب می‌شود. برای جمعیت کوچک معمولاً از $Q = 2$ و برای جمعیت بزرگ از $Q = 5$ استفاده می‌شود.

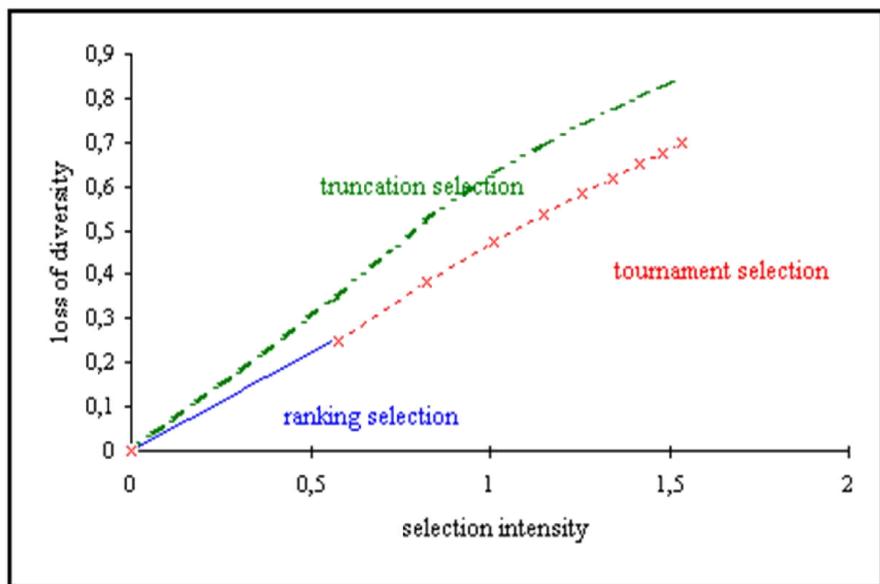
۱-۴-۶- مقایسه سه روش

نمودار زیر سه روش برشی، رتبه بندی معمولی و Q -tournament را مقایسه می‌کند.

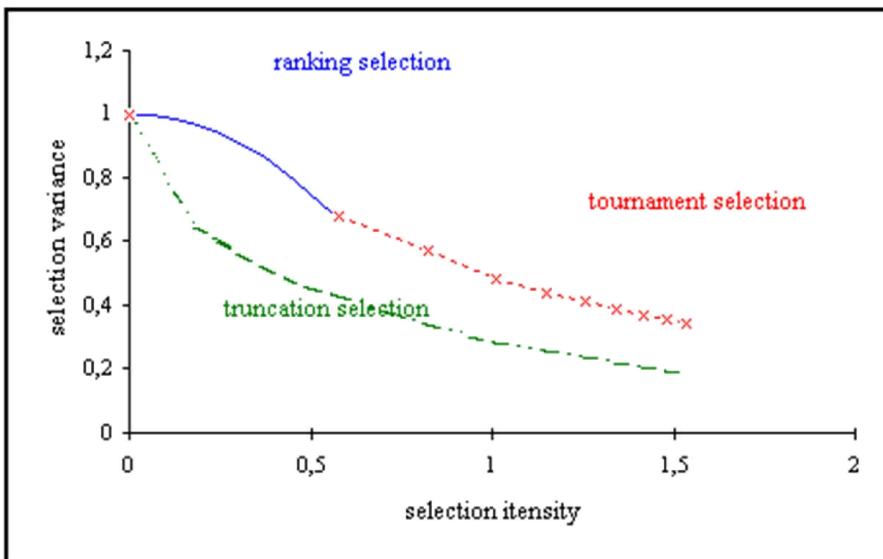
^۱ Tournament Size



در این شکل منحنی‌ها بر حسب SI رسم شده‌اند. به ازای مقادیر SI صفر تا ۱ روش برشی، ۱ تا ۲ رتبه‌بندی معمولی و به ازای مقادیر بزرگ‌تر از ۲، تورنومنت Q را داریم. شکل زیر مقایسه سه روش را با رویکردی دیگر نمایش می‌دهد. محور افقی SI و محور عمودی LOD است.



در SI برابر، روش تورنومنت Q تنوع بالاتری نسبت به روش برشی ایجاد می‌کند. به علاوه اگر روش رتبه‌بندی را در SP های بالاتر از ۲ ادامه دهیم تورنومنت Q خواهیم داشت. این موضوع در شکل هم دیده می‌شود که منحنی تورنومنت Q دقیقاً در ادامه منحنی رتبه‌بندی آمده است.



در هر دو شکل بالا مشاهده می‌کنیم که رتبه بندی خطی و تورنومنت Q عملکرد بهتری نسبت به روش برشی دارند.

دو روش دیگر نیز مطرح هستند:

f^k که به معنای توان k م شایستگی است. اگر $1 > k$ باشد انسپاس و اگر $1 < k$ باشد انقباض خواهیم داشت.
روش بعدی بولتزمان^۱ است.

همچنین برای رتبه بندی نمایی رابطه زیر را داریم:

$$\text{exponential ranking: } P_{\text{exp-rank}}(i) = \frac{1-e^{-i}}{c}$$

شاپیسته سالاری^۲ : به معنای این است که الزاماً کسی ای از بهترین موجود را نگه داشته یا آرشیو کنیم. آرشیو کردن به معنای این است که کسی ای از بهترین موجود داریم اما در تکامل شرکت نمی‌کند. در صورت شرکت در تکامل می‌گوییم موجود را نگه داشته‌ایم. شاپیسته سالاری سبب می‌گردد هیچ وقت بهترین موجود را گم نکنیم. در واقع مهم است که بهترین پاسخی که تا به حال به دست آمده است نباید از بین برود و گم شود و این احتمال در روش‌هایی که تاکنون معرفی کردیم وجود دارد.

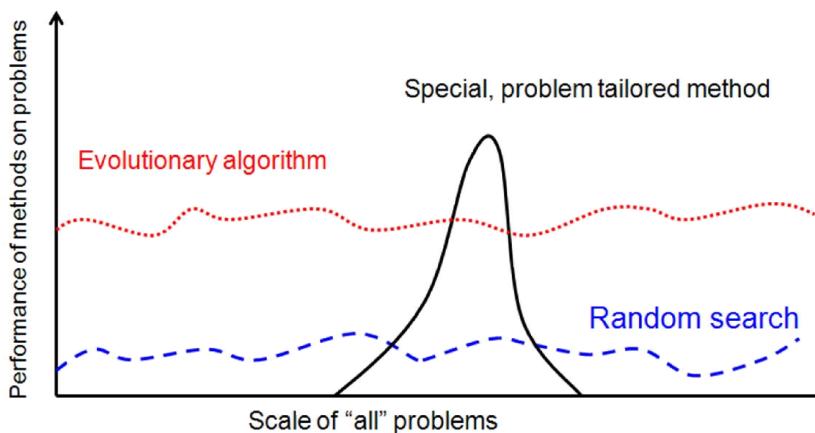
مشخصات عمومی الگوریتم‌های تکاملی

الگوریتم‌های تکاملی شامل سه فاز هستند:

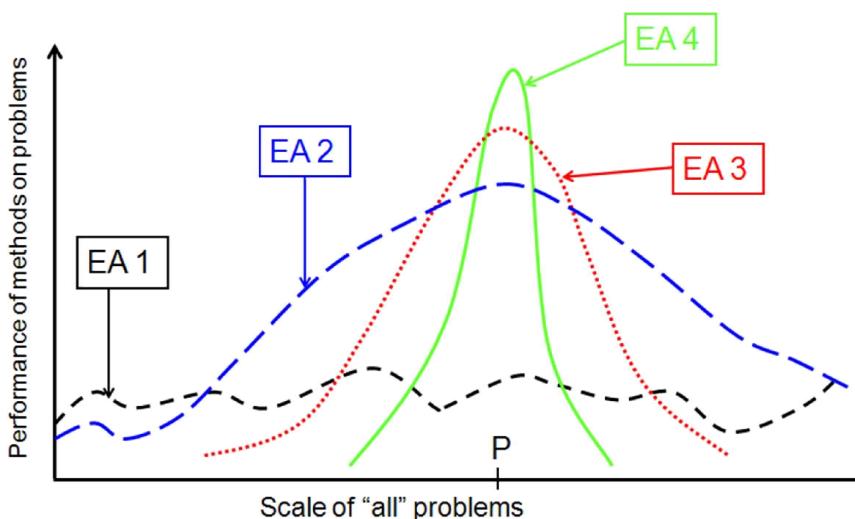
- ۱ - فاز اولیه که موجودات به صورت تصادفی یکنواخت در تمام فضا پخش می‌شوند.
- ۲ - فاز میانی که به قله‌ها نزدیک می‌شوند.
- ۳ - فاز پایانی که به جواب (قله) می‌رسند.

¹ Boltzman
² Elitism

همان طور که گفتیم الگوریتم‌های تکاملی عملکردی قابل قبول برای تمام مسائل دارند چرا که به مسئله وابستگی ندارند. در این روش‌ها صرف بہتر بودن موجود کافی است و حتی نیازی به تعیین شایستگی نیست.



شکل بالا نشان می‌دهد که الگوریتم‌های تکاملی مانند جستجوی تصادفی^۱ محدوده بسیار وسیع‌تری از مسائل را پوشش می‌دهند. به علاوه، به نسبت جستجوی تصادفی عملکرد بهتری دارند. همچنین مشاهده می‌شود الگوریتم‌هایی که با منظور خاص طراحی شده‌اند، محدوده کوچک‌تری از مسائل را پوشش می‌دهند و با این وجود عملکردشان چندان بالاتر از الگوریتم‌های تکاملی نیست.



شکل بالا نشان می‌دهد که هر چه الگوریتم‌های تکاملی را به مسئله‌ای خاص محدود کنیم، عرض منحنی‌ها کمتر و ارتفاع قله‌های آن‌ها افزایش می‌یابد. به این معنی که مسائل تحت پوشش کمتر شده و در مقابل عملکرد ارتقا می‌یابد.

تئوری‌ای وجود دارد که می‌گوید وقتی چیزی به دست می‌آوریم باید برای آن هزینه‌ای بپردازیم. مثلاً تغییر در بازنگری و جهش، و به این ترتیب، خاص کردن الگوریتم برای مسئله‌ای خاص.

ویژگی‌های الگوریتم‌های تکاملی

¹ Random Search

- عمومی هستند
- هزینه قابل قبولی دارند
- به سادگی قابل تفسیر هستند
- همزمان چند پاسخ پیشنهاد می‌دهند

در مقابل مزایای گفته شده در بالا، دارای ضعف‌های زیر هستند:

- عدم ضمانت همگرایی
- ضعف تئوری
- نیاز به تنظیم پارامترها
- سرعت پایین در حالت کلی

موارد استفاده از الگوریتم‌های تکاملی

- مسائل پیچیده با تعداد زیادی پارامتر
- وجود روابط پیچیده مابین پارامترها
- مسائل دارای پارامترهایی از انواع مختلف
- مسائل با بهینه‌های محلی متفاوت
- مسائل نویزی

و در مجموع می‌توان گفت که الگوریتم‌های تکاملی برای حل کلیه مسائل با هر درجه از پیچیدگی و تعداد پارامتر کارآیی دارند.

تا به اینجا در بازه بازنمایی، جمعیت و انتخاب‌ها صحبت کردیم. در ادامه به شرط خاتمه به عنوان یکی دیگر از بخش‌های بلاک دیاگرام کلی الگوریتم‌های تکاملی خواهیم پرداخت:

شرط خاتمه

همان گونه که اشاره شد، در مواردی ممکن است الگوریتم‌های تکاملی واگرا نشوند. به این ترتیب عملاً پایانی برای فرآیند محاسباتی آنها وجود ندارد. به این ترتیب لازم است شرایطی را به لحاظ محاسباتی و یا زمانی برای توقف الگوریتم در نظر گرفته و در آن لحاظ کنیم. بر این اساس چند شرط خاتمه متعارف را معرفی می‌کنیم:

۱-۵-۱ تعداد نسل

ساده‌ترین شرط خاتمه، تعداد نسل یا تعداد ارزیابی است. وقت داشته باشید که این دو متفاوت هستند. عموماً از تعداد ارزیابی استفاده می‌شود.

مزیت این شرط خاتمه سادگی است و تنها شرطی است که حتماً تمام خواهد شد. عیب این روش این است که اگر تعداد کم باشد ممکن است به جواب نرسیم و اگر زیاد باشد، زمان را هدر می‌دهیم. به علاوه، پیدا کردن تعداد نسل‌هایی که باید داشته باشیم کار ساده‌ای نیست.

۱-۱۷-۵-۲ تعریف بگ حد

اگر شایستگی بهینه را بدانیم می‌توانیم از آن برای خاتمه استفاده کنیم. اما معمولاً شایستگی بهینه را نمی‌دانیم. در جاهایی که می‌دانیم، مانند مسأله هشت وزیر، حدی تعریف می‌کنیم که با رسیدن به آن از الگوریتم خارج شود. این تنها روشی است که اگر از الگوریتم خارج شود، تضمین می‌کند به جواب رسیده‌ایم و این مزیت روش مذکور است. عیب این روش این است که ممکن است الگوریتم تمام نشود و هیچ وقت به جواب نرسد. مسأله دیگر این است که معمولاً شایستگی را نمی‌دانیم.

۱-۱۷-۵-۳ بررسی همگرایی

اگر در k نسل متوالی تغییرات شایستگی بهینه هر نسل، از حد آستانه‌ای کمتر باشد الگوریتم متوقف می‌شود. می‌توان به جای شایستگی بهینه، متوسط شایستگی نسل در نظر گرفته شود. در برخی موارد شایستگی بهینه و برخی دیگر بیشترین شایستگی نسل محاسبه می‌شود.

مزیت این روش این است که وقتی همگرایی (زودرس یا واقعی) داریم از الگوریتم خارج می‌شود و زمان را هدر نمی‌دهد. چون در صورت عدم خروج از الگوریتم ادامه دادن عملاً بی‌فایده و اتلاف زمان خواهد بود. عیب این روش این است که ممکن هیچ وقت اتفاق نیفتد. اگر از الگوریتم خارج شویم، تضمینی ندارد که جواب را یافته باشیم.

۱-۱۷-۵-۴ بررسی تنوع

در k نسل آخر، واریانس ژنتیپ جمعیت را محاسبه می‌کنیم. اگر واریانس از آستانه‌ای کمتر باشد تنوع نداریم و به تبع آن تکامل نخواهیم داشت و می‌بایست از الگوریتم خارج شویم.

مزیت این روش این است که با توقف تکامل، از الگوریتم خارج می‌شود و زمان را هدر نمی‌دهد. در مقابل، عیب این روش این است که تضمینی ندارد به جواب رسیده باشیم.

همگرایی واریانس شایستگی و تنوع، واریانس ژنتیپ را محاسبه می‌کنند. باید توجه داشت که در بسیاری موارد واریانس ژنتیپ و فونوتیپ (شایستگی) یکی هستند چرا که الگوریتم‌های تکاملی بالذات تمایل دارند به یک قله همگرا شوند. در مسائل چند جوابی می‌توانیم برای چند ژنتیپ مختلف (چند کد) شایستگی یکسان داشته باشیم.

در خصوص شرط‌های خاتمه گفته شده باید گفت که هیچ یک از آن‌ها به تنها یعنی مناسب نیستند. به این ترتیب در مسائل کاربردی (غیر از شرط دوم که همیشه نیز عملی نیست) معمولاً از مجموعه این شروط استفاده می‌شود. در یک مسأله واقعی، زمان حداکثری را که در اختیار داریم محاسبه می‌کنیم. بر اساس آن تعداد نسلی را که در آن از

الگوریتم خارج می‌شویم به دست می‌آید. در کنار این شرط، شرط‌های دیگر را نیز قرار می‌دهیم. به این ترتیب اگر شرط‌های مذکور نتوانستند زودتر به الگوریتم خاتمه دهند، نهایتاً شرط تعداد نسل، آن را خاتمه خواهد داد. توجه داشته باشید که در مسائل تحقیقاتی تئوری، تنها از تعداد نسل استفاده می‌شود.

جهش^۱ و بازترکیبی^۲

پیش‌تر دیدیم که در روند تولید مثل دو فرآیند بازترکیبی و جهش وجود دارند. از این دو به عملگرهای ژنتیکی یاد می‌شود. در طبیعت، جهش و بازترکیبی به ترتیب جستجوی محلی و عمومی را انجام می‌دهند. گرچه بهتر است در الگوریتم‌های تکاملی، یکی از عملگرها جستجوی محلی و دیگری جستجوی عمومی را انجام دهد، اما معمولاً با قاطعیت نمی‌توان عملگرهای انجام دهنده این دو جستجو را از یکدیگر تفکیک کرد و یا عملگری را خوب یا بد دانست. همان‌گونه که در طبیعت نیز اگر جهش زیاد باشد، می‌تواند جستجوی عمومی را هم انجام دهد. ایده بازترکیبی این است که ویژگی‌های خوب والدین کنار هم قرار گرفته و موجودی شایسته‌تر را به وجود آورند. البته بازترکیبی به تنهایی ممکن است منجر به کنار هم قرار گرفتن بد نیز بشود. اما مجموعه تنوع و بازترکیبی، منجر به کنار هم قرار گرفتن ویژگی‌های خوب می‌گردد.

جهش، بر خلاف بازترکیبی، بر روی یک فرد ایجاد شده و منجر به ایجاد فرد جدیدی می‌گردد. در جهش، عنصر تصادف ضروری است و همین عامل است که وجه تمایز جهش از عملگرهای هیورستیک دیگر به شمار می‌آید. اهمیت و کاربرد جهش به روش ارائه و الگوریتم تکاملی مربوطه وابسته است که در فصول آتی به تفصیل نمونه‌های آن را خواهیم دید.

جهش با امید تولید ویژگی‌های خوب انجام می‌پذیرد و عملگرهای بازترکیبی و جهش برای کنار هم قرار دادن و یا ایجاد ویژگی‌های خوب مورد استفاده قرار می‌گیرند.

دیدیم که اگر همگرایی داشته باشیم و تنوع نداشته باشیم، تکامل وجود نخواهد داشت. برای داشتن تکامل، وجود تنوع لازم است. در الگوریتم‌های تکاملی، تنوع می‌تواند تعداد ورودی^۳ برابر ۱، ۲ و یا بالاتر از آن داشته باشد. هدف از عملگرهای ژنتیکی، تولید راه حل‌های جدید از راه حل‌های موجود است. مقصود از تعداد ورودی، تعداد موجودات لازم برای انجام عمل مورد نظر است. در تعداد ورودی برابر ۱، جهش، در $2 > \text{arity}$ بازترکیبی و در حالت خاص برابر ۲، تقاطع^۴ خواهیم داشت.

¹ Mutation

² Recombination

³ Arity

⁴ Cross-over

۱-۶-۱۷-۱ عملگرهای بازترکیبی

در این بخش به عملگرهای بازترکیبی خواهیم پرداخت. بازترکیبی در واقع تولید فرزندان با ترکیبی ویژگی‌های والدین است. انتخاب اطلاعات از والدین به صورت تصادفی انجام می‌شود. بدیهی است که احتمال تولید فرزندان نامناسب و یا مشابه والدین در این روش زیاد است. با اسن حال استفاده از بازترکیبی با امید ترکیب ویژگی‌های مناسب والدین و تولید جواب‌های بهتر انجام می‌گیرد.

شایان توجه است که عملگرهای ژنتیکی برخلاف روش‌های انتخاب، به نحوه بازنمایی وابسته‌اند.

۱-۶-۱۷-۱-۱ بازترکیبی تک نقطه‌ای

در این حالت سه رخداد تصادفی داریم:

- والدهایی که با یکدیگر جفت می‌شوند.
- رخداد و یا عدم رخداد بازترکیبی بین والدهای مذکور
- نقطه شروع بازترکیبی

پس از مرحله اول که مشخص می‌کند کدام یک از والدها قرار است با یکدیگر جفت شوند، در مرحله دوم احتمالی برای رخداد بازترکیبی داریم. در صورت صفر بودن این احتمال، بازترکیبی رخ نداده و خود والدها به مرحله بعد خواهند رفت. و چه خود والدها به مرحله بعد بروند و چه حاصل بازترکیبی آن‌ها، ممکن است جهش روی آن‌ها داشته باشیم.

برای روشن‌تر شدن موضوع، دو والد زیر و نقطه مشخص شده برای بازترکیبی روی آن‌ها را در نظر بگیرید:



0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1

اعمال بازترکیبی تک نقطه‌ای منجر به تولید موجود زیر خواهد شد:

0	0	1	1	1	1	1	1
---	---	---	---	---	---	---	---

بازترکیبی تک نقطه‌ای، در ابتدا و انتهای بایاس مکانی دارد. به تعبیر دیگر، احتمال تغییر کردن ابتدا و انتهای کروموزوم پایین است. در واقع، مانند RW است، به این معنی که احتمال تغییر ابتدا و انتهای در آن‌ها خیلی کمتر است.

مزیت این روش، تمایل به حفظ ژن‌های مجاور هم است. با این حال، هرگز ژن‌های ابتدا و انتهایی یک فرد همزمان حفظ نخواهد شد. در خصوص مشکل بایاس نیز، افزایش تعداد نقاط بازترکیبی سبب بهبود بایاس خواهد شد. نمونه‌ای از بازترکیبی چند نقطه‌ای در شکل زیر آمده است:

parents	<table border="1" style="border-collapse: collapse; text-align: center;"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1						
0	0	0	0	0	0	0	0	0	0	0	0	0	0																						
1	1	1	1	1	1	1	1	1	1	1	1	1	1																						
children	<table border="1" style="border-collapse: collapse; text-align: center;"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td></tr></table>	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	1																			
1	1	1	1	1	0	0	0	1	1	1	1	1	1	0	0	0																			

کارآیی بازترکیبی تک نقطه‌ای در شرایطی است که ساختار مسأله معلوم باشد که در عمل، معمولاً چنین نیست.

۱-۶-۲-باختگی یکنواخت

در این روش، برای هر بیت، یک عدد تصادفی مابین صفر و یک انتخاب می‌کنیم. اگر این عدد کوچک‌تر از نیم بود، بیت را معکوس کرده و در غیر این صورت خود بیت را خواهیم داشت. به تعبیر دیگر می‌توان از شیر و خط استفاده کرد. آمدن شیر را به یک والد و آمدن خط را به والد دیگر منتبث کنیم و بر اساس نتیجه پرتاب سکه، بیت متناظر در فرزند را از والد ۱ یا ۲ انتخاب کنیم. نمونه‌ای از این بازترکیبی در شکل زیر آمده است:

parents	<table border="1" style="border-collapse: collapse; text-align: center;"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1								
0	0	0	0	0	0	0	0	0	0	0	0	0	0																								
1	1	1	1	1	1	1	1	1	1	1	1	1	1																								
children	<table border="1" style="border-collapse: collapse; text-align: center;"><tr><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	0	1	0	0	1	0	1	1	0	0	0	1	0	1	1	0	0	1	1	0	1	1	0	0	0	0	1	1	1	0	1	0	0	1	1	0
0	1	0	0	1	0	1	1	0	0	0	1	0	1	1	0	0	1																				
1	0	1	1	0	0	0	0	1	1	1	0	1	0	0	1	1	0																				

در بازترکیبی یکنواخت، توارث هر ژن مستقل از مکان آن است. به علاوه در این روش بایاس مکانی وجود ندارد. اشاره کردیم که عملگرهای بازترکیبی و جهش به نحوه بازنمایی وابستگی دارند. در ادامه به بازترکیبی‌هایی که برای بازنمایی اعداد حقیقی ارائه شده‌اند، خواهیم پرداخت.

۱-۶-۲- بازترکیبی در بازنمایی عدد حقیقی

در بازنمایی باینری عملاً امکان ترکیب مقادیر ویژگی‌ها که به صورت ۰ و ۱ هستند، وجود ندارد. به علاوه در شرایطی که برای هر ویژگی یک بیت در نظر می‌گیریم، حجم محاسبات بسیار بالا خواهد بود.
در این شیوه بازنمایی، برای جهش با یک احتمال کوچک عددی تصادفی مابین ۰ و ۱ ایجاد می‌کنیم. اگر عدد مذکور از P_m کوچک‌تر شود، بیت را معکوس می‌کنیم و در غیر این صورت بدون تغییر باقی می‌گذاریم. P_m نیز به صورت زیر تعیین می‌شود:

$$\frac{1}{N} < P_m < \frac{1}{\mu}$$

که N تعداد ژن (بیت)‌های کروموزم و μ تعداد افراد جمعیت هستند.

در بازنمایی عدد حقیقی، برخلاف بازنمایی باینری، امکان بازترکیبی از میان ژن وجود ندارد. در طبیعت هر ژن تعدادی کدن دارد و امکان جابجایی از بین آن‌ها نیز وجود دارد. به این معنی که بخشی از یک والد و بخشی از والد دیگر بیاید. به این ترتیب، بازنمایی عدد حقیقی در این زمینه با آن چه در طبیعت وجود دارد همخوانی ندارد چرا که در این بازنمایی، هر ژن یک عدد حقیقی است و وسطی ندارد.
بازترکیبی در بازنمایی حقیقی می‌تواند به دو صورت گستته و محاسباتی (عددی) باشد.

۱-۶-۲-۱- بازترکیبی گستته

مقدار هر آلل تنها از یکی از والدین گرفته می‌شود. در این شیوه، مشابه حالت باینری می‌توان از شیر و خط استفاده کرد. به بیان دقیق‌تر، از کلیه روش‌های گفته شده برای ارائه باینری می‌توان استفاده کرد.

۱-۶-۲-۲- بازترکیبی عددی (محاسباتی)

در این شیوه که به آن میانه نیز گفته می‌شود، ایده تولید فرزندانی با ویژگی‌هایی بین ویژگی‌های والدین مطرح می‌گردد.

اگر والدین را به صورت زیر در نظر بگیریم،

	x_i	
--	-------	--

	y_i	
--	-------	--

ژن متناظر در فرزندان به صورت زیر است:

$$\begin{cases} x'_i = \alpha x_i + (1 - \alpha) y_i \\ y'_i = (1 - \alpha) x_i + \alpha y_i \end{cases}$$

پارامتر α می‌تواند ثابت، تصادفی و یا بر اساس تولید نسل متغیر باشد. اگر $\alpha = 0.5$ باشد، دقیقاً متوسط را خواهیم داشت.

بازترکیبی‌های عددی گونه‌های متفاوتی دارند که در ادامه به آن‌ها خواهیم پرداخت:

۱ - بازترکیبی تک حسابی^۱

در بازترکیبی تک حسابی در یک حالت تنها یک ژن به طور تصادفی انتخاب شده و فرآیند فوق برای آن انجام می‌گیرد. نمونه‌ای از این بازترکیبی را در شکل زیر برای $\alpha = 0.5$ آورده شده است:

0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
-----	-----	-----	-----	-----	-----	-----	-----	-----

0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.5	0.9
-----	-----	-----	-----	-----	-----	-----	-----	-----



0.3	0.2	0.3	0.2	0.3	0.2	0.3	0.2	0.3
-----	-----	-----	-----	-----	-----	-----	-----	-----

0.3	0.2	0.3	0.2	0.3	0.2	0.3	0.5	0.3
-----	-----	-----	-----	-----	-----	-----	-----	-----

به علاوه، ممکن است فرآیند گفته شده را برای جایی به بعد از دنباله، یا به تعبیر دقیق‌تر کروموزوم به کار ببریم. نمونه‌ای از این حالت (مجدداً برای $\alpha = 0.5$) در شکل زیر آورده شده است:

0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
-----	-----	-----	-----	-----	-----	-----	-----	-----

0.1	0.2	0.3	0.4	0.5	0.6	0.5	0.5	0.6
-----	-----	-----	-----	-----	-----	-----	-----	-----



0.3	0.2	0.3	0.2	0.3	0.2	0.3	0.2	0.3
-----	-----	-----	-----	-----	-----	-----	-----	-----

0.3	0.2	0.3	0.2	0.3	0.2	0.5	0.5	0.6
-----	-----	-----	-----	-----	-----	-----	-----	-----

۲ - بازترکیبی تمام حسابی^۲

در این روش که متدائل‌تر است، همان گونه که از نام آن پیداست، کلیه ژن‌ها تغییر خواهند کرد. اگر والدها را به ترتیب x_1, x_2, \dots, x_n و y_1, y_2, \dots, y_n در نظر بگیریم؛ فرزند اول به صورت

$$\alpha \cdot \bar{x} + (1 - \alpha) \cdot \bar{y}$$

خواهد بود. برای فرزند دوم، جای ضرایب عوض می‌شود.

شکل زیر نمونه‌ای را برای $\alpha = 0.5$ نمایش می‌دهد:

0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
-----	-----	-----	-----	-----	-----	-----	-----	-----

0.2	0.2	0.3	0.3	0.4	0.4	0.5	0.5	0.6
-----	-----	-----	-----	-----	-----	-----	-----	-----



0.3	0.2	0.3	0.2	0.3	0.2	0.3	0.2	0.3
-----	-----	-----	-----	-----	-----	-----	-----	-----

0.2	0.2	0.3	0.3	0.4	0.4	0.5	0.5	0.6
-----	-----	-----	-----	-----	-----	-----	-----	-----

¹ single arithmetic

² whole arithmetic

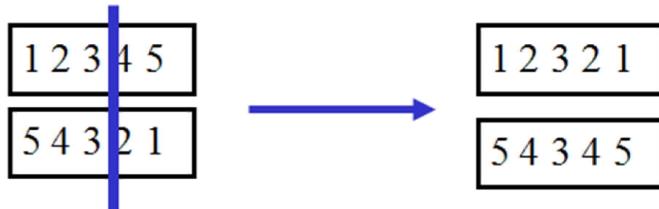
همان گونه که از رابطه فوق مشخص است، از این ایده استفاده می‌شود که اگر هر دو والد ژن‌های خوبی دارند، ممکن است مابین این ژن‌ها، بهتر باشد و اگر دارای ژن‌های خوبی نیستند، ممکن است بین آن‌ها خوب باشد. در پایین اشاره به این نکته خالی از لطف نیست که در بازنمایی اعداد حقیقی، جهش به صورت زیر صورت می‌گیرد:

$$x'_i = x_i + \sigma N(0,1)$$

۳-۲-۶-۱۷-۱ بازترکیبی در بازنمایی جایگشت

این روش برای تمامی مسائل کاربرد ندارد. اما در مواردی که استفاده از آن موضوعیت داشته باشد قادر است عملکرد بسیار خوبی را به نمایش بگذارد، به این دلیل که فضای حالت را کوچک می‌کند. خوب‌بختانه بسیاری از مسائل این قابلیت را دارند که به گراف تبدیل شده و با جایگشت مدل شوند.

با توجه به این که در جایگشت امکان استفاده تکراری از عناصر را نداریم، استفاده از روش‌های متعارف بازترکیبی منجر به تولید پاسخ‌های غیر قابل قبول می‌گردد. که نمونه‌ای از آن را در مثال زیر می‌بینید:



در بازترکیبی معمولی، فقط با کنار هم قرار گرفتن ویژگی‌های خوب سر و کار داشتیم. این در حالی است که در جایگشت، ترتیب نیز اهمیت دارد. از ترتیب می‌توان به عنوان جستجوی عمومی یاد کرد. مسئله دیگر حائز اهمیت، مجاورت (همسايگي) است که معادل جستجوی محلی در نظر گرفته می‌شود. در عملگرهای بازترکیبی ارائه شده برای بازنمایی جایگشت، تلاش شده است اطلاعات مربوط به ترتیب یا مجاورت ویژگی‌ها با يكديگر ترکيب شوند.

۱ - بازترکیبی مرتبه ۱

اساس این روش بر حفظ ترتیب نسبی عناصر است. در این روش بخشی از والد اول به تصادف انتخاب می‌شود. این بخش به صورت مستقیم در ژن‌های متناظر فرزند اول کپی می‌شود. عناصر باقی مانده از والد اول را مشخص می‌کنیم. این عناصر را با ترتیب قرار گیری‌شان در والد دوم می‌چینیم. به این صورت که از نقطه بعد از قسمت کپی شده شروع کرده و مانند يك حلقه عمل می‌کنیم.

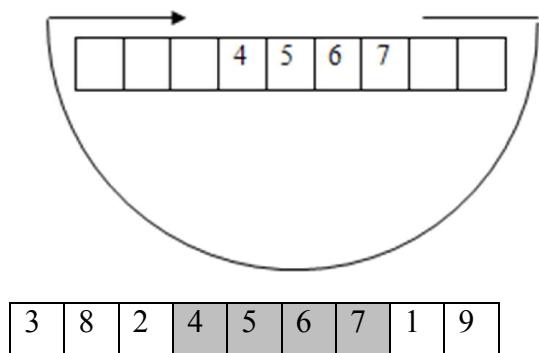
برای تولید فرزند دوم مشابه همین روند را با تغییر نقش والدین خواهیم داشت.

به عنوان مثال، فرض کنید والدها به صورت زیر باشند:

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

9	3	7	8	2	6	5	1	4
---	---	---	---	---	---	---	---	---

همچنین فرض کنید برای تولید فرزند اول، ژن‌های مشخص شده از والد اول را کپی می‌کنیم. با توضیحاتی که در بالا گفته شد، این فرزند به صورت زیر خواهد بود:



به صورت مشابه فرزند دوم برابر است با:

3	4	7	8	2	6	5	9	1
---	---	---	---	---	---	---	---	---

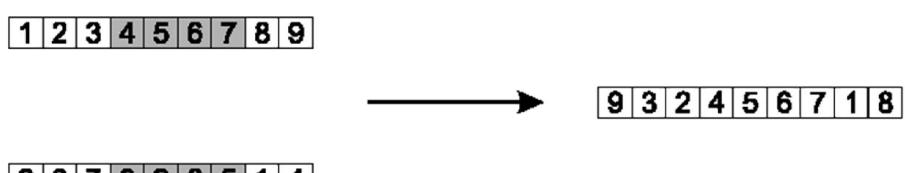
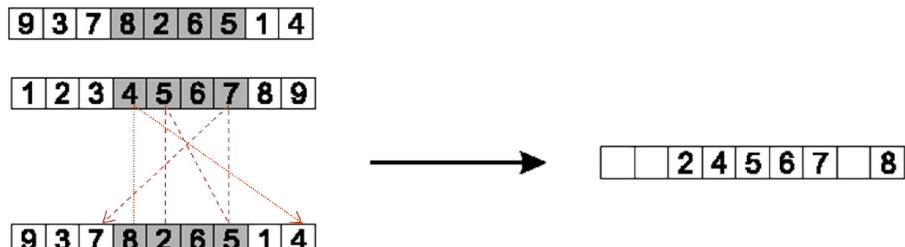
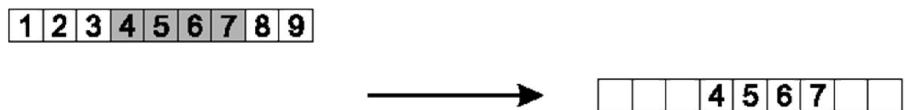
در این روش، در بخشی که مستقیماً کپی شده است ترتیب حفظ شده. به این ترتیب، اگر رشته بزرگ باشد جستجوی عمومی و اگر کوچک باشد جستجوی محلی خواهیم داشت.

۲- بازترکیبی تقاطع نسبتاً نگاشت شده (PMX)

تجییه روش^۱ PMX از روی روش‌های گرافیکی و به صورت تجربی است. چرا که حلقه‌هایی که بین دو والد ایجاد می‌شود از نظر گرافیکی معنی دارد. مشابه حالت قبل، یک زیر رشته را به صورت تصادفی از یکی از والدین انتخاب کرده و مستقیماً در جایگاه متناظر فرزند قرار می‌دهیم. فرض کنید این انتخاب از والد اول صورت گرفته است. از ابتدای بخش انتخاب شده، عناصری از والد دوم را که کپی نشده‌اند پیدا می‌کنیم. برای هر عنصر i با این ویژگی، عنصر j را که در محل این عنصر در فرزند کپی شده است، مشخص می‌کنیم. عنصر i را در فرزند، در محل عنصر j در والد اول کپی می‌کنیم. اگر مکان عنصر j در والد اول، در حال حاضر در فرزند با عنصر k پر شده است، عنصر i را در محل این عنصر در والد اول می‌کنیم. بعد از اتمام کپی عناصر بخش انتخاب شده از والد دوم، عناصر باقی مانده از والد دوم کپی می‌شوند. فرزند دوم نیز به همین ترتیب تولید می‌شود.

بر اساس والدهای در نظر گرفته شده در بخش قبل، مثال زیر، نحوه انجام بازترکیبی PMX را نمایش می‌دهد:

¹ partially mapped cross-over

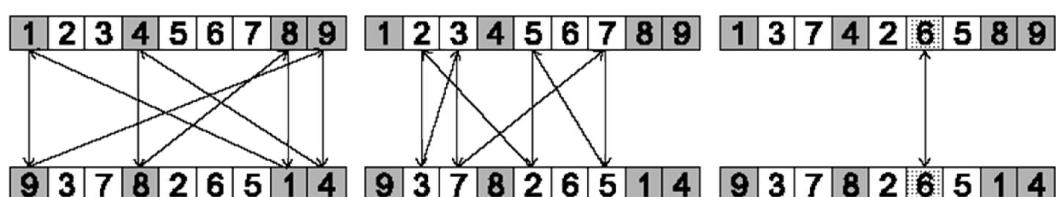


۳ - بازترکیبی حلقه^۱

این بازترکیبی مشابه بازترکیبی چند نقطه‌ای است با این حال جایگشت را نقض نمی‌کند. در این شیوه اگر فواصل زیاد باشند، ترتیب حفظ می‌شود و اگر کم باشند، همسایگی.

ایده اصلی این روش این است که هر آلل به همراه مکان خود، از یکی از والدین گرفته می‌شود. شیوه کار به این ترتیب است که حلقه‌ای از آلل‌های والد اول ایجاد می‌کنیم. از اولین آلل در والد اول (i) آغاز می‌کنیم. آلل موجود در مکان مشابه در والد دوم (j) را مشخص می‌کنیم. به مکان آلل j در والد اول می‌رویم. این آلل را به حلقه اضافه می‌کنیم. مراحل فوق را تا رسیدن به اولین آلل تکرار می‌کنیم. سپس آلل‌های حلقه ایجاد شده را در مکان‌های مشابه در فرزند اول کپی می‌کنیم.

به طریق مشابه، برای تولید فرزند دوم، حلقه بعدی را از والد دوم انتخاب می‌کنیم. مثال این روش را در زیر مشاهده می‌کنید:



¹ Cycle

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

1	3	7	4	2	6	5	8	9
---	---	---	---	---	---	---	---	---



9	3	7	8	2	6	5	1	4
---	---	---	---	---	---	---	---	---

9	2	3	8	5	6	7	1	4
---	---	---	---	---	---	---	---	---

۴- بازترکیبی لبه^۱

این بازترکیبی قادر به حفظ همسایگی‌های است. ابتدا جدولی تشکیل داده و همسایگی‌های هر عنصر را به صورت حلقوی در آن می‌نویسیم. به عنوان مثال برای دو والد گفته شده در مثال‌های پیشین، جدول به صورت زیر است:

Element	Edges	Element	Edges
1	2,5,4,9	6	2,5+,7
2	1,3,6,8	7	3,6,8+
3	2,4,7,9	8	2,7+, 9
4	1,3,5,9	9	1,3,4,8
5	1,4,6+		

عناصری که با علامت + مشخص شده‌اند، عناصری هستند که در هر دو والد با عنصر مورد نظر همسایه باشند. پس از تعیین همسایگی‌ها و پر کردن جدول فوق، با چند قانون فرزندان را تولید می‌کنیم. ژن اول را کاملاً تصادفی انتخاب می‌کنیم. انتخاب ژن بعدی را با اولویت دادن به همسایگی‌های + انجام می‌دهیم. در صورت عدم وجود چنین همسایگی‌ای، از عنصری که همسایگی‌های کمتری داشته باشد استفاده خواهیم کرد و اگر هیچ یک از این موارد وجود نداشت، ژن را به صورت تصادفی انتخاب می‌کنیم.

مثالی از آنچه گفته شد برای دو والد بالا به صورت زیر است:

Choices	Element selected	Reason	Partial result
All	1	Random	[1]
2,5,4,9	5	Shortest list	[1 5]
4,6	6	Common edge	[1 5 6]
2,7	2	Random choice (both have two items in list)	[1 5 6 2]
3,8	8	Shortest list	[1 5 6 2 8]
7,9	7	Common edge	[1 5 6 2 8 7]
3	3	Only item in list	[1 5 6 2 8 7 3]
4,9	9	Random choice	[1 5 6 2 8 7 3 9]
4	4	Last element	[1 5 6 2 8 7 3 9 4]

۵- بازترکیبی لبه توسعه یافته^۲

¹ Edge

² Modified Edge Recombination

نکته اصلی در این روش، حفظ همسایگی‌های مشترک است. در والد اول و دوم همسایگی‌های یکسان را حفظ می‌کنیم. عناصری را که در هر دو همسایگی مشترک است می‌نویسیم و عناصری را که باقی می‌مانند، با ترتیب والد دیگر و یا به صورت کاملاً تصادفی، جایگزین می‌کنیم. این که از کجا شروع می‌کنیم، می‌تواند تصادفی باشد. مثالی از این روش را در شکل زیر مشاهده می‌کنید:

Parent 1	Common links	Offspring 1
(6 7 8 1 2 3 9 4 5 0)	(- 7 8 1 ----- 5 0)	(3 7 8 1 4 6 2 9 5 0)
Parent 2	Common links	Offspring 2
(1 8 7 9 5 0 2 6 3 4)	(1 8 7 - 5 0 -----)	(1 8 7 3 5 0 6 2 4 9)

۶- بازترکیبی با استفاده از چند والد

گرچه در طبیعت و بازترکیبی‌هایی که تا به حال بررسی شد، موجودات از دو والد تولید می‌شوند، اما در عمل الگوریتم محدودیت دو والد را ندارد. روش‌های بازترکیبی متعارف از دو والد استفاده می‌کنند اما امکان استفاده از بیش از دو والد نیز وجود دارد. بازترکیبی‌های چند والدی مشتمل بر سه دسته اصلی هستند:

- مبتنی بر تکرار ژن‌ها
p-sexual voting
- مبتنی بر بخش بندی و ترکیب والدین
روش بازترکیبی قطری
- مبتنی بر عملگرهای عددی در مورد بازنمایی حقیق
روش مرکز ثقل

به عنوان مثال در بازترکیبی قطری، اگر سه والد داشته باشیم و هر یک را به سه قسمت مساوی تقسیم کنیم، فرزند حاصله نیز هر یک از سه قسمت خود را از یکی از والدها گرفته است.

۱-۱۷-۳- عملگرهای جهش

پیش از این نیز گفتیم که جهش در طبیعت خطاست و به ندرت صورت می‌گیرد. به این ترتیب جهش در واقع معادل جستجوی محلی است، مگر در مواردی که در سطح بسیار وسیع انجام گیرد. عملگرهای جهش بر روی یک فرد اعمال شده و فرد جدیدی ایجاد می‌کنند. عامل تصادف در جهش نقش مهم و تعیین کننده دارد و وجه تمایز آن از سایر عملگرهای هیورستیک به شمار می‌آید.

در ادامه خواهیم دید که اهمیت و کاربرد جهش به روش ارائه و الگوریتم تکاملی بستگی دارد. در GA جهش عملگری برای ایجاد و حفظ تنوع است. در EP تنها یک عملگر جستجو است و در GP به ندرت مورد استفاده قرار می‌گیرد.

۱-۳-۶-۱۷- جهش بیتی

برای جهش بیتی^۱، با یک احتمال کوچک عددی تصادفی مایین ۰ و ۱ ایجاد می‌کنیم. اگر عدد مذکور از P_m کوچک‌تر شود، بیت را معکوس می‌کنیم و در غیر این صورت بدون تغییر باقی می‌گذاریم. P_m نیز به صورت زیر تعیین می‌شود:

$$\frac{1}{N} < P_m < \frac{1}{\mu}$$

که N تعداد ژن (بیت)‌های کروموزم و μ تعداد افراد جمعیت هستند.

نمونه‌ای از آن چه گفته شد را در ذیل می‌بینید:

parent

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

child

0	1	0	0	1	0	1	1	0	0	0	1	0	1	1	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

با استدلال مشابه آن چه برای بازترکیبی گفته شد، ممکن است جهش در بازنمایی جایگشت منجر به پاسخ‌های غیر قابل قبول گردد. برای اجتناب از این امر لازم است در هر جهش، دست کم دو بیت تغییر کنند.

۲-۳-۶-۱۷- درج

عملگر درج^۲ تا حدود زیادی اطلاعات ترتیب و همسایگی را حفظ می‌کند. روش عمل آن به این گونه است که دو مقدار ژن را به تصادف انتخاب کرده و دومی را به اولی منتقل می‌کند و مابقی مقادیر را شیفت می‌دهد. نمونه‌ای از درج را در مثال زیر مشاهده می‌کنید:

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

 →

1	2	5	3	4	6	7	8	9
---	---	---	---	---	---	---	---	---

۳-۳-۶-۱۷- تعویض

در این حالت دو ژن به تصادف انتخاب شده و مکان آن‌ها تعویض می‌شوند. در این روش بیش‌تر اطلاعات ترتیب از دست می‌رود و همسایگی بیش‌تر حفظ می‌شود. نمونه‌ای از این عملگر را در مثال زیر مشاهده می‌کنید:

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

 →

1	5	3	4	2	6	7	8	9
---	---	---	---	---	---	---	---	---

¹ bit-wise

² Insert

۱-۴-۳-۶-۷-۸-۹ عکس کردن

دو ژن به تصادف انتخاب شده و زیر رشته‌ای که میان آن‌ها وجود دارد عکس می‌شود. در این روش بیشتر اطلاعات مربوط به همسایگی حفظ می‌شود و اطلاعات مربوط به ترتیب بیشتر از دست می‌روند. نمونه‌ای از این عملگر را در مثال زیر مشاهده می‌کنید:

1	2	3	4	5	6	7	8	9
→								

1	5	4	3	2	6	7	8	9
→								

۱-۷-۶-۳-۵-۴-۲-۱ تقلیل^۱

در این روش زیر مجموعه‌ای از ژن‌ها انتخاب می‌شوند. مقادیر موجود در این ژن‌ها به تصادف جابجا می‌شوند. توجه داشته باشید که نیازی به پیوسته بودن زیرمجموعه انتخابی نیست. مثال زیر نمونه‌ای از نحوه عمل این عملگر را نمایش می‌دهد:

1	2	3	4	5	6	7	8	9
→								

1	3	5	4	2	6	7	8	9
→								

¹ scramble

فصل پنجم: الگوریتم‌های پایه

۱۸-۱ مقدمه

در این فصل //////////////

در ادامه چهار الگوریتم مشهور را بررسی خواهیم کرد.

۱۹-۱ الگوریتم ژنتیک

اولین الگوریتم پیشنهاد شده، الگوریتم ژنتیک^۱ (GA) نام دارد که در ۱۹۷۰ توسط هولند^۲ و گلدبگ^۳ در امریکا معرفی شد. کاربری اصلی این روش در بهینه سازی گستته است. برنامه نویسی ژنتیک معمولاً دارای ساختار گرافی و درختی بوده و ویژگی اصلی آن در یافتن مدل بهینه بجای یافتن پاسخ بهینه برای مسأله است. البته روش‌های یادگیری دیگر مانند شبکه‌های عصبی نیز برای پاسخ بهینه، سعی در پیدا کردن مدل می‌کنند، اما مزیت برنامه نویسی ژنتیک نسبت به چنین روش‌هایی، معنادار بودن مدل به دست آمده و در نتیجه قابلیت تحلیل آن است. این روش قادر است مسائلی با پارامترهای ترکیبی (باينری، بولین و ...) را حل کند. با این حال، ضعف عمدۀ آن سرعت پایین است. سرعت پایین سبب می‌شود GA در حالت پیوسته خیلی کند باشد. تکیه GA روی بازنگری است که با احتمالی بالاتر از ۸۰٪ رخ می‌دهد و در مقابل آن، جهش اهمیت به مراتب پایین‌تری داشته و احتمال بسیار کمی دارد.

شناسنامه GA

باينری	بازنمایی
تصادفی یکنواخت-تعداد μ	جمعیت اولیه
$B \rightarrow x \rightarrow f(x)$	ارزیابی
متناوب با شاستگی با روش‌های RW و SUS	انتخاب والدین ($\lambda = \mu$)
چند نقطه‌ای	بازترکیبی
۱/n و $1/\mu$ بین p_m با عکس شدن بیت	جهش

¹ Genetic Algorithm

² Holland

³ Goldberg

نadarad ($\mu, \bar{\mu}$)	انتخاب بازماندگان
تعداد ارزیابی	شرط خاتمه
تأکید بر روی بازترکیبی	ویژگی

فرض کنید مسأله تک تابع حقیقی مانند $f(x) = x^{10} + x^5 - 3x + 10$ باشد و هدف به دست آوردن کمینه آن در بازه $a < x < b$ باشد:

0	1	0	0	1	1	0	1
---	---	---	---	---	---	---	---

تنها در محاسبه شایستگی است که لازم است عدد باینری به حقیقی تبدیل شود. در اینجا ۸ بیت (در حالت کلی n بیت) داریم که بازه a تا b را پوشش می‌دهند. با استفاده از تابع خطی زیر، تبدیل را انجام می‌دهیم:

شکل

$$B = b_0 2^0 + b_1 2^1 + \dots + b_{n-1} 2^{n-1} = \sum_{i=0}^{n-1} b_i 2^i$$

$$x = a + \frac{b-a}{2^n - 1} \cdot B$$

در این روش، ابتدا B و سپس x را از دو رابطه فوق خواهیم داشت و نهایتاً $f(x)$ را به دست می‌آوریم. توجه داشته باشید که با این روش نمی‌توان تمامی اعداد حقیقی را داشت. فاصله اعداد، یا به تعبیر دقیق‌تر، دقت، برابر $\frac{b-a}{2^n - 1}$ است.

دقت مورد نیاز در مسائل مختلف، متفاوت خواهد بود. برای مثال در مسأله مرتبط با فرودگاه، دقتی معدل چند متر یا کیلومتر مورد نیاز است. با در دست داشتن دقت مورد نیاز (A)، می‌توان تعداد بیت‌ها را به صورت تعیین کرد:

$$\frac{b-a}{2^n - 1} = A$$

$$\frac{b-a}{A} + 1 = 2^n$$

$$n = \log_2\left(\frac{b-a}{A} + 1\right)$$

بدیهی است که هر چه دقت مورد نظر بیشتر باشد، به تعداد بیت‌ها افزایش یافته و الگوریتم ژنیک کد خواهد شد. چند متغیر داشته باشیم و به دقت بالایی نیاز باشد، تعداد بیت‌ها افزایش یافته و الگوریتم ژنیک کد خواهد شد.

در شناسنامه GA دیدیم که در این الگوریتم، تعداد فرزندان با تعداد والدین برابر است. همچنین بازترکیبی چند نقطه‌ای است و با احتمال $0.8 = p_c$ است. برای بازترکیبی، عددی تصادفی بین 0 و 1 انتخاب می‌شود. اگر این عدد از p_c کوچک‌تر باشد، بازترکیبی انجام می‌شود و در غیر این صورت، خیر.

به عنوان مثال جمعیت جاری متشکل از $4 = \mu$ موجود با شایستگی $10, 20, 50$ و 80 را در نظر بگیرید: سطر اول جمعیت جاری را نمایش می‌دهد. شایستگی هر موجود بر روی آن نوشته شده است. در فرآیند بازترکیبی

برای هر بیت یک عدد تصادفی مابین 0 و 1 تولید می‌شود. فرض کنید دو عدد تصادفی تولیدی بیش از p_c و دو عدد دیگر، کمتر از آن هستند. به این ترتیب برای دو موجود اول بازترکیسی انجام خواهد شد ($3', 1'$) و دو موجود دیگر، خودشان به مرحله بعد خواهند آمد. در مرحله بعد و برای تولید فرزندان (سطر چهارم) جهش را در نظر می‌گیریم. مجدداً برای هر بیت عددی تصادفی بین 0 و 1 تولید می‌کنیم. اگر این عدد از $p_m = 0.5$ کوچک‌تر باشد، بیت مورد نظر معکوس شده و در غیر این صورت، خیر. فرض کنید موجودات اول و آخر دستخوش جهش شوند. به این ترتیب سطر چهارم که فرزندان (λ) را تشکیل می‌دهد به دست می‌آید. توجه داشته باشید که دو مرحله میانی دور ریخته می‌شوند.

1	2	3	4
80	20	50	10

3	1	1	2

3'	1'	1	2

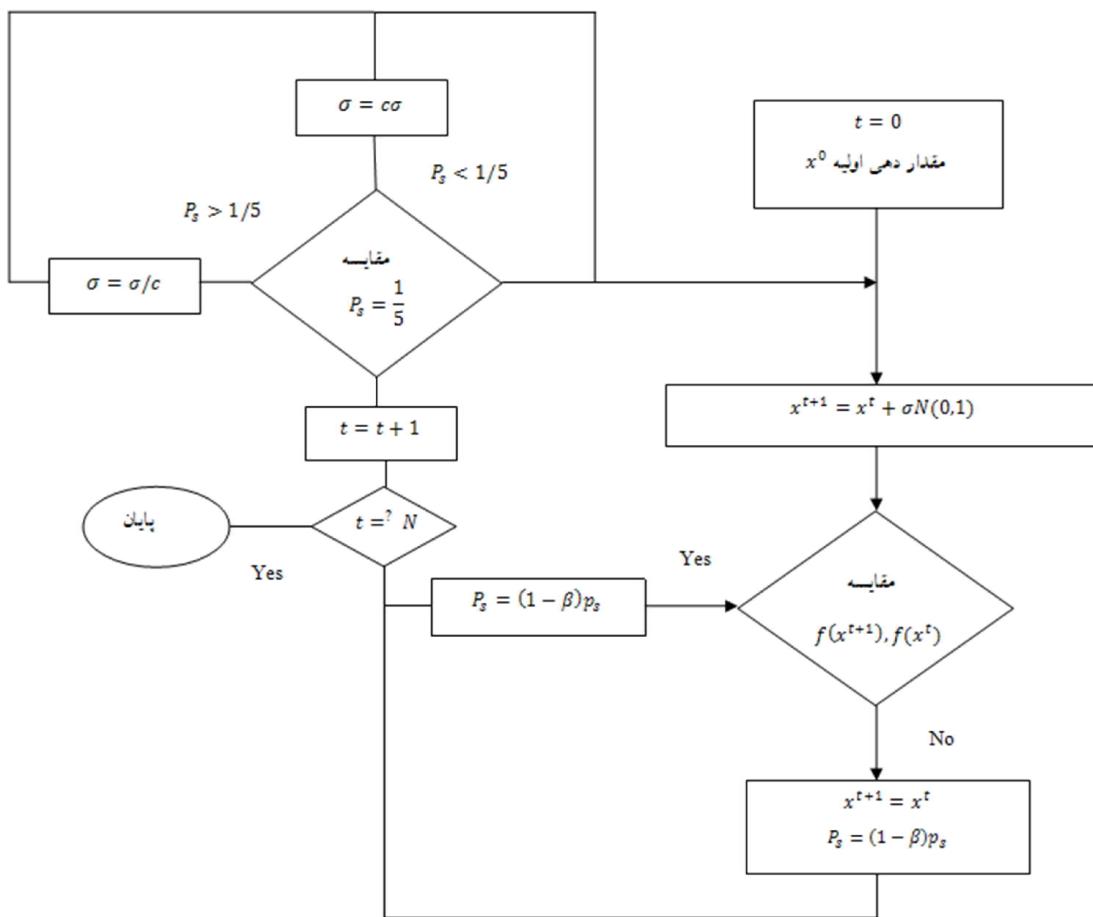
3''	1'	1	2'

عیب این روش، انتخاب متناسب با شایستگی است. به انتخاب بازماندگان با این روش، نسلی گفته می‌شود. که سبب همگرایی زودرس و ساکن شدن جمعیت می‌گردد. اگر مسئله تنها یک جواب داشته باشد، این روش برای آن خوب است. که البته در چنین شرایطی، تقریباً هر روشی خوب است!

۱-۲۰ استراتژی‌های تکامل

روش بعدی که به آن خواهیم پرداخت، استراتژی‌های تکامل^۱ (ES)، نام دارد. در این الگوریتم به منظور بالا بردن سرعت همگرایی از استراتژی‌هایی استفاده می‌شود. به این ترتیب که دور از جواب گام‌های بزرگ و به تعبیری سرعت بالا دارد و در نزدیکی پاسخ، که جستجو مستلزم دقت بالاتر است، گام‌های کوچک و سرعت پایین. اصطلاحاً گفته می‌شود که اندازه قدم تطبیقی دارد. در این روش، جستجوی عمومی با گام‌های بزرگ و جستجوی محلی با گام‌های کوچک انجام می‌گیرد.

¹ Evoloutaionary Strategies



ES فلوچارتی مشابه تپه نورده دارد. اولین ES ارائه شد، ۱ + ۱ بود. در سال ۱۹۷۰ Rechenberg و Schwefel آلمانی قانون $\frac{1}{5}$ موفقیت را برای ES ۱ + ۱ ثابت و منتشر کردند. بر اساس این قانون، اگر احتمال موفقیت در فلوچارت ۱ + ۱ برابر $1/5$ باشد، سرعت رسیدن به کمینه محلی، حداقل خواهد بود. لازم است برای موفقیت یک احتمال (P_s) تعریف کنیم. این احتمال در واقع متعلق به k تکرار قبل است. در فلوچارت، خط واصل $P_s = (1 - \beta)p_s$ و مقایسه f ها، بیان گر موفقیت است. اگر از k تکرار، $1/5$ آنها موفق بوده باشد، موفق است. به این ترتیب یا می‌بایست k تکرار قبل را در حافظه نگه داریم و یا از رابطه زیر استفاده کنیم:

اگر مقادیر متناظر با k تکرار قبل را x_1 تا x_k بنامیم، متوسط آنها برابر

$$\bar{x} = \frac{x_1 + \dots + x_k}{k}$$

است. در گام بعد و برای x_2 تا x_{k+1} متوسط جدید برابر است با

$$\bar{x}_{new} = \frac{k\bar{x} - x_1 + x_{k+1}}{k}$$

برای حذف x_1 آن را تقریب متوسط جایگزین می‌کنیم:

$$\frac{k-1}{k} \bar{x} + \frac{x_{k+1}}{k} = \left(1 - \frac{1}{k}\right) \bar{x} + \frac{x_{k+1}}{k}$$

رابطه متوسط گیری پنجره‌ای

اگر در رابطه بالا $1/k$ را برابر β در نظر بگیریم خواهیم داشت:

$$\bar{x}^{new} = (1 - \beta)\bar{x}^{old} + \beta x_{new}$$

که k تعداد نمونه‌های درون پنجره است. به عبارتی:

$$P_s = (1 - \beta)P_s + \beta P_s$$

اگر $P_s = 1/5$ باشد، اندازه قدم مناسب را داریم و آن را تعییر نمی‌دهیم. $1/5 < P_s < 1$ به معنای بزرگ بودن اندازه قدم است که می‌بایست اصلاح شده و کوچک شود. به این ترتیب σ را در $1 < c < 0$ ضرب می‌کنیم. به طور مشابه، $P_s > 1/5$ به معنای کوچک بودن اندازه گام‌هاست که برای اصلاح، σ را بر c فوق تقسیم می‌کنیم.

شکل

ES بر خلاف GA که بیشتر در حالت گشته مورد استفاده است، برای بهینه سازی پیوسته و اعداد حقیقی کاربرد دارد و نیز در مواردی که با تعداد زیادی از پارامترها سر و کار داریم. ویژگی ES یرعت بالای آن است. همچنین به خاطر جهش با توزیع نرمال، کارهای ریاضی قوی‌ای روی تئوری آن انجام شده است.

ویژگی دیگر ES اندازه قدم تطبیقی آن (مانند $1/5$ موقیت گفته شده در بالا) و یا خود تطبیقی است. در هر دو حالت اندازه قدم تطبیقی یا خود تطبیقی، ES اندازه قدم‌ها را به صورت اتوماتیک تنظیم می‌کند. دو نمونه از متعارف‌ترین روش‌ها عبارتند از:

- روشن: Steepest Decsent

$$x^{new} = x^{old} \pm \eta \nabla F(x^{old})$$

- روشن نیوتون

$$x^{new} = x^{old} \pm H^{-1}(x^{old}) \nabla F(x^{old})$$

دقت داشته باشید که این روش در کمینه محلی متوقف می‌شود. با این حال، با حداکثر سرعت به آن خواهد رسید. البته نسخه‌های دیگر ES قادرند از کمینه‌های محلی بگریزند.

به علاوه، باید توجه داشت که قانون $1/5$ موقیت تنها در $1 + Es$ مورد استفاده است و در سایر موارد موضوعیت نخواهد داشت.

شناسنامه الگوریتم ES

اعداد حقیقی	بازنمایی
-------------	----------

تصادفی یکنواخت (μ)	جمعیت اولیه
تصادفی یکنواخت (λ)	انتخاب والدین
گسسته و میانگین گیری	بازترکیبی
نویز گوسی	جهش
μ, λ $\mu + \lambda$	انتخاب بازماندگان
تعداد ارزیابی	شرط خاتمه
تطبیق با خود تطبیقی بودن اندازه قدمها و جهش	ویژگی خاص

در ES نیز تعداد فرزندان و والدین برابر هم و برابر با λ است.

انتخاب بازماندگان به دو شیوه انجام می‌گیرد. اگر از λ, μ استفاده شود، انتخاب با استفاده از Q-tournament توصیه می‌شود و اگر از $\lambda + \mu$ استفاده شود، انتخاب متناسب با شایستگی. پیش‌تر در مورد Q-tournament گفتیم که تنوع بالایی دارد و به علاوه قابلیت گریز از کمینه محلی را دارد. همان طور که اشاره شد، در انواعی دیگر ES غیر از $1 + 1$ مانند $ES \mu, \lambda$ یا $ES \mu + \lambda$ ، دیگر قادر به استفاده از قانون $1/5$ موفقیت نخواهیم بود. در این قانون، که در ادامه آن را اثبات خواهیم کرد، یک تابع سرعت نوشتہ و آن را بیشینه می‌کنیم. به این منظور، مشتق نسبت به σ (اندازه قدم) را برابر صفر قرار می‌دهیم.

خود تطبیقی

برای استفاده از یک روش تطبیقی، نیازمند داشتن فیدبک‌هایی از محیط هستیم. به علاوه، آستانه‌ای را تعیین کرده و بر اساس آن اندازه قدم را تعیین می‌کنیم. با این حال، پیدا کردن فیدبک مناسب و آستانه‌ای ثابت بر اساس آن کار ساده‌ای نیست. به این ترتیب، روش‌های خود تطبیقی مطرح می‌شوند. در این روش‌ها، برای یافتن مقدار بهینه x از تکامل استفاده می‌شود. همچنین اندازه قدم بهینه نیز با استفاده از تکامل به دست می‌آید و به این دلیل خود تطبیقی نامیده می‌شود.

ژن‌های روی کروموزوم، x_1 تا x_n پارامترهای مسئله هستند. σ را هم به صورت زیر کنار آن‌ها قرار می‌دهیم:

x_1	x_2	...	x_n	σ
-------	-------	-----	-------	----------

معمولًاً بر روی σ بازترکیبی انجام نمی‌شود و فقط به صورت زیر جهش پیدا می‌کند:

$$\sigma^{new} = \sigma^{old} \cdot e^{-\tau N(0,1)}$$

که $N(0, 1)$ نویز نرمال و τ ضریب یادگیری است.

جهش برای σ به صورت نمایی فوق و برای x به صورت خطی زیر است:

$$x^{new} = x^{old} + \sigma^{new} N(0, 1)$$

از آن جا که اندازه قدم‌ها نباید تغییرات شدیدی داشته باشد، دو مقدار σ^{old} و σ^{new} به یکدیگر نزدیک هستند. به علاوه، بدیهی است که اندازه قدم نمی‌تواند منفی باشد. به این ترتیب $e^{-\tau N(0, 1)}$ مقداری حدود ۱ خواهد داشت. در روابط فوق، رابطه اول اندازه قدم را مشخص می‌کند و رابطه دوم برداشتن گام را.

نکته‌ای که می‌بایست مورد توجه قرار گیرد، این است که ابتدا σ را جهش می‌دهیم و بعد از آن x را. چرا که ارزیابی σ بر اساس توانایی آن در تولید x مناسب صورت می‌گیرد. بنابراین، اگر x^{new} به دست آمده از رابطه بالا خوب باشد، می‌توان نتیجه گرفت که σ^{new} آن نیز خوب بوده است. در حالت عکس، یعنی اگر ابتدا x و سپس σ را جهش دهیم، ممکن است x^{new} را بر اساس خوب بودن x نگه داشته و به نسل بعد ببریم، در حالی که σ آن بد بوده است. در واقع در این حالت، σ قبلی خوب بوده است که سبب شده x خوب باشد. در حالی که ما σ بد را به نسل بعد فرستاده‌ایم.

البته باید توجه داشت حتی در شرایطی که ابتدا σ را جهش دهیم، ممکن است σ بد باشد و نویز خوب که x^{new} به این دلیل خوب شده باشد. با این حال این شیوه بهتر و مطمئن‌تر از این است که ابتدا x را جهش دهیم. بر اساس آن چه گفته شد توصیه می‌شود از λ, μ استفاده کنیم که قادر است ویژگی‌های بد را فراموش کند.

شكل

اگر متغیر تصادفی Z با توزیع نرمال، میانگین صفر و واریانس σ را به صورت شکل بالا داشته باشیم، σ با احتمال ۹۰٪ بین $2 \pm$ ، حدود ۶۰٪ بین $1 \pm$ و با احتمال ۹۹٪ بین $3 \pm$ است.

نرخ یادگیری (τ)

σ نمی‌تواند از مقدار آستانه‌ای (ϵ_0) کوچک‌تر شود، در غیر این صورت با صفر شدن σ الگوریتم متوقف می‌شود.

$$\tau \propto \frac{1}{n^{\frac{1}{2}}}$$

که n تعداد بعد است. در این حالت σ واحدی برای تمامی ابعاد در نظر گرفته‌ایم، حال آن که برای هر بعد می‌بایست گام متفاوتی داشته باشیم. چون فواصل و به تبع آن، تعداد گام‌های لازم در ابعاد مختلف یکسان نیست. بر این اساس حالت زیر را خواهیم داشت:

x_1	\dots	x_n	σ_1	\dots	σ_n
-------	---------	-------	------------	---------	------------

بدیهی است که در این حالت n پارامتر داریم و به طرز آشکاری فضای جستجو گسترده‌تر از حالت قبل با $1 +$

پارامتر است، واقعیت این است که گستردگی شدن فضای جستجو سبب می‌گردد در اکثر کاربردها، استفاده از یک σ زودتر به جواب برسد. به ویژه در شرایطی که نزدیک جواب باشیم این حالت به مراتب زودتر به نتیجه می‌رسد. با این حال، در شرایطی که خیلی از پاسخ فاصله داشته باشیم، استفاده از فضای جستجوی گستردگی‌تر سرعت بیشتری خواهد داشت.

تا به اینجا σ متناظر با ابعاد مختلف را در نظر گرفتیم. برای افزایش سرعت می‌توانیم زوایای چرخش را نیز به مجموعه فوق اضافه کنیم. به این ترتیب خواهیم داشت:

x_1	...	x_n	σ_1	...	σ_n	α_1	...	α_m
-------	-----	-------	------------	-----	------------	------------	-----	------------

شکل زیر سه حالت گفته شده در بالا را نمایش می‌دهد. محل فعلی نقطه میانی دایره و بیضی‌هاست و هدف نقطه بیشینه محلی مشخص شده است. بدیهی است که در حالت دوم بیضی جستجو می‌تواند افقی یا عمودی باشد، اما نه مورب. مقادیر σ جهت و کشیدگی بیضی مربوطه را مشخص می‌کند. به عنوان مثال در راستایی که فاصله بیشتری تا هدف داریم قطر بیضی بزرگ‌تر و در راستایی که فاصله تا هدف کمتر است، قطر بیضی کوچک‌تر خواهد بود.

شکل

چگونگی جعش برای به دست آوردن σ و α های فوق

$$\begin{aligned}\sigma_i^{new} &= \sigma_i \cdot e^{\tau^{new} \cdot N(0,1) + \tau N_i(0,1)} \\ x_i^{new} &= x_i + \sigma_i^{new} \cdot N_i(0,1)\end{aligned}$$

در رابطه اول با دو نرخ یادگیری مواجه هستیم؛ یک نرخ یادگیری سراسری و یک نرخ مختص بعد داریم. $N(0,1)$ برای تمامی σ ها یکسان است و اندازه قدم در حالت کلی به صورت برداری است. در مقابل، $N_i(0,1)$ برای σ های مختلف متفاوت است.

۱-۵-۲-۱ جعش نوع سوم

با استفاده از σ و α ها یک ماتریس کوواریانس (C) تولید می‌شود که از روی آن نویز همبسته‌ای ایجاد می‌گردد که با x جمع شده و ما را به سمت جواب می‌برد. به تعبیر دیگر، بیضی‌ها را در آن سمت می‌چرخاند.

$$\begin{aligned}\sigma\alpha &\rightarrow C \\ x^{new} &= x^{old} + N(0, C)\end{aligned}$$

ایده این روش، مشابه چیزی است که در MATLAB انجام می‌شود؛ به این ترتیب که ماتریس C را به دو ماتریس پایین و بالا متشی می‌شکنیم:

$$C = AA^T$$

به این ترتیب خواهیم داشت:

$$N(0, C) = AN(0, I)$$

$$Y = AN(0, I)$$

اگر $X \sim N(0, I)$ باشد، X را بنامیم یعنی $X \sim N(0, I)$ خواهیم داشت:

$$E[YY^T] = \sum Y = E[AXX^TA] = AE[XX^T]A^T$$

با توجه به این که $XX^T = I$ است، عبارت فوق برابر خواهد بود با:

$$AA^T = C$$

روشن است که اگر ماتریس C نیمه معین^۱ نباشد، نمی‌توان آن را به فرم بالا نوشت.

به جای به دست آوردن σ و α ها، می‌توان مستقیماً a های ماتریس زیر را با استفاده از تکامل به دست آورد:

$$A = \begin{bmatrix} a_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$$

مقادیر a می‌توانند منفی نیز باشند. بنابراین باید به صورت خطی جهش داده شوند چون ماتریس کوواریانس می‌تواند المان‌های منفی داشته باشد:

$$A^{new} = A^{old} + \alpha N(0, I)$$

به این ترتیب، از روی A و نویز سفید، نویز همبسته را به دست می‌آوریم:

$$x^{new} = x^{old} + A^{new}N(0, I)$$

روش دیگر، این است که نویز سفید را در ماتریس‌های چرخش ضرب کنیم. این روش، با توجه به هزینه محاسباتی بالا روش خوبی به شمار نمی‌آید.

به عنوان نمونه برای α_{12} ، α_{13} و α_{23} ماتریس‌های چرخش زیر را داریم:

$$\alpha_{12}: \begin{bmatrix} \cos \alpha_{12} & \sin \alpha_{12} & 0 \\ -\sin \alpha_{12} & \cos \alpha_{12} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\alpha_{13}: \begin{bmatrix} \cos \alpha_{13} & 0 & \sin \alpha_{13} \\ 0 & 0 & 0 \\ -\sin \alpha_{13} & 0 & \cos \alpha_{13} \end{bmatrix}$$

$$\alpha_{23}: \begin{bmatrix} 0 & 0 & 0 \\ 0 & \cos \alpha_{23} & \sin \alpha_{23} \\ 0 & \sin \alpha_{23} & \cos \alpha_{23} \end{bmatrix}$$

یک کاربرد عملی

اولین استفاده عملی از $ES1 + 1$ تولید یک نازل جت در ناسا بود. نازل جایی است که اکسیژن و سوخت در آن

^۱ Semi-Positive

مخلوط و فشرده می‌شوند. نحوه ترکیب باید به گونه‌ای باشد که مخلوط به دست آمده در مخزن احتراق کامل بسوزد. شکل پیچیده زیر حاصل الگوریتم ES است. بدینهی است که با الگوریتم‌های تحلیلی نمی‌شد به چنین شکلی دست یافت.

شکل

۱۲۱-۱ اثبات قانون ۱/۵ موققت

بدون این که به عام بودن اثبات خللی وارد شود، در حالت دو بعدی صحبت خواهیم کرد. همچنین برای سادگی کمینه را در نقطه صفر در نظر می‌گیریم. بدینهی است که تنها با یک تغییر متغیر می‌توان هر پاسخی را به صفر برد. فرض می‌کنیم در نقطه X هستیم. سرعت همگرایی بیشتر به این معنی است که در هر گام بیشتر به نقطه صفر نزدیک شویم.

شکل

توجه داشته باشید که X نمایش‌گر متغیر تصادفی و x بیان کننده متغیر قطعی است.

$$\begin{aligned}f(x) &= \sum x_i^2 \\r_t^2 &= \sum x_i^2 = f(x^t) \\r_{t+1}^2 &= f(X^{t+1})\end{aligned}$$

از دو رابطه فوق داریم:

$$\Delta f = f(x^t) - f(X^{t+1})$$

Δf تعریف شده به صورت بالا را سرعت همگرایی می‌نامیم.

اگر سرعت همگرایی را بیشینه کنیم، در هر گام سرعت را حداکثر کرده‌ایم. البته یک راه هم این است که سرعت را در کل حداکثر کنیم. زمانی که سرعت را در هر گام حداکثر می‌کنیم، در کل هم حداکثر می‌شود. در حالتی که گام‌ها قطعی باشند، دو روش گفته شده برابر هستند. اما در اینجا گام‌ها متغیرهای تصادفی هستند و ما در حقیقت امید ریاضی آن‌ها را حداکثر می‌کنیم. در این حالت، حداکثر کردن امید ریاضی هر گام، با حداکثر کردن امید ریاضی کل متفاوت است. در این حالت، می‌توانیم یک گام یا مجموعه‌ای از تعداد بیشتری از گام‌ها را در نظر گرفته و حداکثر کنیم. هر چه این تعداد بیشتر باشد بهتر است، با این حال، همان طور که گفته شد در حالتی که با مقادیر قطعی سر و کار نداریم، آن چه به دست می‌آید با نتیجه بیشینه کردن کل متفاوت خواهد بود.

$$X^{t+1} = x^t + \sigma Z$$

که در عبارت بالا Z متغیری تصادفی با توزیع نرمال است ($Z \sim N(0, 1)$).

$$f(x^t) = \sum (x_i^t)^2$$

$$Z = \begin{bmatrix} Z_1 \\ \vdots \\ Z_n \end{bmatrix}, \quad x^t = \begin{bmatrix} x_1^t \\ \vdots \\ x_n^t \end{bmatrix}$$

$$\sum x_i^2 = r^2$$

$$\sum (x_i^t)^2 - \sum (x_i^t + \sigma Z_i)^2 = \sum (x_i^t)^2 - \sum (x_i^t)^2 - \sigma^2 \sum Z_i^2 - 2\sigma \sum x_i^t Z_i$$

به این ترتیب خواهیم داشت:

$$\Delta f = \sigma^2 \sum_{i=1}^n Z_i^2 - 2\sigma \sum_{i=1}^n x_i^2 Z_i$$

اگر متغیر تصادفی با توزیع گوسی نرمال زیر را داشته باشیم و x_i ها مستقل باشند خواهیم داشت:

$$X_i \sim N(\mu_i, \sigma_i^2)$$

بر اساس استقلال خواهیم داشت:

$$a + \sum_{i=1}^n b_i X_i \sim N(\mu, \sigma^2)$$

که:

$$\mu = a + \sum_{i=1}^n b_i \mu_i$$

$$\sigma^2 = \sum_{i=1}^n b_i \sigma_i^2$$

در واقع ترکیب خطی چنین متغیرهایی نیز گوسی و با ویژگی‌های بالاست.

به این ترتیب اگر در رابطه $\Delta f = b_i x_i^t$ باشد، بر اساس رابطه فوق خواهیم داشت:

$$\sum_{i=1}^n x_i^t Z_i = Y \sim N\left(0, \sum_{i=1}^n (x_i^t)^2\right) \sim N(0, r_t^2)$$

$$X_i \sim N(\mu_i, \sigma_i^2)$$

از رابطه فوق:

$$X_i = \mu_i + \sigma_i Z$$

که $Z \sim N(0, 1)$. به این ترتیب:

$$\Delta f = \sigma^2 \sum_{i=1}^n Z_i^2 - 2\sigma r_t Z$$

توجه کنید که آن چه انجام می‌شود، به این دلیل است که نمی‌توان از یک متغیر تصادفی مشتق گرفت اما از امید ریاضی آن می‌توان. برای به دست آوردن امید ریاضی باید توزیع زیر را داشته باشیم:

$$E[X] = \int X P_X(x) dx$$

ابتدا توزیع Δf را پیدا می‌کنیم:

اگر X_i ها دارای توزیع نرمال و مستقل از هم باشند:

$$X_i \sim N(0, 1) \xrightarrow{\text{independency}} \sum_{i=1}^n X_i^2 \sim Chis1(n)$$

که توزیع مربع خای است.

برای $n > 30$

$$\sim \frac{1}{2} Y^2, \quad Y \sim N(\sqrt{2n-1})$$

شرطی که لزوماً برقرار نیست، استقلال X_i ها از یکدیگر است. اگر مستقل نباشند در رابطه قبل:

$$\begin{cases} \mu = a + \sum_{i=1}^n b_i \mu_i \\ \sigma^2 = \sum_{i=1}^n \sum_{j=1}^n b_i b_j c_{ij}, \quad c_{ij} = cov(X_i, X_j) \end{cases}$$

فرض دیگر $n > 30$ است.

$$\Delta f = -\sigma^2 \left(\frac{1}{2} Y^{*2} \right) - 2\sigma r_t z$$

$$Y^* \sim N(\sqrt{2n-1}, 1)$$

در نتیجه:

$$Y^{*2} = (\sqrt{2n-1} + z)^2 = 2n - 1 + z^2 + 2\sqrt{2n-1} \cdot z$$

اگر n (تعداد ابعاد برای نمودار) بزرگ باشد خواهیم توانست از ترم‌های z^2 و 1 و $2\sqrt{2n-1} \cdot z$ صرف نظر کرد
یعنی:

$$Y^{*2} = 2n$$

در نتیجه:

$$\Delta f = n\sigma^2 - 2\sigma r_t z$$

البته واقعیت این است که بهتر است از z صرف نظر نکنیم.

با ضرب طرفین در $\frac{n}{r_t^2}$ ، تفاوتی در ماکریم ایجاد نمی‌شود. عبارت به دست آمده، سرعت همگرایی نامیده می‌شود.

$$\varphi = \frac{n\Delta f}{r_t^2} = -\frac{n^2\sigma^2}{r_t^2} - \frac{2\sigma r_t n}{r_t^2}$$

لذا خواهیم داشت:

$$\sigma^* = \frac{n\sigma}{r_t}$$

به این ترتیب:

$$\varphi = -\sigma^{*2} - 2\sigma^* z$$

اکنون تنها یک متغیر σ^* داریم که نسبت به آن مشتق می‌گیریم.

$$\varphi \sim N(-\sigma^{*2}, 4\sigma^{*2})$$

تنها بخش‌های مثبت φ را می‌خواهیم. بنابراین امید ریاضی آن‌ها را به دست می‌آوریم. دلیل این کار این است که در $ES(1 + 1)$ تنها در صورت بهتر بودن فرزند نسبت به والد آن را می‌پذیریم، لذا برای φ مقدار منفی نخواهیم داشت.

$$\begin{aligned} E[\varphi^+] &= \int_0^{+\infty} \varphi^+ P_{\varphi^+} d\varphi^+ \\ &= \int_0^{+\infty} \varphi^+ \cdot 1/(2\sigma^* \sqrt{2\pi}) \cdot e^{-\frac{1}{2}(\frac{\varphi^++\sigma^{*2}}{2\sigma^*})^2} d\varphi^+ \\ &= \int_{\frac{\sigma^2}{2}}^{+\infty} \frac{2\sigma^* u - \sigma^{*2}}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2} du \end{aligned}$$

اگر تغییر متغیر زیر را اعمال کنیم:

$$\frac{\varphi^+ + \sigma^{*2}}{2\sigma^*} = u$$

خواهیم داشت:

$$d\varphi^+ = 2\sigma^* du$$

و به طور معادل:

$$\frac{d\varphi^+}{2\sigma^*} = du$$

به این ترتیب:

$$\begin{aligned} \varphi^+ &= 2\sigma^* u - \sigma^{*2} \\ &= \int_{\frac{\sigma^2}{2}}^{+\infty} \frac{2\sigma^* u}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2} du - \sigma^{*2} \int_{\frac{\sigma^2}{2}}^{+\infty} \frac{e^{-\frac{1}{2}u^2}}{\sqrt{2\pi}} du \end{aligned}$$

شکل

$$\Phi(x) = \int_0^x \frac{e^{-\frac{1}{2}u^2}}{\sqrt{2\pi}} du$$

و با استناد به جدول، ترم دوم عبارت برابر است با:

$$-\sigma^{*2} \left(\frac{1}{2} - \Phi \left(\frac{\sigma^*}{2} \right) \right)$$

همچنین برای محاسبه ترم اول از تغییر متغیر زیر استفاده می‌کنیم:

$$-\frac{1}{2} u^2 = v$$

$$-udu = dv$$

به این ترتیب ترم اول برابر خواهد بود با:

$$\int_{-\infty}^{-\frac{\sigma^2}{8}} \frac{2\sigma^* e^v dv}{\sqrt{2\pi}} = \frac{2\sigma^*}{\sqrt{2\pi}} (e^{-\frac{\sigma^2}{8}} - 0)$$

لذا خواهیم داشت:

$$E[\varphi^+] = \frac{2\sigma^*}{\sqrt{2\pi}} e^{-\frac{\sigma^{*2}}{8}} - \sigma^{*2} \left(\frac{1}{2} - \Phi\left(\frac{\sigma^*}{2}\right) \right)$$

از عیارت فوق، نسبت به σ^* مشتق گرفته و برابر صفر قرار می‌دهیم.

با توجه به غیرخطی بودن عبارت حاصل، می‌بایست از روش‌های عددی برای حل مسأله استفاده کرد و به این ترتیب بهتر است در همینجا، به جای مشتق گرفتن، از رسم شکل استفاده کنیم:

شکل

به این ترتیب برای داشتن حداقل سرعت همگرایی لازم است σ^* برابر ۱,۲۲ باشد.

واقعیت این است که از پارامتر σ^* که در بالا اشاره شد نمی‌توان اشتفاده کرد، چرا که برای به دست آوردن آن نیازمند پارامترهای مجھولی چون r_t هستیم. بنابراین احتمال موفقیت را محاسبه می‌کنیم:

$$P(\varphi^+ > 0) = \int_0^{+\infty} \frac{1}{2^{*1.22}\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{\varphi^++(1.22)^2}{2^{*1.22}}\right)^2} d\varphi^+$$

در ادامه برای سادگی تغییر متغیرهای زیر را در نظر می‌گیریم:

$$u = \frac{\varphi^+ + a}{b}$$

که به این ترتیب:

$$d\varphi^+ = bdu$$

بر این اساس خواهیم داشت:

$$\begin{aligned} P(\varphi^+ > 0) &= \int_{\frac{a}{b}}^{+\infty} \frac{bu - a}{\sqrt{2\pi}} \cdot e^{-\frac{1}{2}u^2} du \\ &= \int_{\frac{a}{b}}^{+\infty} \frac{bu}{\sqrt{2\pi}} \cdot e^{-\frac{1}{2}u^2} du - a \int_{\frac{a}{b}}^{+\infty} \frac{e^{-\frac{1}{2}u^2}}{\sqrt{2\pi}} \cdot du - a \left(\frac{1}{2} - \Phi\left(\frac{a}{b}\right) \right) \end{aligned}$$

که برای محاسبه $\Phi\left(\frac{a}{b}\right)$ به جدول رجوع می‌کنیم.

همچنین مشابه حالت قبل تغییر متغیرهای زیر را در نظر می‌گیریم:

$$v = -\frac{1}{2}u^2$$

$$-udu = dv$$

به این ترتیب خواهیم داشت:

$$\int_{-\infty}^{-\frac{1}{2}a^2/b^2} \frac{b}{\sqrt{2\pi}} e^v dv = \frac{b}{\sqrt{2\pi}} e^{-\frac{a^2}{2b^2}}$$

لذا

$$P(\varphi^+ > 0 | \sigma^* = 1.22) = 2 * \frac{1.22}{\sqrt{2\pi}} e^{-\frac{1.22^2}{8}} - 1.22^2 \left(\frac{1}{2} - \Phi\left(\frac{1.22}{2}\right) \right) \approx 1/5$$

در واقع هنر این روش، مستقل کردن σ^* از r_t (فاصله تا هدف) است که نامعلوم بود. به تعبیر دیگر، توانسته‌ایم روابط را از هدف مستقل کنیم.

در محاسبات فوق با σ^* توانسته‌ایم به یک عدد (1/5) برسیم که از نظر عملی بسیار مطلوب و کارآمد تلقی می‌گردد. اکنون چون می‌دانیم به ازای پارامترهای بهینه احتمال موفقیت 1/5 است؛ آن را در هر مرحله حفظ می‌کنیم تا به حالت بهینه برسیم. توجه داشته باشید که تنها فرض به کار گرفته شده در این مسأله استقلال است. بر اساس فرض استقلال، این روش سرعت رسیدن به کمینه محلی را حداکثر می‌کند. از آن جا که پارامتر کنترلی اندازه قدم‌هاست، در نزدیکی کمینه محلی احتمال موفقیت کم می‌شود و نهایتاً الگوریتم در کمینه محلی متوقف خواهد شد. در حقیقت آن چه انجام شده است تنها حداکثر کردن سرعت است و در اولین (نزدیک‌ترین) کمینه متوقف خواهد شد و نخواهد توانست کمینه عمومی را پیدا کند. نکته‌ای که بیشتر نیز به آن اشاره داشتیم ضعف تئوریکی است که عمدتاً بر روش‌های تکاملی حاکم است. اثبات‌هایی از این دست معمولاً به ندرت در مورد الگوریتم‌های تکاملی وجود دارند. نکته پایانی این که برای (μ, λ) ، $ES(\mu, \lambda) \approx 7\mu$ سرعت همگرایی حداکثر را می‌دهد.

۱۲۲-۱ انواع ES و استراتژی‌های آن

بیش از پرداختن به انواع روش‌های ES و استراتژی‌های مربوطه، ابتدا ویژگی‌های ES را در قالب جدول زیر مرور می‌کنیم:

اعداد حقیقی	بازنمایی
تصادفی یکنواخت (μ)	جمعیت اولیه
تصادفی یکنواخت ($\lambda \approx 7\mu$)	انتخاب والدین
$P_c < 0.4$ محلي-عمومي	بازترکیبی
$P_M = 1, \sigma$	جهش
μ, λ متناسب با شایستگی Q-tournament $\mu + \lambda$	انتخاب بازماندگان
تعداد ارزیابی	شرط خاتمه
تطبیق با خود تطبیقی بودن اندازه قدم‌ها و جهش	ویژگی خاص

البته مطالب ذیل در خصوص استراتژی‌های ES نیز بر اساس نحوه جهش در آن است که در بخش‌های پیشین دیدم و در اینجا به اختصار جمع‌بندی خواهد شد.

$ES(1 + 1)$

همان گونه که در بخش پیش به تفصیل بررسی شد، برای این نوع از ES از استراتژی $1/5$ موفقیت استفاده می‌شود.

$ES(\mu, / + \lambda)$

در این حالت، ابتدا σ و سپس x را جهش می‌دهیم.

x_1	x_2	...	x_n	σ
-------	-------	-----	-------	----------

به این ترتیب مقادیر جدید به صورت زیر به دست می‌آیند:

$$\begin{cases} \sigma' = \sigma e^{-\tau N(0,1)} \\ x' = x + \sigma' N(0, 1) \end{cases}$$

$ES(\mu, / + \lambda)$

در این حالت به جای در نظر گرفتن اندازه گام واحد برای تمامی ابعاد، گام را برای هر بعد متفاوت در نظر می‌گیریم و همان گونه که پیش‌تر دیدیم با استفاده از تکامل و به صورت زیر می‌توان σ_i ها را به دست آورد:

x_1	...	x_n	σ_1	...	σ_n
-------	-----	-------	------------	-----	------------

در حقیقت، اندازه و جهت حرکت، به ترتیب یادگیری عمومی و محلی تلقی می‌گردند:

$$\sigma'_i = \sigma_i e^{-\tau N(0,1) + \tau' N_i(0,1)}$$

$ES(\mu, / + \lambda)$

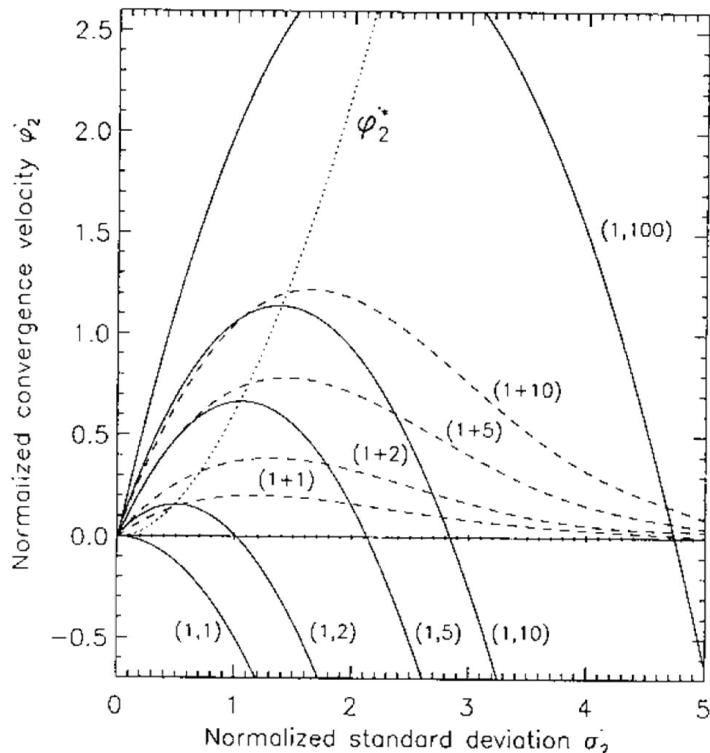
همچنین دیدم که می‌توانیم زوایای چرخش را نیز به مجموعه فوق اضافه کنیم. به این ترتیب خواهیم داشت:

x_1	...	x_n	a_{11}	...	a_{nn}
-------	-----	-------	----------	-----	----------

که a_{ii} ها داریه‌های ماتریس پایین مثلثی زیر هستند:

$$\begin{bmatrix} a_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$$

پیش از پایان بحث، منحنی نرمال شده سرعت همگرایی بر حسب اندازه قدم را بررسی می‌کنیم. توجه داشته باشید که اندازه قدم نرمالیزه شده، در واقع همان σ^* است.



در شکل فوق منحنی‌ها برای $ES(1 + \lambda)$ و $ES(1, \lambda)$ رسم شده‌اند. روشن است که در $ES(1, 1)$ که از ابتدا مقادیر همگرایی منفی داریم، اساساً همگرایی خواهیم داشت. به تعبیر دیگر الگوریتم تصادفی محسوب شده و به سمت هدف حرکت نمی‌کند. با این حال در سایر $ES(1, \lambda)$ ‌ها، از جایی به بعد سرعت همگرایی منفی شده است. این رفتار به معنای توانایی در فرار از کمینه محلی است.

به علاوه برای $ES(1 + \lambda)$ نیز نحوه همگرایی در این شکل مشاهده می‌شود. در این حالت با توجه به حافظه دار بودن، تضمین همگرایی را داریم. با این حال، محدوده اندازه قدم‌ها کاملاً مشخص است و امکان تخطی از آن وجود ندارد. با افزایش λ حداقل سرعت همگرایی افزایش می‌یابد. در مقابل تعداد ارزیابی‌ها نیز افزایش خواهد داشت و به تعبیر دقیق‌تر λ برابر خواهد شد. در $\mu = 7\lambda$ ، افزایش اندازه گام‌ها منطقی است. از آن جا به بعد تعداد ارزیابی‌ها افزایش قابل ملاحظه‌ای خواهد داشت. به تعبیر دیگر، اگر منحنی‌ها را به تعداد ارزیابی‌ها نرمال کنیم، تا $\mu = 7\lambda$ افزایش خواهیم داشت و بعد از آن، نه.

اگر مقادیر حداقل $ES(1 + \lambda)$ را به هم متصل کنیم، تقریباً یک خط خواهیم داشت و مقدار $\mu = 7\lambda$ از همین جا به دست می‌آید.

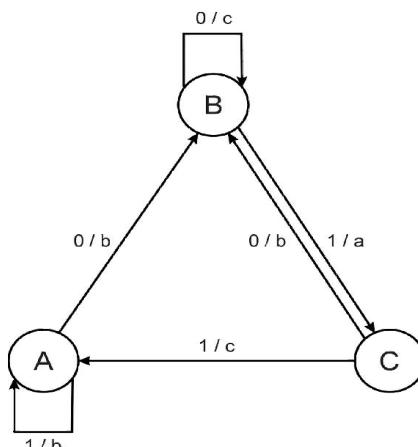
۲۳-۱ برنامه نویسی تکاملی^۱ (EP)

ابداع این روش در دهه ۱۹۶۰ و در امریکا توسط Fogel صورت گرفته است. موارد استفاده آن در ایندا در یادگیری ماشین و امروزه عمدهاً در بهینه سازی عددی است.

EP دو نسخه گراف و اعداد حقیقی دارد و عمدت تفاوت آن با ES این است که در EP بازترکیبی نداشته و تنها جهش داریم. EP فریم ورک بالاتر از ES دارد و توان حل مسائلی را دارد که ES قادر به حل آنها نیست. به ویژه نسخه اعداد حقیقی EP بسیار به ES شبیه است، البته با این تفاوت که بازترکیبی ندارد. تفاوت دیگر EP با ES، طریقه انتخاب بازماندگان است که در هر دو نسخه $\mu + \mu$ است. همچنین شرط خاتمه EP نیز تعداد ارزیابی هاست.

ماشین حالت محدود^۲ (FSM)

در اینجا خروجی‌ها در تغییر حالت به وجود می‌آیند. اما می‌توانند در خروجی هم باشد.



$$S = \{A, B, C\}$$

$$I = \{0, 1\}$$

$$O = \{a, b, c\}$$

یکی از کارهای مهم این ماشین‌ها پیش‌بینی است. اگر خروجی این ماشین ورودی بعدی باشد، در حال پیش‌بینی است.

یکی از مسائلی که سعی شد با این ماشین حل شود، پیش‌بینی اعداد اول بود که خیلی خوب هم نتوانست آن را حل کند. اگر ورودی ۵ تا ۰ و ۱ باشد و خروجی نیز ۵ تا و از این تعداد ۳ تا مانند هم باشند، شایستگی ۰.۶ خواهد داشت (تعداد درست تقسیم بر تعداد کل).

در کلیه مواردی که بازنمایی به صورت گراف باشد، نمی‌توان بازترکیبی تعریف کرد. کل گراف یک کروموزوم در نظر گرفته می‌شود. در الگوریتم‌هایی که تا به اینجا دیدیم، کروموزوم‌های خطی با تعداد زن ثابت وجود داشتند. به تعبیر

¹ Evolutionary Programming

² Finite State Machine

دیگر تکامل درون گونهای وجود داشت. با این حال، زمانی که تعداد ژن‌ها تغییر می‌کند، تغییر گونه یا تکامل بین گونهای خواهیم داشت. تکامل بین گونهای یکی دیگر از قابلیت‌های EP است که این به ساختار گراف برمی‌گردد. چرا که وقتی با گراف سر و کار داریم نمی‌توان ژن‌ها را مشخص کرد و به این ترتیب تکامل بین گونهای خواهیم داشت.

جهش در EP، می‌تواند به صورت حذف یا اضافه شدن حالت، تغییر در گذار حالت، تغییر حالت اولیه، تغییر یک سمبول خروجی، و ... باشد. انتخاب بازماندگان بر اساس $\mu + \mu$ صورت می‌گیرد و با استفاده از تورنومنت Q.

جهش در EP

همان طور که پیشتر به تفصیل بررسی شد، اگر r فاصله تا هدف و به تعییری، میزان شایستگی باشد:

$$\frac{n\sigma}{r} = \sigma^* = 1.22$$

روشن است که $\sigma = \sqrt{\frac{r}{n}} * 1.22$ و در واقع $r \propto \sigma$. به این ترتیب، σ می‌بایست متناسب با عکس شایستگی، یا هزینه باشد یعنی:

$$\sigma \propto \sqrt{\phi}$$

و در نتیجه:

$$\sigma = \sqrt{a\phi + b}$$

r^2 را هزینه در نظر می‌گیریم. بنابراین جذر ترکیب خطی‌ای از هزینه می‌تواند به عنوان اندازه قدم در نظر گرفته شود:

$$x'_i = x_i + \sqrt{\beta_i \phi(\bar{x}) + \gamma_i N_i(0,1)}$$

اندازه قدم EP متناسب با جذر هزینه در نظر گرفته می‌شود. یعنی هر چه هزینه کمتر باشد، اندازه قدم کوچک‌تر است و بالعکس. به علاوه، این ترکیب خطی کمک می‌کند عبارت زیر رادیکال بزرگ‌تر از ۱ باشد.

برای $EP(1+1)$ داریم:

$$x'_i = x_i + \sqrt{\gamma_i} N(0,1)$$

$$\gamma'_i = \gamma_i + \sqrt{\xi \gamma_i} N(0,1)$$

در اینجا انتخاب ضریب به گونه‌ای است که واریانس جدید و قدیم خیلی نزدیک باشند. وقتی اندازه قدم‌ها خیلی نزدیک باشد، تفاوتی ندارد که ابتدا جهش را اعمال کنیم یا بازترکیبی. به همین دلیل، در اینجا ترتیب گفته شده برای بازترکیبی و جهش در ES برقرار نیست.

برای $ES(\mu, +\lambda)$ ، α به گونه‌ای است که σ های جدید و قدیم خیلی نزدیک هستند ($\alpha \approx 0.2$).

x_1	...	x_n	σ_1	...	σ_n
-------	-----	-------	------------	-----	------------

لذا خواهیم داشت:

$$\sigma'_i = \sigma_i (1 + \alpha N(0,1))$$

$$x'_i = x_i + \sigma'_i N(0,1)$$

در ES هم علاقمندیم تغییرات قدم‌ها تدریجی باشد، با این حال در ES، سرعت اندکی بیشتر از EP است. اگر تغییرات زیاد باشد، فرصتی برای بررسی اثر آن‌ها خواهیم داشت و نمی‌توانیم مشخص کنیم اندازه قدم به کار گرفته شده مناسب بوده است یا نه. باید توجه داشت که حتی اگر x خوبی داشته باشیم که به واسطه یک σ ضعیف به جایی نامناسب بردشود، x مذکور در فرآیند تکامل حذف خواهد شد. به علاوه ذکر این نکته ضروری است که در اعداد حقیقی عملکردی به مراتب بهتر از EP دارد. به عنوان مثال می‌توان بهتابع *Ackley* اشاره کرد که به صورت زیر تعریف می‌شود:

$$f(x) = -20 \exp(-0.2 \sqrt{\frac{1}{n} \cdot \sum_{i=1}^n x_i^2}) - \exp(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i)) + 20 + e$$

۱-۲۴ برنامه نویسی ژنتیک^۱ (GP)

پیش از آغاز بحث، مرور مختصری بر مباحث گفته شده درباره EP خواهیم داشت. دیدیم که در ES جهش حتماً گوسی است در حالی که در EP می‌تواند غیرگوسی باشد، مثلاً با توزیع زنگولهای یا کوشی صورت گیرد. به علاوه در EP، نحوه انجام جهش و خطی و غیرخطی بودن آن گستردگرتر از ES است که بیش از این به چهارچوب^۲ گستردگر از آن یاد کردیم. همچنین دیدیم که EP به دلیل $\mu + \mu$ و عدم داشتن بازترکیبی عملکرد خوبی روی اعداد حقیقی ندارد و دچار همگرایی زودرس می‌گردد. در مقابل EP مزایایی دارد که از جمله آن‌ها می‌توان به بازنمایی گراف و امکان تغییر حالت اولیه، ورودی و خروجی و همچنین حالت‌ها اشاره کرد. در انجام جهش نیز، در EP می‌تواند هر بار یک جهش انجام شود و یا با استفاده از RW، هر بار به یکی امکان جهش داده شود.

EP دارای قابلیت تکامل بین گونه‌ای است و از این جهت GP نیز به آن شبیه است.

GP یکی از جدیدترین الگوریتم‌های تکاملی است که در ۱۹۹۰ توسط کوزا^۳ معرفی شد. این الگوریتم اساساً برای تولید اتوماتیک برنامه پیشنهاد شده بود، با این حال در آن زمینه نتوانست توفیقی را که در یادگیری ماشین کسب کرده به دست آورد. در حوزه یادگیری ماشین، GP توانست با شبکه عصبی رقابت کند، از این جهت که همانند شبکه عصبی قادر به تولید مدل است. در روش‌های تکاملی ارائه شده پیش از EP در هر نسل پاسخ‌ها تکامل پیدا می‌کرد تا به جواب بهینه برسد. اما EP در حالت استاندارد بازنمایی گراف و GP بازنمایی درخت دارند. این بازنمایی‌ها ما را قادر می‌سازند که به جای جواب، مدل تولید کنیم. مانند آن‌چه در شبکه عصبی رخ می‌دهد که رابطه ورودی و خروجی یک سیستم مدل می‌شود. بر اساس آن‌چه گفته شد، GP برای طبقه‌بندی، تقریب تابع،

¹ Genetic Programming

² framework

³ Koza

پیش بینی و مدل سازی کاربرد دارد.

یکی دیگر از ویژگی‌های GP نیاز آن به جمعیت با تعداد زیاد است. در حالی که تعداد متعارف جمعیت در الگوریتم‌های معمولی ۱۰ تا ۱۰۰ است، در GP به جمعیتی در رنچ هزار و میلیون نیاز داریم. به علاوه، در GP کروموزم‌ها غیرخطی‌اند یعنی درخت‌ها می‌توانند هم از لحاظ پهنا و هم از نظر عمق متفاوت باشند. در GP نیز مانند EP تکامل به جای درون گونه، مابین گونه‌ها اتفاق می‌افتد. یعنی فرزندان شباhtی با والدین ندارند. یکی از معایب GP نیز همین مسأله به شمار می‌آید، چرا که اختلاف زیاد فرزندان با والدین سبب می‌شود کندی مورد نیاز در تکامل را نداشته باشد.

بر خلاف EP که در آن تنها جهش انجام می‌شد و بازترکیبی نداشتیم، در GP نرخ جهش بسیار پایین و حتی صفر است، در حالی که بر عکس بازترکیبی در آن از اهمیت ویژه‌ای برخوردار است.

به عنوان مثال فرض کنید بانکی اطلاعاتی از وام گیرندگان را دارد و بناست مدلی ایجاد کند که به چه کسانی وام بدهد یا ندهد. شایستگی با گذاشتن داده‌ها در مدل ساخته شده و به دست آوردن درصد درست‌ها به دست می‌آید.

در شبکه عصبی و خیلی اوقات GP، تابع هدف حداقل کردن مجموع مربعات خطاست:

$$\frac{1}{2} \sum_{l=1}^L (y^l - y^{*l})^2$$

مسائلی که می‌توان با استفاده از درخت حل کرد شامل توابع ریاضی، فرمول‌های منطقی، برنامه نوبسی.

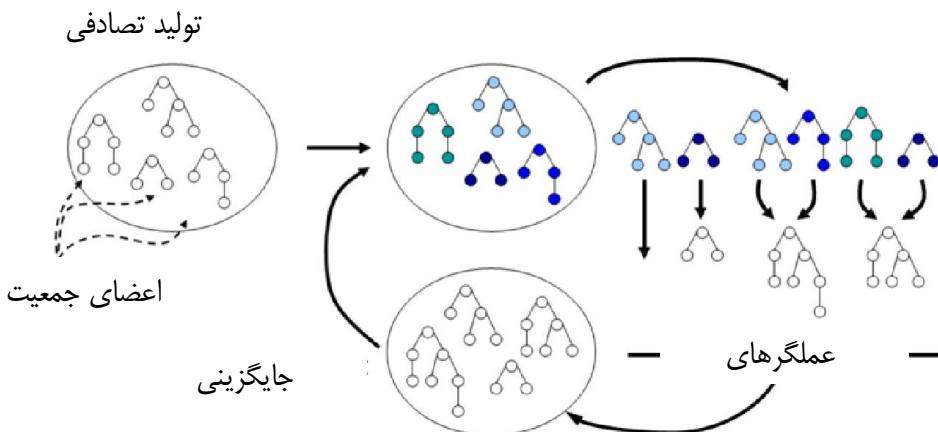
هر مسأله‌ای را که بتوان با یکی از این سه یا ترکیب آن‌ها مدل کرد را می‌توان با درخت مدل کرد. در درخت تعدادی گره^۱ و ترمینال^۲ که به ترتیب مجموعه توابع^۳ (عملگر) هستند و متغیرها و ثوابت.

عملکرد کلی در GP به این صورت است که هر یک از افراد جمعیت یک درخت را تشکیل می‌دهد که در حالت ساده نشان‌دهنده‌ی یک فرمول ریاضی یا منطقی و در حالت پیچیده‌تر، می‌تواند یک برنامه کامپیوتروی باشد. این فرمول یا برنامه، خروجی‌ای تولید می‌کند که با استفاده از آن، می‌توان مقدار شایستگی را محاسبه کرد. سایر روش‌های مورد استفاده در برنامه‌نویسی ژنتیک، مشابه بقیه روش‌های تکاملی است. شمای کلی برنامه‌نویسی ژنتیک در شکل زیر آورده شده است.

¹ Node

² Terminal

³ Function set



شکل ۱-۰- شمای کلی برنامه نویسی ژنتیک

در EP کروموزوم گراف است و در GP درخت. پیشتر در بحث بازنمایی دیدیم که بازنمایی مطلوب می‌بایست از یک سو همه راه حل‌های ممکن را نمایش دهد و از سوی دیگر حتی المقدور فضا جستجو را کوچک کند. اما بازنمایی درخت فضای جستجوی خیلی بزرگی دارد چرا که درخت‌ها هم از پهنا و هم از عمق می‌توانند بزرگ شوند و به این ترتیب فضا خیلی بزرگ خواهد شد و این یکی از ضعف‌های GP به شمار می‌آید.

عبارت سمبولیک^۱

اولین گام تعریف دو مجموعه توابع^۲ (F) و مجموعه ترمینال‌ها (T) برای مسأله است. به عنوان مثال مجموعه توابع را $\{/, +, -, *, \text{---}\}$ و مجموعه ترمینال‌ها، یا به عبارتی متغیرهای مسأله را اعداد حقیقی در نظر می‌گیریم. در ادامه بر اساس قواعد ساده‌ای که وجود دارد، می‌توان عبارت سمبولیک را تولید کرد یا سمبولیک بودن یا نبودن یک عبارت را مشخص کرد. این قواعد به شرح ذیل هستند:

- ۱- هر مجموعه ترمینال که زیر مجموعه مجموعه ترمینال باشد، یک عبارت سمبولیک است.
- ۲- هر تابع f از n عبارت سمبولیک e_1, e_2, \dots, e_n که f متعلق به مجموعه توابع باشد، عبارت سمبولیک است. n برابر تعداد ورودی‌های^۳ تابع f است.
- ۳- در حالت ترکیب دو مورد فوق، لازم است که نوع آرگومان نیز صحیح باشد. مثلاً برای یک عبارت منطقی می‌توان درست و نادرست داشت و نه یک مقدار عددی.

این فرمولاسیون تولید و شناسایی درخت را ساده می‌کند.

مثال: برای دو مجموعه توابع و ترمینال زیر، عبارت سمبولیک بودن یا نبودن هر مورد را مشخص کنید:

```
function set = {+, -, *}
terminal set = {x, r}
```

¹ Symbolic

² Function set

³ Arity

$x + 2$, $2x$, x^2 عبارت سمبليک هستند و $x/2$ عبارت سمبليک نیست.

اکنون اگر مجموعه توابع و ترمinal‌ها را به صورت زیر در نظر بگيريم:

function set = $\{+, -, \sin\}$

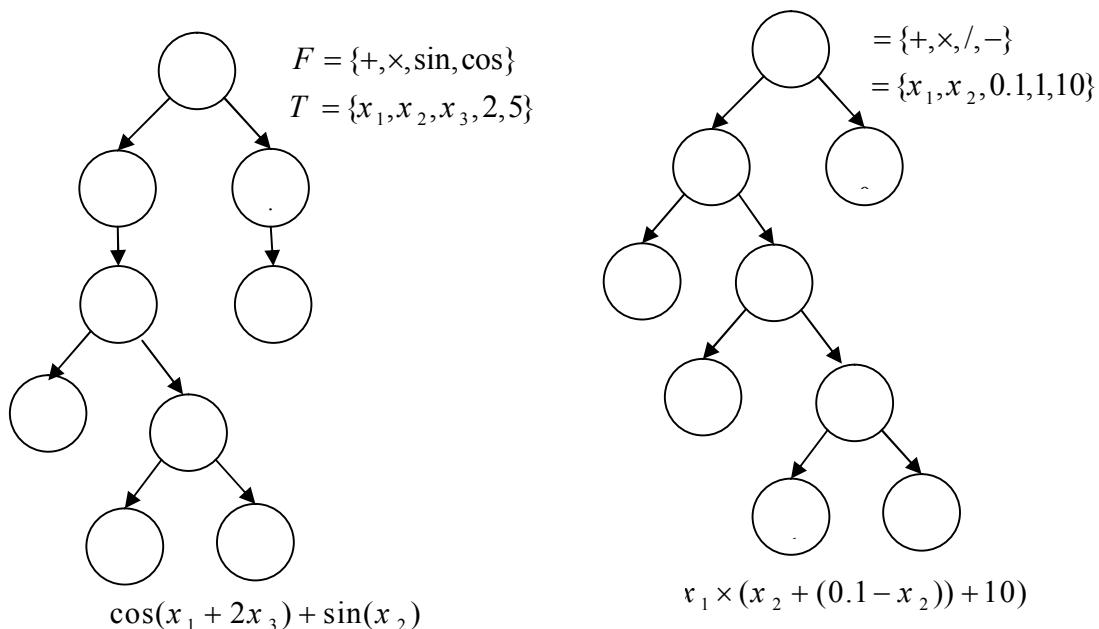
termnal set = $\{x, r\}$

عبارت سمبليک $\sin(x + 2)$ عبارت سمبليک هستند اما $\sin4$, $\sin x$ عبارت سمبليک نیست.

بازنمایی در GP

بازنمایی در GP عموماً به صورت گراف و به ویژه درخت صورت می گیرد. که اساس آن را در بخش تولید عبارات

سمبلیک توضیح دادیم. شکل زیر نمونه هایی از درخت را نمایش می دهد:



شکل ۲-۰- نمونه‌ای از یک درخت GP

انتخاب مجموعه توابع و مجموعه پایانه‌ها باید به گونه‌ای انتخاب باشد که دو شرط بسته‌بودن^۱ و کافی بودن^۲ را ارضاء کنند. ویژگی بسته بودن به این معناست که هر کدام از تابع‌های مجموعه توابع بتواند مقادیری را که توسط تابع‌های دیگر آن مجموعه برگردانده می‌شود و همچنین هر کدام از عناصر مجموعه پایانه‌ها را به عنوان ورودی خود بگیرد. اما برای خیلی از مسائل اراضی این شرط امکان پذیر نیست و مسئله نیاز به نوع-داده‌های^۳ مختلفی دارد. به عنوان مثال، مسئله‌ای ممکن است هم نیازمند توابع بولی مثل *And*, *OR* و هم تابع حسابی مثل

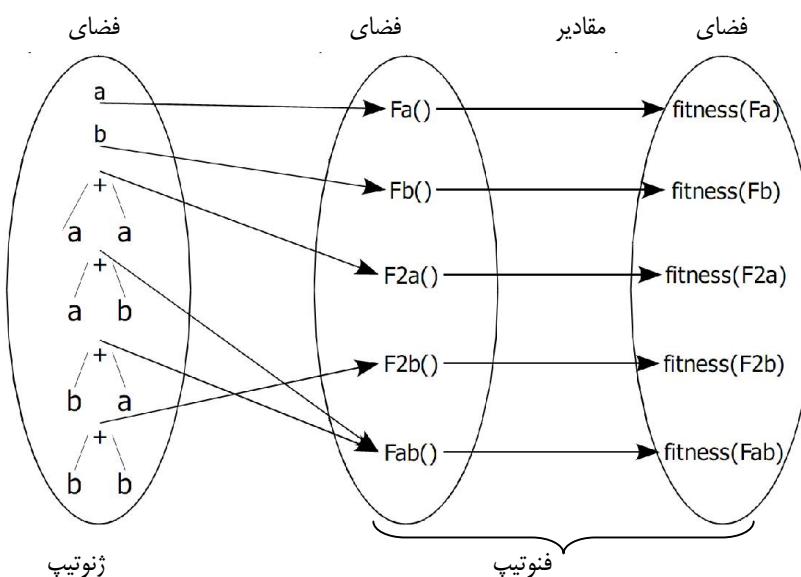
¹ Closure

² Sufficiency

³ Data Type

/,-,×,+ باشد. برای چنین حالت‌هایی برنامه نویسی ژنتیک قویاً تعیین نوع شده^۱ ارائه شده است که برای هر تابعی مشخص می‌کند که نوع ورودی‌هایی که می‌پذیرد و همچنین نوع خروجی آن به چه صورت است. علاوه بر این، نوع هر پایانه نیز مشخص می‌گردد. ویژگی کافی بودن نیز به این معناست که مجموعه پایانه‌ها و مجموعه توابع توانایی بیان پاسخ را داشته باشند. برای برخی از مسائل به راحتی نمی‌توان گفت که آیا یک مجموعه توابع و مجموعه پایانه‌های خاص برای یک مسئله کافی است یا نه، و معمولاً برای این گونه مسائل با توجه به آزمایش‌ها و تجربه یک مجموعه عناصر اولیه مناسب انتخاب می‌شود.

در برنامه نویسی ژنتیک، غالباً نگاشت ژنوتیپ به فنوتیپ یک نگاشت چند به یک است. برای یک مجموعه توابع و مجموعه پایانه‌ها درخت‌های مختلفی با عملکرد یکسان می‌توان ساخت. در شکل ۳-۱ نمونه‌ای از یک نگاشت ژنوتیپ به فنوتیپ آورده شده است.



شکل ۳-۰- نگاشت بین فنوتیپ و ژنوتیپ در برنامه نویسی ژنتیک

روش‌های بازنمایی دیگری نیز برای برنامه نویسی ژنتیک پیشنهاد شده است. به عنوان مثال برنامه نویسی ژنتیک خطی، برنامه نویسی ژنتیک گرافی، برنامه نویسی ژنتیک کارتزین^۲ و برنامه نویسی ژنتیک گرامری از جمله برنامه نویسی‌های ژنتیکی هستند که از بازنمایی‌های غیر از درخت استفاده می‌کنند.

۱-۲-۲۴-۱ نحوه ایجاد درخت تصادفی

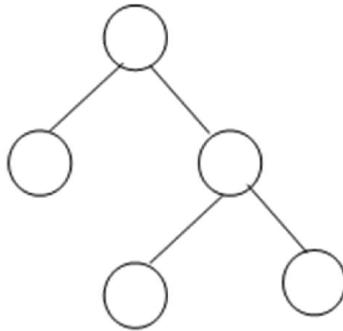
در GP جمعیت اولیه متشکل از درخت‌های تصادفی است. یک عمق حداقل، d_{max} تعریف می‌کنیم و تا زمانی که یه آن نرسیده‌ایم از مجموعه توابع و یا مجموعه ترمینال‌ها برای تولید افزودن به درخت استفاده می‌کنیم. مثلاً

¹ Strongly Typed GP

² Cartesian

برای $d_{max} = 2$ و مجموعه‌های زیر خواهیم داشت:

function set = {+, -}
terminal set = {x, r}



اگر در دایره بالایی $+$ قرار دهیم، نیاز به دو ورودی خواهیم داشت. ورودی سمت چپ را x و ورودی سمت چپ را یک شاخه قرار می‌دهیم. اگر در دایره سمت چپ $-$ قرار دهیم، مجدداً برای آن نیز نیازمند دو ورودی خواهیم بود که می‌توان آن‌ها را x و 2 در نظر گرفت. به این ترتیب با رسیدن به حداکثر عمق مجاز فرآیند تولید درخت متوقف می‌گردد. این درخت معادل عبارت $(x - 2) + x - 2$ یعنی $2x - 2$ است.

برای تولید درخت کامل، فقط در عمق حداکثر (پایین‌ترین سطح) از مجموعه ترمینال‌ها استفاده می‌کنیم. اما برای ساختن درخت ناقص، در هر عمقی امکان استفاده تصادفی از هر دو مجموعه ترمینال‌ها و توابع را خواهیم داشت. در جمعیت اولیه یا تمام درخت‌ها کامل هستند یا همگی ناقص و یا بخشی کامل و بخشی ناقص؛ که معمولاً از حالت آخر استفاده می‌شود.

d_{max} در فرآیند تولید جمعیت اولیه و جهش که تولید زیردرخت تصادفی داریم مورد استفاده قرار می‌گیرد. در GP یا بازترکیبی انجام می‌شود و یا جهش. هر دو همزمان انجام نخواهند شد. هر یک از این دو رخداد احتمالی دارند که مجموع این احتمال‌ها برابر یک است. دلیل رخداد تنها یکی از این دو است که حتی با رخداد یکی از آن‌ها نیز تفاوت فرزندان و والدین زیاد است.

تولید جمعیت اولیه

در GP روش‌های مختلفی برای تولید جمعیت اولیه وجود دارد که مهم‌ترین آنها عبارتند از:

۱- روش کامل^۱: در این روش، درخت‌های اولیه تا رسیدن به سقف مشخصی از تعداد گره‌ها، توسعه داده می‌شوند (یعنی ابتدا از عناصر F استفاده می‌شود و در انتهای تولید برگ‌های نهایی از عناصر T استفاده می‌شود). البته لزوماً تعداد گره‌های نهایی برابر با سقف تعیین شده نیست، اما نزدیک به آن بوده و از آن فراتر نخواهد کرد. درختانی که در این حالت تولید می‌شوند، دارای ساختار متوازن هستند.

^۱ Full

۲- روش رشده‌ی ۱: در این روش، درخت‌های اولیه با انتخاب تصادفی از مجموعه F و T انتخاب شده و درختان نهایی می‌توانند کاملاً غیرمتوازن، بسیار کوچک و یا بسیار بزرگ باشند. البته در عمل (آزمایش‌ها انجام شده) مشاهده شده است که معمولاً با این روش، درختان خیلی بزرگ شکل نمی‌گیرند.

۳- روش نیمه کامل^۲: در این روش، نیمی از جمعیت به روش کامل و نیم دیگر به روش رشده‌ی تولید می‌شوند.

جهش در GP

عملگر جهش روی یک عضو از جمعیت اعمال می‌گردد و یک عضو جدید تولید می‌کند. در حالت کلی یک زیردرخت از برنامه انتخاب می‌شود و یک زیردرخت جدید جایگزین آن می‌گردد. نرخ اعمال این عملگر پایین است (مثلاً ۵٪). **Error! Reference source not found.** مثالی از عملکرد جهش را نشان می‌دهد

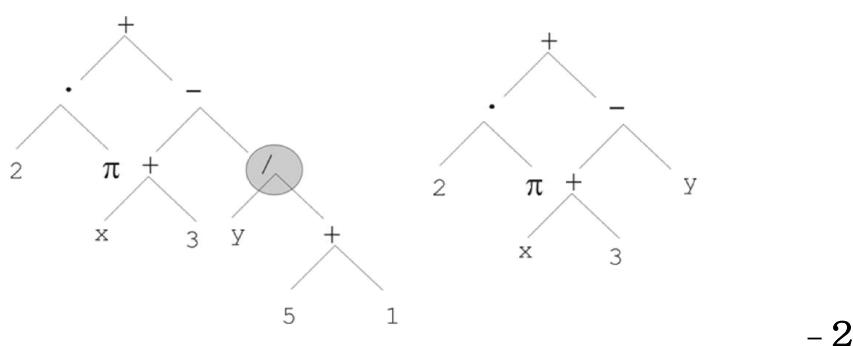
عملگرهای جهش مختلفی برای برنامه نویسی ژنتیک معرفی شده است برخی از آنها از این قرار هستند:

- ۱ - جهش زیردرخت: در این روش، یک زیردرخت از درخت اصلی به صورت تصادفی انتخاب شده و با زیردرخت تصادفی جدید دیگری جایگزین می‌شود.

یک زیردرخت به صورت تصادفی انتخاب می‌شود و آن بخش با زیردرختی که خود نیز به صورت تصادفی تولید شده است، جایگزین می‌گردد. پس در سه مرحله وجود تصادف را داریم: انتخاب زیردرختی که بناست جایگزین شود، زیردرختی که جایگزین آن خواهد شد و تصمیم گیری در مورد رخداد جهش یا باز ترکیبی.

همان طور که پیش‌تر نیز اشاره شد، به علت تفاوت‌های فراوان فرزندان و والدین نرخ جهش بسیار اندک و در بسیاری موارد صفر در نظر گرفته می‌شود. جهش می‌تواند هم طول و هم عرض درخت را تغییر دهد. به علاوه، فرزند حاصل ممکن است اندازه‌ای کوچک‌تر از والد یا بزرگ‌تر از آن داشته باشد.

نمونه‌ای از این نوع جهش را در شکل زیر مشاهده می‌نمایید:

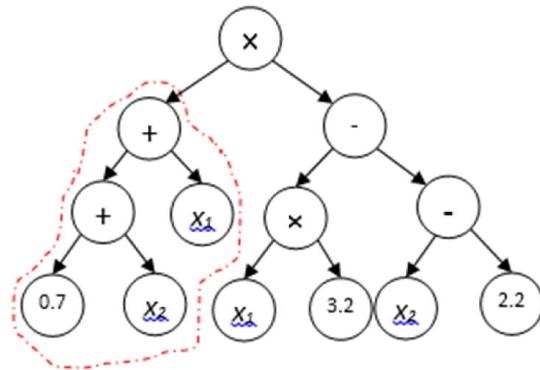
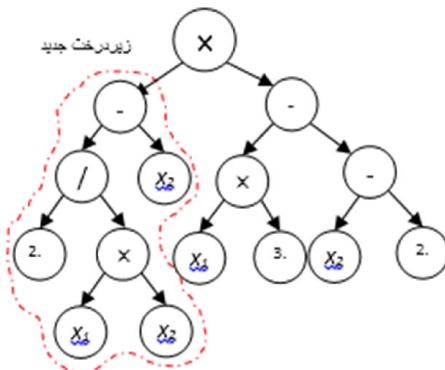
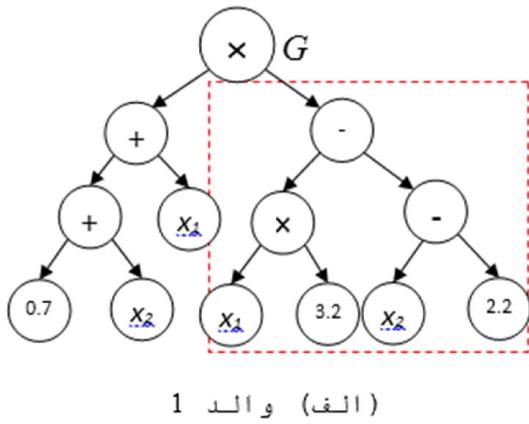
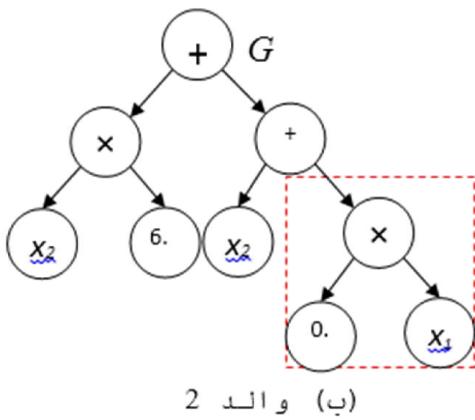


- 2

¹ Growing

² Ramped Half-and-Half

جهش گره: در این روش، یک گره به صورت تصادفی با نوع مشابه خود (اگر ترمینال است، با ترمینال و اگر تابع است، با تابع) جایگزین می‌گردد.



جهش انقباض^۱: در این روش، یک زیردرخت تصادفی از درخت اصلی، با زیردرخت خود، جایگزین می‌شود.

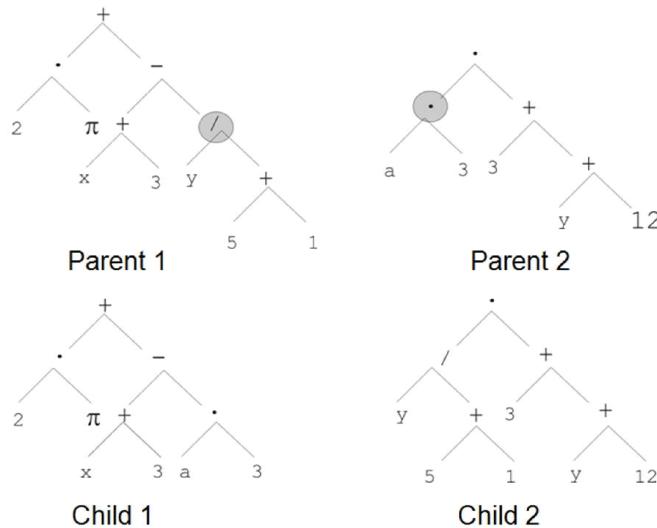
جهش جایگزینی^۲: در این روش، دو زیردرخت تصادفی غیر همپوشان از درخت اصلی، با یکدیگر جایگزین می‌شوند.

بازترکیبی در GP

به منظور انجام باز ترکیبی، دو نقطه تصادفی در دو درخت والد در نظر گرفته می‌شود و زیردرخت‌های مربوطه آن‌ها جا به جا می‌شوند. نمونه‌ای از این فرآیند در شکل زیر نمایش داده شده است:

¹ Shrink

² Swap



انتخاب والدین

استراتژی انتخاب مفهومی است که برنامه نویسی ژنتیک را از جستجوی تصادفی متمایز می‌سازد. سه روش عمدۀ انتخاب در برنامه نویسی ژنتیک، انتخاب مبتنی بر شایستگی^۱، انتخاب حریصانه^۲ و انتخاب رقابتی^۳ است.

1-انتخاب مبتنی بر شایستگی: در این روش انتخاب، احتمال انتخاب هر عضو از جمعیت با شایستگی آن عضو مناسب است.

به این ترتیب که دو گروه ساخته می‌شوند که یکی متشکل از x درصد بهترین جمعیت و دیگری شامل $x - 100$ درصد باقی مانده است. برای والدین 80% از گروه اول و 20% از گروه دوم انتخاب می‌شوند. این انتخاب از گروه‌های اول و دوم متناسب با شایستگی است که با SUS یا RW انجام می‌گیرد. موجوداتی که به این ترتیب به دست آمده‌اند، به صورت تصادفی، دو به دو جفت می‌شوند.

مقدار x برای جمعیتی با تعداد 1000، 2000، 3200، 1600 و 400 به ترتیب ۸۰۰۰، ۴۰۰۰، ۱۶۰۰ و ۸۰ پیشنهاد شده است که به صورت تجربی به دست آمده‌اند.

در GP نیز مشابه GA انتخاب بازماندگان وجود ندارد. به صورت μ, μ (مولد^۴) است. البته در برخی کارها، استفاده از روش حالت پایدار نیز گزارش شده است. ویژگی حالت پایدار این است که فرزند تولید شده، جایگزین بدترین موجود یا یک موجود که به صورت تصادفی انتخاب شده است می‌گردد. لذا تضمین همگرایی خواهیم داشت. به علاوه، در حالتی که بدترین موجود را با فرزند تولید شده جایگزین کنیم، هرگز بهترین موجود را گم نخواهیم کرد.

¹ Fitness Proportionate Selection

² Greedy Selection

³ Tournament Selection

⁴ generational

۲- انتخاب حریصانه: انتخاب حریصانه تا حد زیادی مشابه روش انتخاب مبتنی بر شایستگی است. ولی در این روش انتخاب اعضای جمعیت به دو بخش تقسیم می‌شوند. بخش اول شامل ۲۰٪ از جمعیت که بالاترین شایستگی‌ها را دارند و بخش دوم شامل باقی اعضای جمعیت می‌شود. در این مدل انتخاب، ۵۰٪ انتخاب‌ها از بخش اول و ۵٪ از بخش دوم انتخاب می‌شوند.

2- انتخاب رقابتی: در انتخاب رقابتی به صورت تصادفی k فرد از جمعیت انتخاب می‌شود. آنگاه فردی که بیشترین شایستگی را دارد در این رقابت پیروز می‌شود، این عمل M بار (به تعداد اندازه جمعیت) انجام می‌شود و در نهایت فردی که به دفعات بیشتری در این رقابت پیروز شود برنده می‌شود. در این روش انتخاب هر چه اندازه رقابت بزرگ باشد آنگاه افرادی که شایستگی‌آنها کم است، شанс کمتری برای انتخاب شدن پیدا می‌کنند.

خاتمه الگوریتم

برای خاتمه برنامه نویسی ژنتیک از یکی از معیارهای ۱) رسیدن به حداقل شایستگی، ۲) تکرار تا رسیدن به تعداد نسل‌های مشخص، ۳) همگرایی (نزدیک شدن شایستگی‌ها در تمام جمعیت) و ۴) سکون (شبیه شدن ساختار کروموزم‌ها در تمام جمعیت)، استفاده می‌شود. اما معیار تعداد نسل‌های مشخص معمولاً بیشتر مورد توجه قرار می‌گیرد زیرا پایان پذیر بودن الگوریتم را تضمین می‌کند. جدول مشخصات الگوریتم برنامه نویسی ژنتیکی را می‌توان به صورت آنچه که در جدول ۱-۱ آمده است خلاصه نمود.

جدول ۱-۰- مشخصات الگوریتم برنامه نویسی ژنتیک

گراف، درخت	بازنمایی
جا به جا کردن زیردرخت‌ها	بازترکیبی
جایگزین کردن یک زیردرخت با زیردرخت تصادفی دیگر	جهش
مبتنی بر شایستگی	انتخاب والدین
(μ, μ) یا ($\mu + \mu$) و حفظ بهترین فرد	انتخاب بازماندگان

چاقی در GP

یکی از مشکلات GP این است که تعداد گره‌های درخت به طور بی‌رویه افزایش یابد، در حالی که افزایش قابل ملاحظه‌ای در شایستگی نداشته باشیم. این اتفاق می‌تواند در طول یا عرض درخت رخ دهد. GP ماهیت‌آ علاقمند است که در هر نسل تعداد گره‌های درختان افزایش یابد. برای چاقی چندین دلیل وجود دارد که بر اساس آن‌ها راه

حل‌هایی نیز پیشنهاد شده است:

خیلی وقت‌ها در GP به جای این که یک عدد ثابت با یک پارامتر ساخته شود، با یک زیر درخت ساخته می‌شود. دلیل این امر هم این است که عدد ثابت در واقع یک نقطه در فضا است و احتمال آن برابر عکس فضا و طبعاً صفر است. در حالی که در مقابل به روش‌های متعدد می‌توان زیر درخت‌هایی برای تولید آن عدد داشت که احتمال انتخاب یکی از آن‌ها را نسبت به احتمال انتخاب خود عدد ثابت به شدت افزایش می‌دهد. راه حل‌های کلاسیک در این حالت محدود کردن عمق و محدود کرن تعداد گره‌ها هستند. محدود کردن عمق این امکان را ایجاد می‌کند که درخت پهنانی بیشتری پیدا کند. در واقع هیچ یک از این دو روش خوب نیستند چرا که محدود کردن از یک سو این احتمال را به وجود آورد که محدودیت گذاشته شده از رسیده به جواب جلوگیری کند و از سوی دیگر سبب کاهش تنوع و آسیب به تکامل می‌شود.

پیشنهاد دیگر قرار دادن جریمه برای تعداد گره‌های است. به این معنی که هر چه تعداد گره‌ها بیشتر شود، شایستگی را کم کنیم. به این منظور می‌باشد ضریبی منفی از تعداد گره‌ها را در تابع هدف قرار دهیم و پیدا کردن ضریب مذکور کار ساده‌ای نیست. اگر ضریب بزرگ در نظر گرفته شود، تکامل به دنیال درخت‌های کوچک‌تر خواهد رفت. در مقابل اگر ضریب بیش از حد کوچک باشد، مانند در نظر نگرفتن اثر تعداد گره‌ها خواهد بود. اگر بخواهیم از ضریبی ثابت استفاده کنیم، عملأً نمی‌توان ضریبی یافت. با این حال، روش‌هایی برای یافتن ضرایب متغیر وجود دارند.

روش‌های فوق کلی هستند. رویکرد دیگر این است که در موارد خاص، از راه حل‌های مختص به آن مشکل استفاده کنیم. مثلاً در خصوص اعداد ثابت، راه حل ویرایش^۱ است. در راه حلی دیگر، واریانس هر گره محاسبه می‌شود و برای عدد ثابت واریانس صفر در نظر گرفته می‌شود. به این ترتیب اگر واریانس از آستانه‌ای کم‌تر باشد، مقدار گره با میانگین آن جایگزین می‌گردد. به تعبیر دقیق‌تر، واریانس هر گره را به ازای داده‌های آموزشی مختلف به دست می‌آوریم و در صورتی که خیلی کوچک بود، مقدار آن گره را با عدد ثابتی که برابر میانگین داده‌های فوق‌الذکر است، جایگزین می‌کیم. اگر از ابتدا این کار را در مدل انجام دهیم، از بزرگ شدن بی‌رویه درخت جلوگیری خواهد کرد.

۱-۸-۲۴-۱- تپه نوردی برای مقابله با چاقی

در GP درصد موفقیت بازترکیبی پایین است به این معنی که در 80% موقع شایستگی پایین می‌آید و اصطلاحاً گفته می‌شود که بازترکیبی مخرب بوده است. در بازترکیبی مجموع تعداد گره‌های فرزندان و والدین برابر است. وقتی درختی بزرگ‌تر می‌شود، شایستگی آن مقدار اندکی افزایش پیدا می‌کند، این در حالی است که با کوچک‌تر شدن درخت شایستگی آن به میزان قابل ملاحظه‌ای افت خواهد داشت. لذا فرزندی برای رفتن به نسل بعد انتخاب می‌شود که شایستگی آن اکیداً بزرگ‌تر از شایستگی والدین آن باشد.

¹ Edit

کدهایی وجود دارند که به آن‌ها بی‌اثر گفته می‌شود. اضافه شدن این کدها به درخت، شایستگی آن را تغییر نخواهد داد. به تعبیر دیگر، اگر زیر درخت خیلی بزرگ باشد، اضافه شدن آن سبب تغییر در شایستگی نخواهد شد. در تپه نوردی، فرزندی انتخاب می‌شود که شایستگی بزرگ‌تر یا کوچک‌تر از والدین دارد و نه برابر آن‌ها. اگر درخت تر شایسته‌تر شده باشد اما در عین حال بزرگ‌تر نیز شده باشد می‌پذیریم، اما اگر بزرگ‌شدن درخت بدون افزایش شایستگی باشد، قابل قبول نیست.

حجم بالای محاسبات در GP

GP به رغم مزایای خود دارای معایبی است که یکی از آن‌ها زمان بسیار طولانی محاسبات آن است. به عنوان مثال در محاسبه رگرسیون با استفاده از GP، به ازای x ‌های مختلف، y ‌های متفاوت داریم. می‌خواهیم با دو مجموعه توابع و ترمینال‌های زیر آن را مدل کنیم:

$$\text{function set} = \{+, -, /, \sin, \cos\}$$

$$\text{terminal set} = RU\{x\}$$

$$err(f) = \sum_{i=1}^n (f(x_i) - y_i)^2$$

به این ترتیب GP درختی معرفی می‌کند که معادل یک رابطه ریاضی است که قادر است داده‌ها را فیت کند. همان‌طور که گفتیم، در GP مدل تولید می‌شود و از تمامی داده‌های آموزشی برای محاسبه شایستگی مدل استفاده می‌گردد. برای ارزیابی مدل لازم است حالت‌های مختلف به وجود آمده و بررسی شوند تا مشخص شود که مدل در حالت‌های مختلف تا چه اندازه کارآیی دارد. بدیهی است که این امر مستلزم زمان زیادی خواهد بود. مثلاً فرض کنید هدف قرار دادن قوانین تصمیم‌گیری برای یک ربات است. در این صورت ممکن است محاسبه یک تابع شایستگی در حد چند دقیقه طول بکشد. برای به دست آوردن دستور زبان فارسی با استفاده از GP چندین ابر کامپیوتر شش ماه مشغول کار بودند. البته به رغم زمان طولانی محاسبات، این فرآیند تنها یک بار انجام می‌شود.

فصل ششم: مسائل چند هدفه

۱- مقدمه ۲۵

مسائلی که تا اینجا مورد بحث قرار دادیم، مسائل تک هدفه بودند. به این معنا که تنها یک نقطه کمینه یا بیشینه وجود داشت که می‌بایست آن را می‌یافتیم. اما دسته دیگری از مسائل وجود دارند که چند هدفه^۱ هستند. برای حل این مسائل می‌توان آن‌ها را تبدیل به چند مسأله یک هدفه کرد و این مسائل را به طور جداگانه حل کرد. راه حل دیگر، وزن دادن به هدف‌هاست. یعنی حالت زیر را در نظر بگیریم:

$$g(x) = \sum_{i=1}^K \omega_i f_i(x)$$

تعیین وزن‌ها کار دشواری است. به علاوه ممکن است وزنی در مقداری دیگر، در نزدیکی مقداری که برایش در نظر گرفته‌ایم، جواب بهتری داشته باشد. لذا لازم است که حالت‌های مختلف را لحاظ کنیم. به علاوه، برای مقادیر f ، نیاز به نرمالیزه کردن وجود دارد. چون در حالت کلی دلیل وجود ندارد که این مقادیر در یک محدوده باشند. برای وزن‌ها هم نیاز مشابهی وجود دارد و چون حداکثر و حداقل آن‌ها معلوم نیست، در نرمالیزه کردن مشکلات جدی خواهیم داشت.

روشی دیگر از تک هدفه کردن، در نظر گرفتن حالت زیر است:

$$g(x) = \sum_{i=1}^K \omega_i \|f_i(x) - c_i\|_p$$

در این روش تلاش می‌شود، حتی المقدور، هر تابع به سمت مقدار بهینه خود برود. در بسیاری مواقع ممکن است مقدار بهینه توابع را نداشته باشیم و این، مشکلی جدی به شمار می‌آید. دو روش گفته شده برای تک هدفه کردن توابع با مشکل نرمالیزه کردن و تعیین وزن مواجه هستند.

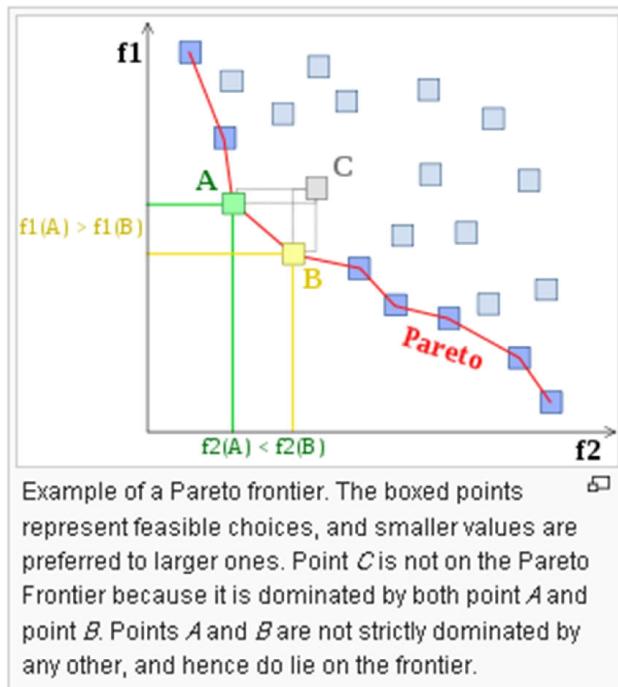
۲- روش پرتو^۲

اگر هدف‌ها مغایرت داشته باشند، یعنی افزایش یکی سبب کاهش دیگری شود، طبعاً امکان حداکثر یا حداقل کردن همزمان آن‌ها را نخواهیم داشت. در حالت ساده فرض کنید دو هدف f_1 و f_2 را داشته باشیم و جمع وزن دار آن‌ها را نیز به صورت:

$$g(x) = \omega_1 f_1 + \omega_2 f_2$$

¹ Multi Modal
² pareto

اگر دستگاه مختصات را بر حسب f_1 و f_2 داشته باشیم، منحنی‌ای که به ازای مقادیر مختلف ω رسم شده است منحنی پرتو می‌نامند.



در شکل بالا نقاط انتهایی روی منحنی قرمز رنگ به ازای $\omega_1 = 0, \omega_2 = 1$ و $\omega_1 = 1, \omega_2 = 0$ به دست آمده‌اند. و طبیعی است که در هر یک از این نقاط نسبت به دیگری f_1 بهتر و f_2 بدتر است یا بالعکس. پس نمی‌توان گفت که دان نقطه بر دیگری برتری دارد. به بیان دیگر، نقاط روی منحنی نسبت به هم نه غالب هستند و نه مغلوب.

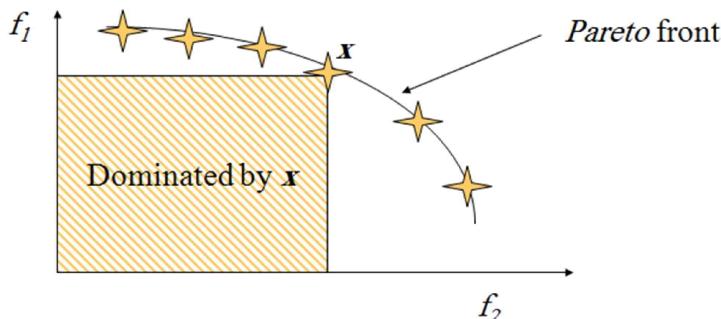
غلبه پرتو^۱

همان طور که دیدیم، نقاط روی منحنی نسبت به هم غالب ندارند. با این حال اگر برای یک نقطه روی منحنی، با کشیدن خطوط موازی محور عمومی و افقی، مستطیلی مشابه آن چه در شکل‌ها آمده است تشکیل دهیم، نقاط داخل مستطیل مذکور نسبت به نقطه x در نظر گرفته شده مغلوب هستند. به نقاط روی منحنی، نامغلوب گفته می‌شود.

مفهوم غالب پرتو با دو شرط زیر محقق می‌شود. یعنی x_1 بر x_2 غالب دارد ($x_1 > x_2$) اگر و تنها اگر:

$$\forall i: f_i(x_1) \geq f_i(x_2), \exists j: f_j(x_1) > f_j(x_2)$$

^۱ Pareto Dominance



دقت داشته باشید که شرط دوم الزام برتری اکید برای دست‌کم یک مورد را مطرح می‌سازد و برای بقیه موارد می‌توانیم بهتر یا مساوی داشته باشیم. به علاوه، مسئله برتری است که در مسائلی که به دنبال کمینه و بیشینه هستیم، مفهومی عکس یکدیگر دارد.

نقاط نامغلوب نقاطی هستند که به ازای ترکیب‌های مختلف ω جواب‌های بهینه را به دست آورده باشیم. اما این که به ازای تمامی ω ها نقاط را به دست آوریم، کار ساده و معقولی نیست. به این دلیل روشی پیشنهاد می‌شود که به جای اختصاص وزن، آن‌ها را با استفاده از روش‌های تکاملی به دست آوریم. پس به یک الگوریتم تکاملی چند جوابی نیاز داریم که به جای یک نقطه، چندین نقطه را به دست آورد؛ یعنی نقاط روی منحنی را.

چند جوابه حل کردن مسائل، در بسیاری از موارد، حتی در مسائل تک جوابه توصیه می‌شود. چرا که حتی اگر بخواهیم یک جواب داشته باشیم، مثلاً در مسائلی که تعداد زیادی کمینه محلی دارند، می‌توانیم کلیه جواب‌ها (کمینه‌های محلی) را پیدا کرده و از بین آن‌ها کمینه عمومی را بیابیم.

مسئله دیگر این است که الگوریتم‌های تکاملی، حتی در صورت وجود دو قله با شایستگی برابر، بر روی یکی از آن‌ها خواهند رفت. به این ترتیب، خیلی اوقات در مسائلی که کمینه‌های محلی زیاد یا برابر با هم دارند، بهتر است مسئله را چند جوابه حل کنیم تا یک جوابه. چرا که الگوریتم‌های تکاملی، یک قله را مدام افزایش می‌دهند و دیگری را محو می‌کنند. به این ترتیب تک جوابه حل کردن مسئله، سبب حذف یکی از دو قله برابر خواهد شد.

۲۷- الگوریتم‌های صریح و غیر صریح

دو دسته الگوریتم وجود دارند که قله‌ها را به صورت صریح و ضمنی می‌سازند. در دسته اول، به جای یک جمعیت، چند جمعیت داریم که هر جمعیت متعلق به یک قله است. اگر هر همسایگی، یک قله (جواب) را پوشش دهد، جمعیت به این همسایگی‌ها تقسیم می‌شوند که روش‌های مبتنی بر همسایگی هستند. از دسته دوم روش‌های تشریک برازنده‌گی^۱ و ازدحام^۲ را مورد بحث قرار می‌دهیم.

¹ fitness sharing
² crowding

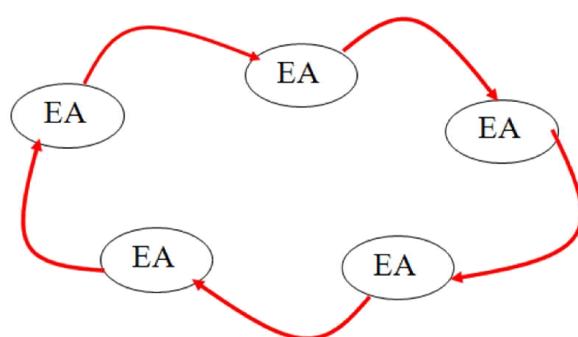
جمعیت داشتن چند m است و در هر جمعیت تکامل برای آن جمعیت و مجزا از سایر جمعیت‌ها صورت می‌گیرد. در همسایگی یک جمعیت داریم که چند همسایگی در آن تولید می‌شود.

همان طور که گفته شد، روش‌های چند جوابه را می‌توان به روش‌های مستقیم و غیر مستقیم تقسیم کرد. در روش‌های مستقیم، برای هر قله جمعیتی در نظر گرفته می‌شود. مشخص کردن تعلق موجودات به قله‌ها، به صورت اتوماتیک انجام می‌شود. این حالت را می‌توان با چند جمعیتی یا همسایگی مدل کرد. همچنین می‌توان در یک کروموزوم ژنی قرار داد که آن ژن مشخص کند کدام قله است. با این حال، در هر سه روش، تکامل تعیین می‌کند در هر قله چه جمعیت یا همسایگی‌ای داریم.

توجه به این نکته ضروری است که مقصود از همسایگی، همسایگی فیزیکی و به تعبیر دیگر شباهت موجودات نیست. همسایگی اولیه به صورت کاملاً تصادفی ساخته می‌شود و پس از آن تکامل تعیین می‌کند که کدام موجودات به سمت کدام قله حرکت کنند.

مهاجرت

به علاوه، استفاده از روش‌های چند جمعیتی، در صورتی که جمعیت‌ها مستقل از هم باشند فایده‌ای ندارد. معادل الگوریتمی است که به جای یک بار چند بار اجرا شود. برای موضوعیت داشتن، می‌بایست تبادل اطلاعات وجود داشته باشد. در طبیعت تبادل اطلاعات مابین جمعیت‌ها از طریق مهاجرت صورت می‌پذیرد. در مهاجرت چند نکته باید مد نظر قرار گیرد: این که مهاجرت کی و چطور، با چه تعداد موجود و از کجا به کجا صورت می‌گیرد. این که مهاجرت از چه مکانی به مکان دیگر رخ می‌دهد توسط توپولوژی جمعیت مشخص می‌شود. یک حالت مدل جزیزهای است. مشابه آن چه در شکل زیر می‌بینید، فلش‌ها مبدأ و مقصد مهاجرت را مشخص می‌کنند.



حالت دیگر، مدل کاملاً متصل^۱ است که در آن مهاجرا مابین هر دو منطقه انجام می‌پذیرد. همچنین به صورت دو طرفه.

مسئله دیگری که در مهاجرت اهمیت جدی دارد، این است که این رخداد هر چند نسل یک بار و به چه تعداد انجام گیرد. و این که چه موجوداتی می‌بایست مهاجرت کنند. باید توجه داشت که مهاجرت با تعداد بالا و در هر نسل،

¹ full connected

عملاً سبب می‌شود جمعیت مورد بررسی تبدیل به یک جمعیت واحد گردد. در مقابل نرخ مهاجرت بیش از اندازه پایین، در عمل معادل مجزا بودن جمعیت‌ها خواهد بود. به بیان دیگر، نرخ مهاجرت بالا، تنوع پایین و سرعت همگرایی بالا را برای سیستم در پی خواهد داشت. برای نرخ مهاجرت پایین، بر عکس این وضعیت رخ می‌دهد. در مجموع می‌توان گفت که نرخ مهاجرت پارامتر بسیار مهمی است. باید به گونه‌ای انتخاب شود که موجود بتواند کارآیی یا عدم کارآیی خود را در جمعیت نشان دهد. به طور متعارف، هر ۲۵ یا ۱۰۰ تا ۱۵۰ نسل بین ۲ تا ۵ موجود مهاجرت داده می‌شوند.

تصمیم‌گیری در خصوص این که کدام موجود یا موجودات می‌باشد مهاجرت کنند، به شرایط مسئله بستگی دارد. در حالتی که بخواهیم هر جمعیت یک قله داشته باشد، یعنی داخل هر جمعیت موجودات حتی المقدور شبیه به یکدیگر باشند، یک راه این است که موجود دارای کمترین شباهت به جمعیت مهاجرت داده شود تا جمعیت خود را بیابد. شباهت می‌باشد. در واقع، اگر موجودات بخواهند در یک قله باشند، شباهت ژنتیکی مد نظر قرار می‌گیرد. در حالت فتوتیپ بر عکس عمل می‌کنیم؛ یعنی موجودات قله‌های مختلف را وارد جمعیت خواهیم کرد. الگوریتم‌های چند هدفی و مبتنی بر همسایگی دو کاربرد دارند: یکی پیدا کردن چند جواب و دیگری انجام پردازش موازی. در حالتی که با وضعیت استاندارد الگوریتم سر و کار داریم و هدف به دست آوردن چند جواب نیست و تنها می‌خواهیم هر بخش از پردازش را به یک پردازنده بدهیم، می‌توانیم بر اساس شباهت فتوتیپی عمل کنیم. اما در حالتی که هدف یافتن چند جواب است، از شباهت ژنتیکی استفاده می‌کنیم.

در پردازش موازی، روش انتخاب می‌تواند روش‌های گفته شده پیشین چون تورنومنت Q باشند، با همان مزایا و معایب گفته شده.

فاصله همسایگی

برای درک بهتر مفهوم فاصله همسایگی مثال زیر را در نظر بگیرید. فرض کنید در یک کشور، اهالی شمالی‌ترین و جنوبی‌ترین شهرها در هر نسل تنها بتوانند با اهالی شهرهای مجاور خود ازدواج کنند. بدیهی است که در این حالت زمان بسیار بیشتری نیاز است تا فردی همزمان اطلاعات ژنتیکی شهرهای شمالی و جنوبی را داشته باشد. در مقابل، اگر از ابتدا فردی در شمالی‌ترین شهر اجازه ازدواج با اهالی تمام شهرها را داشته باشد، این روند بسیار تسريع خواهد شد. در حالت دوم می‌گوییم فاصله همسایگی افزایش یافته است. افزایش فاصله همسایگی سبب کاهش تنوع و افزایش سرعت همگرایی خواهد شد.

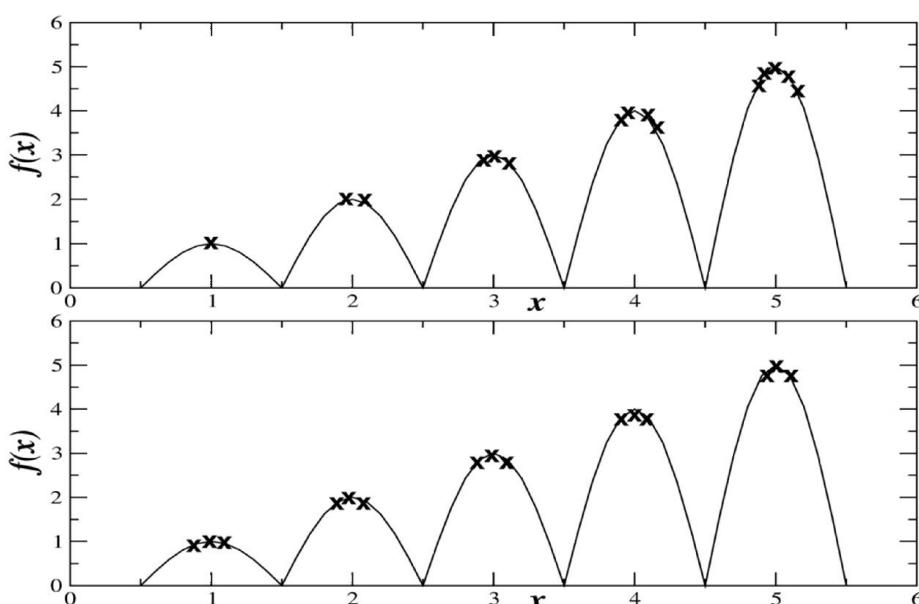
۱-۲-۲۷-۱ نحوه تولید مثل در حالت چند جوابی

در حالت وجود همسایگی لازم است کنترل‌هایی صورت گیرد. مثلاً در حالتی که همسایگی ۱ داریم، موجود تنها می‌تواند با همسایه خود تولید مثل کند. والد اول حتماً می‌باشد به صورت تصادفی از جمعیت انتخاب شود. والد دوم

در همسایگی والد اول و با روش‌هایی که پیش‌تر گفته شد انتخاب می‌شود. نحوه انتخاب به این که هدف پردازش موازی یا حل مسأله چند جوابی است وابسته خواهد بود. در حالت چند جوابی باید کاری کنیم که همسایگی مذکور شبیه هم شوند. لذا والد دوم می‌تواند به صورت کاملاً تصادفی در آن همسایگی انتخاب شود که عموماً چنین است. همچنین می‌تواند بر اساس شباهت با عدم شباهت به والد اول انتخاب شود. در هر حال دو فرزنده تولید می‌شود.

تشریک برازندگی

یکی از روش‌های غیر مستقیم است. در این روش شایستگی‌ها تغییر می‌کنند و می‌توان آن‌ها را عوض کرد. تکامل به دنبال موجوداتی با شایستگی بالاست و تمایل دارد شایستگی را افزایش دهد. فرض کنید سه قله با شایستگی ۴۰، ۲۰ و ۶۰ داشته باشیم. اگر در هر یک از آن‌ها به ترتیب ۱، ۳ و ۲ موجود وجود داشته باشد، از تقسیم شایستگی هر قله بر تعداد موجودات آن، عدد ۲۰ به دست می‌آید. بدیهی است هر چه موجودات تجمع بیش‌تری در یک قله داشته باشند این عدد کاهش خواهد یافت. لذا تکامل موجودات را از تجمع در یک قله دور می‌کند تا شایستگی بالا رود. به این ترتیب، قله دارای شایستگی بالاتر موجودات بیش‌تری خواهد داشت و برعکس. نمونه‌ای از آن چه گفته شد را در شکل زیر مشاهده می‌کنید:



روشن است که تکامل از وضعیت دوم اجتناب می‌کند. دقت داشته باشید که اگر تکامل متعارف را داشتیم تمام موجودات روی یک قله جمع می‌شدند. حال آن که در اینجا می‌توانیم چند قله (جواب) داشته باشیم. شایستگی جدید به صورت زیر تعریف می‌شود:

$$f'_i = f_i / \sum_{j \in C_i} S h_{ij}$$

که f_i شایستگی پیشین و C_i همسایگی i است. همچنین:

$$Sh_{ij} = \begin{cases} 1 - \frac{d_{ij}}{r} & d_{ij} \leq r \\ 0 & d_{ij} > r \end{cases}$$

موجودات خیلی نزدیک ۱ شمرده می‌شوند. برای موجودات دورتر هر دو موجود یکی و با دورتر شدن هر ۳ موجود یکی شمرده می‌شوند. برای هر قله، از r به بعد هم شمرده نمی‌شود. با جمع این مقادیر $\sum_{j \in C_i} Sh_{ij}$ به دست می‌آید که شایستگی قبلی بر آن تقسیم می‌شود. مثلاً اگر داشته باشیم:

$$1 + \left(1 - \frac{1}{2}\right) + \left(1 - \frac{1}{3}\right) = \frac{13}{6}$$

شایستگی بر مقدار $13/6$ تقسیم خواهد شد.

مشکل این الگوریتم در تعیین r است. فرض کنید دو قله در کنار هم داشته باشیم که یکی خیلی تیز و دیگری خیلی پهنه باشد. بدیهی است که انتخاب r واحد برای هر دو منجر به پاسخ مطلوب نخواهد شد. تشریک برازنده استاندارد r ثابت دارد. با این حال الگوریتم‌هایی برای محاسبه r به صورت تطبیقی معرفی شده‌اند.

روش ازدحام

فرض کنید در مسأله چند جوابی، والدین p_1 و p_2 و فرزندان o_1 و o_2 باشند. در روش ازدحام بر اساس شباهت ژنتیکی فرزندان و والدین به شرح زیر عمل می‌کنیم:

```
if  $d(P_1, o_1) + d(p_2, o_2) \leq d(p_1, o_2) + d(p_2, o_1)$ 
  if  $\begin{cases} f(o_1) > f(p_1) \\ f(o_2) > f(p_2) \end{cases}$  then  $p_1 \leftarrow o_1$ 
  else  $p_2 \leftarrow o_2$ 
```

از آن جایی که در این روش، مجموع را در نظر می‌گیریم، خیلی دقیق نیست. دلیل اتخاذ چنین رویکردی این است که نمی‌خواهیم عبارات فوق خیلی دقیق باشند. در روش ازدحام، در هر قله شایستگی را افزایش می‌دهیم و این روند، بدون دور شدن موجودات از والدین خود صورت می‌گیرد. روش ازدحام می‌تواند مانند اینجا همراه با همسایگی مورد استفاده قرار گیرد و یا مستقل از همسایگی به کار گرفته شود.

تفاوت روشهای تشریک برازنده‌گی و ازدحام

در تشریک برازنده‌گی، موجودات متناسب با شایستگی بر روی قله‌ها قرار می‌گیرند اما در ازدحام موجودات فارغ از میزان شایستگی، به تعداد مساوی روی قله‌ها قرار خواهند گرفت. از آن جا که تکامل به نحوی است که در آن نوع

جمعیت حفظ می‌شود و تنها شایستگی افزایش می‌باید، در این روش فرزندان به شیوه گفته شده، جایگزین والدین خود خواهند شد.

۱-۲۸- الگوریتم PSO استاندارد: نسخه تک گروهی

الگوریتم بهینه سازی از دحام ذرات^۱ (PSO) با الهام از زندگی پرندگان و شیوه آنها برای پیدا کردن غذا ارائه شده است. پرندگان به صورت گروهی برای پیدا کردن غذا حرکت می‌کنند و باید به دنبال گروه حرکت می‌کنند. اگر این پرندگان سر گروه نداشته باشند دور ایجاد می‌شود و به منظور اجتناب از این مسئله، میبایست یکی از آنها سر دسته باشد. به این ترتیب، اگر این پرنده غذایی ببیند، مابقی به دنبال آن خواهند رفت. همچنین ممکن است پرنده بر اساس اطلاعات قبلی از یک منبع غذا که در حافظه نگهداری کرده است به سمت منبع مذکور برود. در این صورت پرندگان دیگر نیز به دنبال آن خواهند رفت.

در این حالت با دو تجربه فرد و گروه مواجه هستیم.

هر پرنده یک مکان دارد که با x_i نشان داده می‌شود همچنین بردار سرعتی دارد که جهت و اندازه سرعت را نشان می‌دهد.

$$\text{سرعت به صورت } V = \frac{dX}{dt} \text{ تعریف می‌شود.}$$

در نتیجه

$$V = \frac{X(t+\Delta t) - X(t)}{\Delta t}$$

$$\Delta t = 1 \Rightarrow V = X(t+1) - X(t)$$

$$X(t+1) = v + x(t)$$

p_i : بهترین موقعیتی که فرد i در حافظه دارد.

g : بهترین موقعیت قبلی که گروه در حافظه خود دارد.

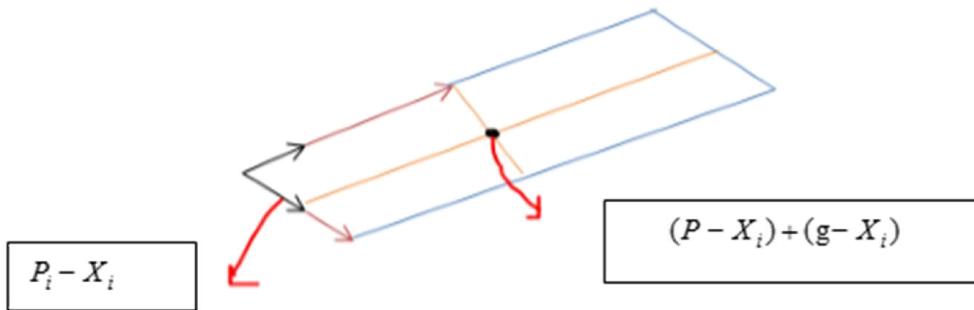
$$V(t+1) = c_1 r_1 (P - X_i) + c_2 r_2 (g - X_i)$$

$$X_i(t+1) = X_i(t) + V_i(t+1)$$

$$c_1, c_2 = 2$$

و c_1 عددی تصادفی بین صفر و یک است.

¹ Particle Swarm Optimization



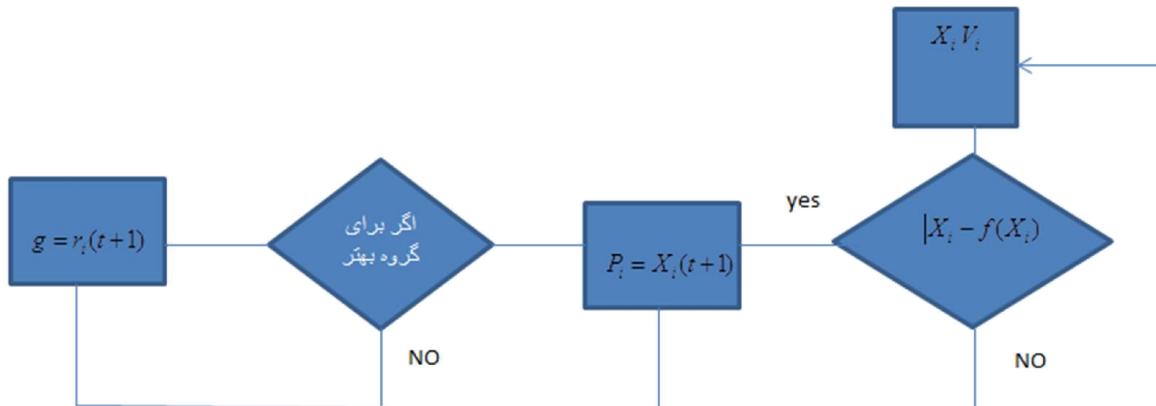
جستجوی محلی، در متوازی الاضلاع نمایش داده شده در شکل و حول بهترین موقعیت شخصی و گروهی انجام می شود.

برای جلوگیری از تغییر جهت ناگهانی ذرات، به رابطه $V(t+1)$ عبارت جدیدی با ضریب کمتر از یک اضافه می شود.

$$V(t+1) = \omega V_i(t) + c_1 r_1 (P_i - X_i) + c_2 r_2 (g - X_i)$$

اگر $\omega > 1$ تغییر جهت اتفاق نمی افتد.

فلوچارت الگوریتم



بهتر شدن فرد لزوماً به معنای بهتر شدن گروه نیست. اما اگر فرد بهتر نشده باشد، حتماً در گروه هم بهتر نیست. به این ترتیب، اگر فردی بهتر شده باشد، نیاز به بررسی گروه خواهد بود. الگوریتم PSO جستجوگر محلی خوبی است با این حال، در تکرارهای ابتدایی جستجوگر عمومی به شمار می آید.

با توجه به این که سرعت های اولیه تصادفی هستند مقدار دهی اولیه تصادفی است. این موضوع که در فرمول سرعت تا چه زمان سرعت تصادفی را خواهیم داشت، قابل محاسبه است.

$$V_i(h) = \omega^n V_i(0)$$

تا زمانی که $V_i(0)$ در $V_i(h)$ تأثیرگذار است، جستجوی عمومی داریم با این حال، زمانی که مقدار آن خیلی کوچک، یعنی در حدود 0.001 شود، عملاً بی تأثیر است.

اگر پیش از اتمام سرعت تصادفی اولیه، داخل متوازی الاضلاع، داخل مینیمم اصلی شویم، مینیمم اصلی پیدا می شود در غیر این صورت دیگر قادر به یافتن آن نخواهیم بود.

در این روش، پس از یافتن بهترین تجربه عمومی گروهی، در اطراف آن جستجوی محلی می کنیم یا مواردی را که پیشتر پیدا کرده بودیم را بهبود میدهیم.

اگر ω نزدیک 1 باشد جستجوی عمومی بیشتر است اما برای همگرایی به ω کوچک نیاز داریم. ω معادل سیگما در ES است و برابر طول گام هاست. در PSO استاندارد، ω ثابت است که کارآیی چندانی ندارد. اگر احتمال موفقیت بالا باشد، ω بزرگ است و برای احتمال موفقیت پایین، کوچک ω . یا باید به صورت خطی تغییر کند و یا اگر مانند قانون یک پنجم موفقیت عمل کنیم می تواند تابعی از موفقیت باشد.

فصل هفتم: کنترل قیدها و محاسبه مقدار بهینه پارامترها

۱- مقدمه

مسائل بهینه سازی به دو دسته مقيید^۱ و نامقيید تقسيم می‌شوند. آن چه تا به حال بحث شد، به مسائل نامقيید اختصاص داشت. در مسائل مقيید با فرم کلی زير مواجه هستيم:

$$\text{minimize } f(x) \quad \text{subject to:} \begin{cases} h_i(x) = 0 \\ g_i(x) \leq 0 \end{cases}$$

كه دسته اول قيدهای تساوی و دسته دوم قيدهای نامساوی نام دارند. بدیهی است که همین مسأله را می‌توان برای حالت یافتن بیشینه نیز داشت.

مجموعه قيدها ناحیه‌ای را در فضای جيتجو مشخص می‌کنند که به آن ناحیه عملی^۲ گفته می‌شود. بدیهی است که پاسخ به دست آمده تنها در صورتی قابل قبول است که در ناحیه عملی قرار گیرد؛ يعني قيدها را برآورده کند. برای لحاظ کردن اثر قيدها چندين راه وجود دارد:

۲- کنترل قیدها

ساده‌ترین راه حذف موجوداتی است که در قيدها صادق نیستند که البته روش مناسبی نیست. به اين دليل که تنوع را کاهش داده و تکامل را به مخاطره خواهد انداخت.

راه ديگر در نظر گرفتن جريمه برای تخطی از قيدهاست. به اين صورت که برای قيدهای تساوی، قدر مطلق مقدار به دست آمده را در وزني ضرب می‌کنيم و برای قيدهای نامساوی، در صورت مثبت بودن مقدار به دست آمده آن را در وزني ضرب می‌کنيم. مجموع وزن دار به دست آمده از اين مقادير را به عنوان جريمه کلی به تابع هدف اضافه می‌کنيم. در صورت اختصاص وزن غير يك، پيدا کردن وزن يكى از مشكلات خواهد بود. با اين حال در نظر گرفتن وزن ۱، يا وزن واحد برای تمام جريمه‌ها، شيوهای متعارف محسوب می‌شود.

راه ديگر در نظر گرفتن هر يك از قيدها به عنوان يك هدف است. در اين صورت با يك مسأله چند هدفه مواجه خواهيم شد که هدف آن بيشينه يا کيمنه کردن تابع هدف و به علاوه، کمينه کردن مقدار قدر مطلق توابع مربوط به قيدهاست.

¹ Constraint
² feasible

۱-۳۱-۱ مقدار بهینه پارامترها

برای به دست آوردن مقدار بهینه پارامترهای مسأله نیز چند راه وجود دارد.

روش‌های غیربرخط

راه اول محاسبه غیربرخط^۱ است. فرض کنیم هدف محاسبه مقادیر بهینه نرخ جهش و تعداد جمعیت است. مثلاً برای $n = 10, 20, 30$ و $P_M = 0, 0.1, \dots, 1$, ^۲ ۳۳ حالت داریم. روشن است که اگر هر آزمایش ۳۰ بار انجام شود و در هر مورد برای ۳۰ بنج مارک^۳ بررسی شود با حجم عظیمی از محاسبات رو به رو خواهیم شد. لذا یکی از مشکلات این رویکرد، محاسبت زیاد است، به ویژه در شرایطی که اندازه قدم کوچک باشد. مشکل جدی‌تر این روش، دینامیک بودن مقادیر بهینه پارامترها است که در هر نسل پارامتر متفاوت خواهد بود. به علاوه، در یک نسل نیز برای موجودات مختلف این تفاوت را خواهیم داشت.

روش‌های برخط

با توجه به اشکال عمدہای، که روش‌های غیربرخط دارند به سراغ روش‌های برخط^۴ می‌رویم که به دسته کلی تقسیم می‌شوند: قطعی، تطبیقی و خودتطبیقی.

۱-۲-۳۱-۱ روش قطعی

در این شیوه، رابطه‌ای که نسبت به زمان قطعی است به دست می‌آید. مثل رسیدن از ۰,۹ به ۰ در PSO. اشکال این روش نیاز به داشتن اطلاعات از رفتار بنج مارک است در حالی که عملاً از آن بی اطلاع هستیم. به این معنی که نمی‌دانیم رفتار خطی است، نمایی است یا ...

۱-۲-۳۱-۲ روش تطبیقی

مشابه ۱/۵ موفقیت است. یک فیدیک پیدا می‌کنیم. اگر میزان موفقیت‌ها برابر ۱/۵ بود که کاری نمی‌کنیم. در غیر این صورت ω را تغییر خواهیم داد. در صورتی که بخواهیم ω را تابعی از زمان در نظر بگیریم، در ابتدا که دور از جواب هستیم می‌بایست کوچک باشد. در واقع، اهمیت کمتری به قیدها می‌دهیم. در حالی که در ادامه با افزایش اهمیت قیدها، افزایش خواهد یافت.

معادل این حرف با قانون ۱/۵ موفقیت یا قوانین مشابه آن به این صورت خواهد بود که اگر k بهترین موجود عملی بودند، جرمیه و به تبع آن ω کاهش خواهد یافت. اگر تمامی k بهترین موجود غیر عملی بودند جرمیه و طبعاً ω

¹ offline

² benchmark

³ online

افزایش خواهد یافت. در حالتی که بخشی از بهترین موجودات مورد بررسی عملی و بخشی از آن‌ها غیرعملی باشند، مقدار ω بدون تغییر باقی خواهد ماند.

۱-۳-۲-۳۱- خود تطبیقی

در این شیوه مشابه آن چه در ES دیدیم، پارامتر بهینه توسط تکامل به دست می‌آید. در این روش نیز می‌توان ω را به عنوان ژن در کروموزوم قرار داد و به وسیله تکامل مقدار بهینه آن را به دست آورد.