الگوریتم ژنتیک

- در دنیای واقعی، مسائل خیلی پیچیده‌ای وجود دارد که با روش‌های معمول بهینه سازی قابل حل نیستند. از دهه ۱۹۶۰ علاقه به تقلید از دنیای اطراف و موجودات زنده در حل چنین مسائل پیچیده‌ی بهینه سازی، زیاد شد.

- شبیه سازی مراحل تکامل، منتج به تکنیک‌های بهینه سازی تصادفی به نام **الگوریتم‌های تکاملی** شد که معمولا در مسائل سخت دنیای واقعی بهتر از روش‌های معمول بهینه سازی عمل می‌کنند.

الگوریتم ژنتیک

- هم اکنون، پنج شاخه اصلی در این زمینه وجود دارد، **استراتژی‌های تکاملی، برنامه نویسی تکاملی، الگوریتم‌های ژنتیک، برنامه نویسی ژنتیک و تکامل تفاضلی** که معمول‌ترین این روش‌ها، همان الگوریتم ژنتیک است.

- الگوریتم ژنتیک روش یادگیری بر پایه تکامل بیولوژیک است.
- این روش در سال ۱۹۷۰توسط John Holland معرفی شد.

زیر شاخه‌ها

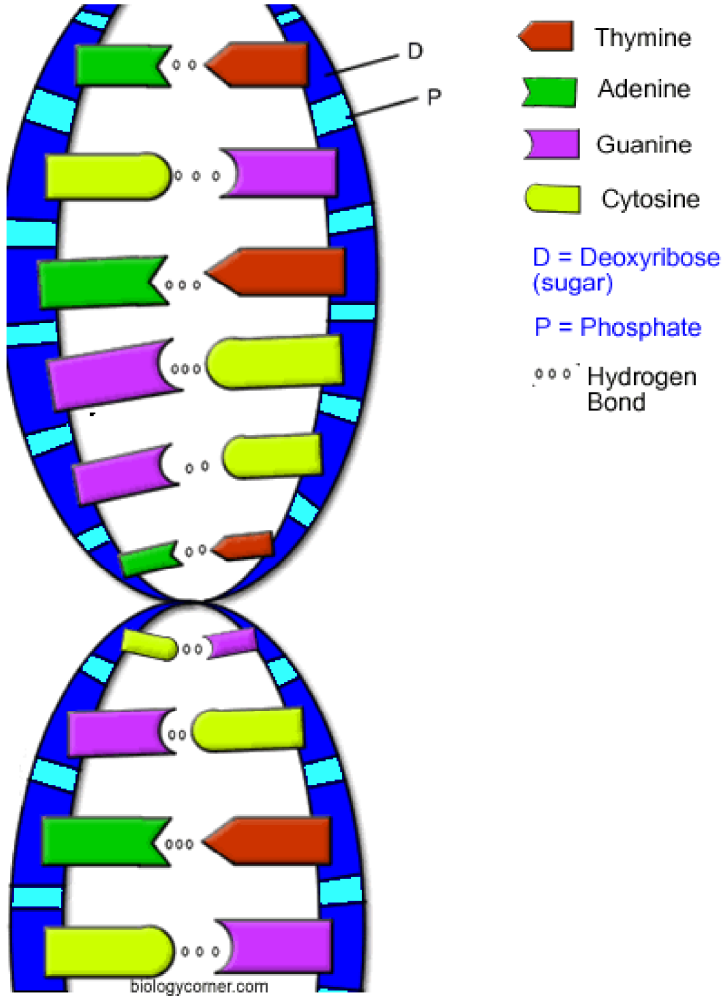
Genetic Algorithms (GAs)

در این روش راه حل یک مساله در ساده ترین حالت به صورت یک Bit string نشان داده می‌شود.

Genetic Programming (GP)

این روش به تولید Expression trees می‌پردازد. بدین ترتیب می‌توان برنامه‌هایی ساخت که قابل اجرا باشند.

الگوریتم ژنتیک



Deoxyribonucleic acid (DNA)

ایده کلی

- یک GA برای حل یک مساله مجموعه بسیار بزرگی از راه حل‌های ممکن را تولید می‌کند.
- هر یک از این راه حل‌ها با استفاده از یک «**تابع برازندگی**» مورد ارزیابی قرار می‌گیرد.
- آنگاه تعدادی از بهترین راه حل‌ها باعث تولید راه حل‌های جدیدی می‌شوند که این‌کار باعث تکامل راه حل‌ها خواهد شد.
- بدین ترتیب فضای جستجو در جهتی تکامل پیدا می‌کند که به راه حل مطلوب برسد.
- در صورت انتخاب صحیح پارامترها، این روش می‌تواند بسیار موثر عمل نماید.

فضای فرضیه

- الگوریتم ژنتیک در عوض جستجوی فرضیه‌های General to Specific و یا Simple to Complex فرضیه‌های جدید را با تغییر و ترکیب متوالی اجزا بهترین فرضیه‌های موجود به دست می‌آورد.

- در هر مرحله مجموعه‌ای از فرضیه‌ها که **جمعیت (Population)** نامیده می‌شوند از طریق جایگزینی بخشی از جمعیت فعلی با فرزندانی که از بهترین فرضیه‌های موجود حاصل شده‌اند به دست می‌آید.

ویژگی‌ها

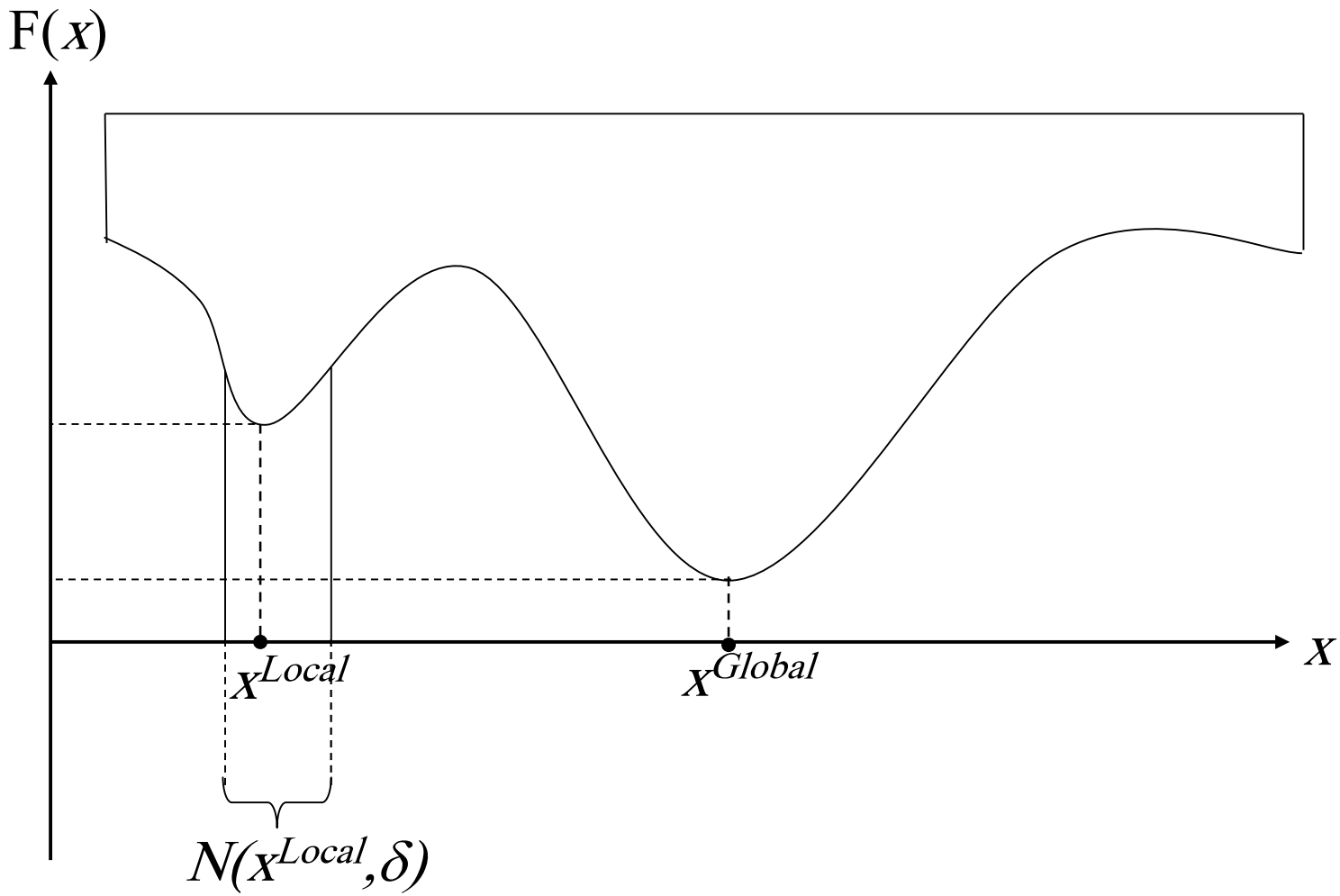
الگوریتم ژنتیک در مسائلی که فضای جستجوی بزرگی داشته باشند می‌تواند به کار گرفته شود.

- همچنین در مسایلی با فضای فرضیه پیچیده که تاثیر اجزای آن در فرضیه کلی ناشناخته باشند می‌توان از GA برای جستجو استفاده نمود.
- این روش برای **بهینه سازی گسسته** بسیار مورد استفاده قرار می‌گیرد.
- الگوریتم ژنتیک را می‌توان به راحتی به صورت موازی اجرا نمود از این رو می‌توان کامپیوترهای ارزان قیمت‌تری را به صورت موازی مورد استفاده قرار داد.

ویژگی‌ها

- امکان افتادن این الگوریتم در تله کمینه محلی کمتر از سایر روش‌ها می‌باشد.
- با این حال این روش‌ها از لحاظ محاسباتی پرهزینه هستند.

بهینه سراسری و محلی



بهینه سراسری و محلی

تعریف بهینه محلی (Local Optimum):

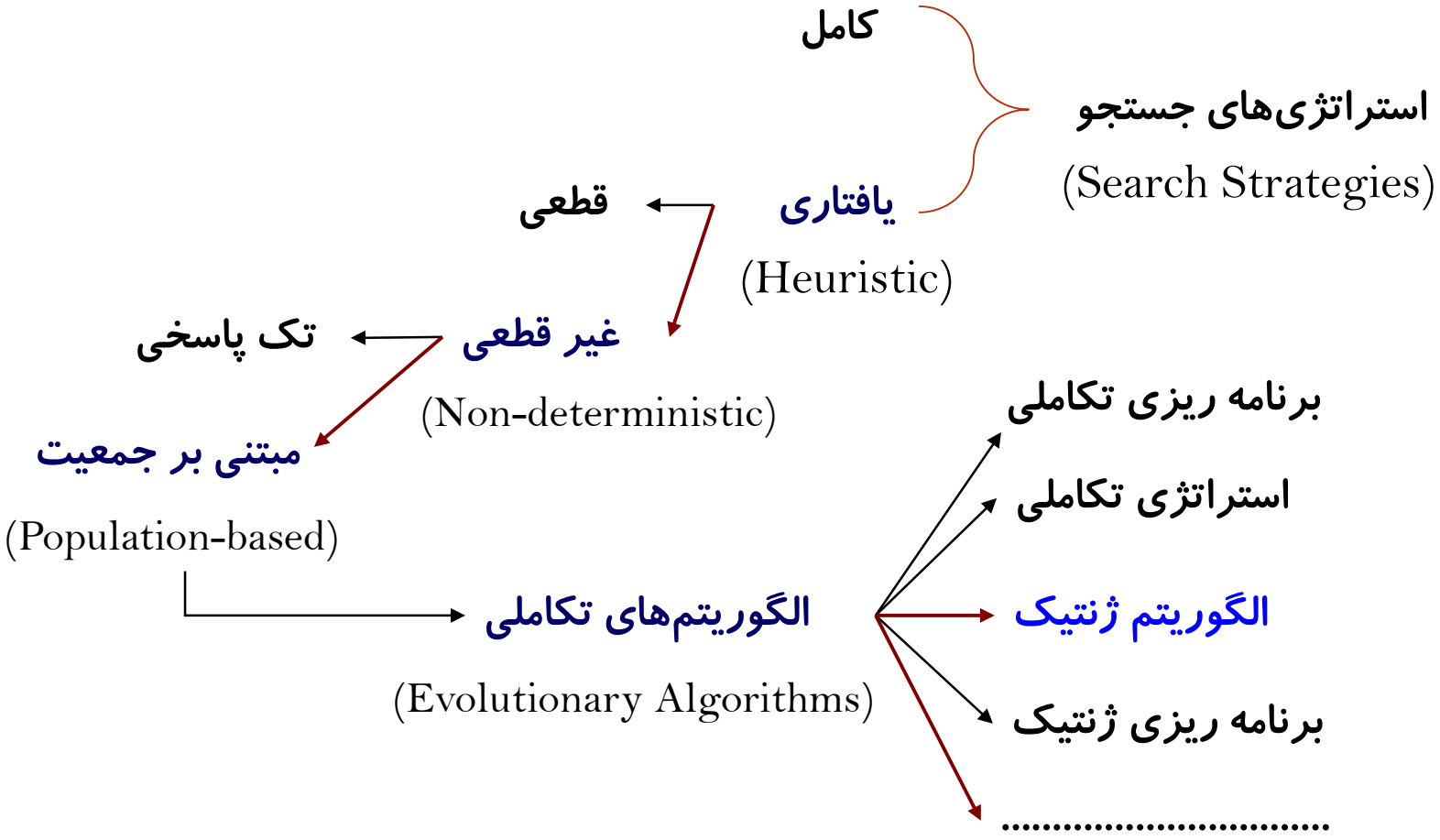
هرگاه یک همسایگی مانند $N(x,\delta)$ یافت شود به گونه ای که حل x از هر حل موجود در آن همسایگی بهتر باشد، به x در اصطلاح بهینه محلی گفته می‌شود.

کاربر دها

- کاربرد الگوریتم‌های ژنتیک بسیار زیاد می‌باشد:

- Optimization,
- Automatic programming,
- Machine learning,
- Economics,
- Operations research,
- Ecology,
- Studies of evolution and learning,
- Social systems,
- ….

الگوریتم‌های ژنتیک



الگوریتم‌های ژنتیک

روش متداول پیاده سازی الگوریتم ژنتیک بدین ترتیب است که:

- مجموعه‌ای از فرضیه‌ها که **Population** نامیده می‌شود تولید و به طور متناوب با فرضیه‌های جدیدی جایگزین می‌شود.
- در هر بار تکرار تمامی فرضیه‌ها با استفاده از یک (چند) تابع برازندگی (تطابق) یا **Fitness Function** مورد ارزیابی قرار می‌گیرند.
- آنگاه تعدادی از بهترین فرضیه‌ها با استفاده از یک تابع احتمال انتخاب شده و جمعیت جدید را تشکیل می‌دهند.

الگوریتم‌های ژنتیک

- تعدادی از این فرضیه‌های انتخاب شده به همان صورت مورد استفاده واقع شده و مابقی با استفاده از عملگرهای ژنتیک نظیر **Crossover** (آمیزش) و **Mutation** (جهش) برای تولید فرزندان (**Offspring**) به کار می‌روند.

پیاده سازی الگوریتم ژنتیک

۱- طراحی ساختار کرموزوم یا نحوه نمایش حل مسأله (Representation).

۲- استراتژی تولید جمعیت اولیه (Seeding).

۳- استراتژی انتخاب جمعیت والد (Mating Pool) یا مکانیزم انتخاب.

۴- طراحی یا انتخاب عملگرهای ژنتیک متناسب با ساختار کروموزوم و قیود مسأله (Operators).

۵- نحوه محاسبه برازندگی یا کیفیت کروموزوم‌ها (Fitness).

۶- تعیین معیار توقف (Stop Criteria).

پارامترهای الگوریتم ژنتیک

کروموزوم:

در الگوریتم ژنتیک، هر کروموزوم نشان دهنده یک نقطه در فضای جستجو و یک راه حل ممکن برای مساله مورد نظر است.

هر کروموزوم از تعداد ثابتی ژن تشکیل می‌شود. برای نمایش کروموزوم‌ها، در ساده ترین حالت از کدگذاری دودویی به صورت رشته‌های بیتی و یا در حالت کلی از اعداد حقیقی استفاده می‌شود.

پارامترهای الگوریتم ژنتیک

جمعیت:

مجموعه‌ای از کروموزوم‌ها یک جمعیت را تشکیل می‌دهند و در طی فرآیند تکامل با تاثیر عملگرهای ژنتیک و انتخاب بر روی هر جمعیت، جمعیت جدیدی با همان تعداد کروموزوم تشکیل می‌شود.

پارامترهای الگوریتم ژنتیک

تابع برازندگی:

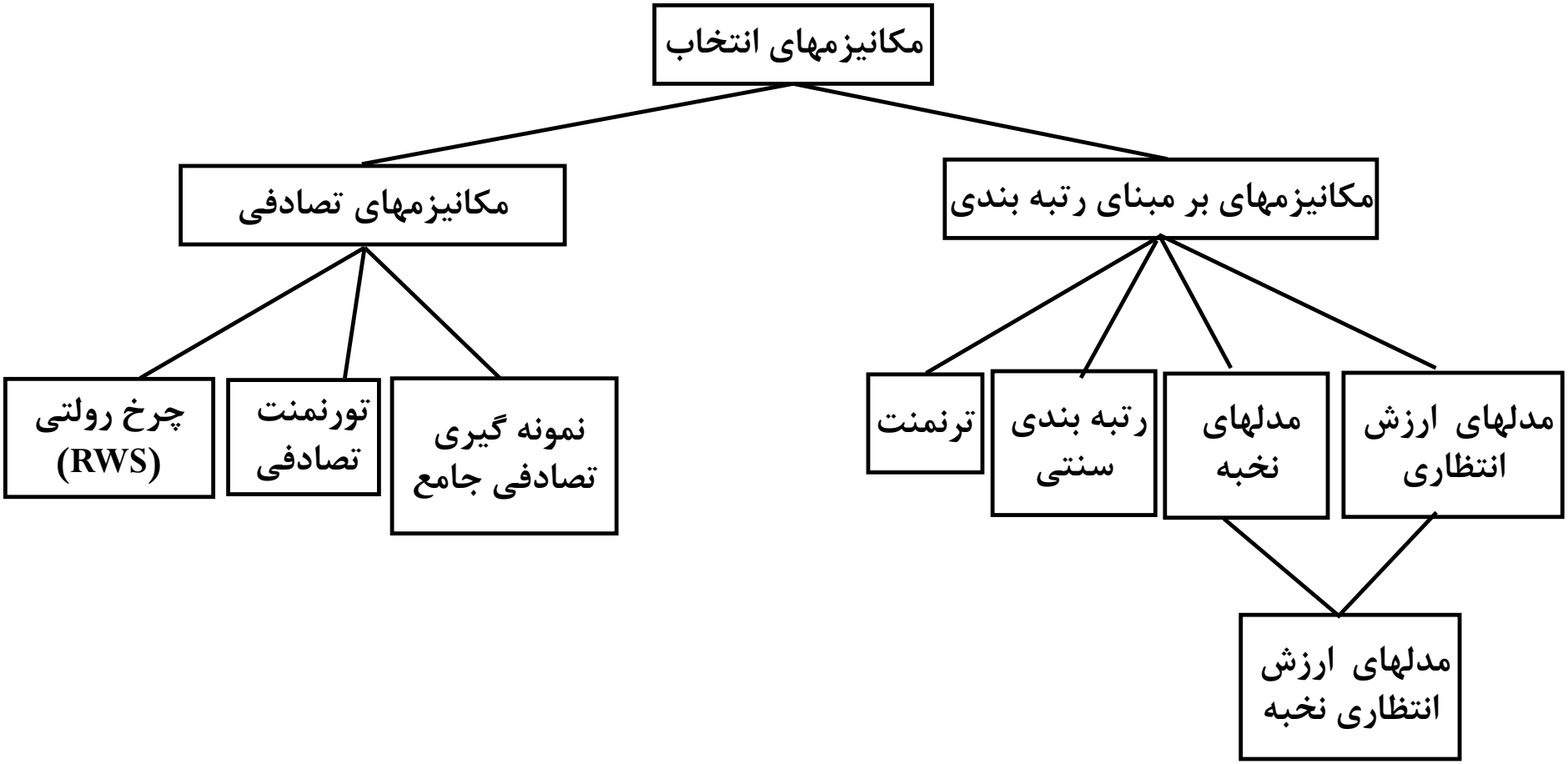
به منظور حل هر مساله با استفاده از الگوریتم ژنتیک، ابتدا باید یک تابع برازندگی برای آن مساله تعریف شود. این تابع برای هر کروموزوم مقدار عددی را باز می‌گرداند که نشان دهنده شایستگی یا توانایی فردی آن کروموزوم می‌باشد.

عملگرهای الگوریتم ژنتیک

عملگر انتخاب:

این عملگر از بین کروموزوم‌های موجود در یک جمعیت، تعدادی کروموزوم را برای تولید مثل انتخاب می‌کند. کروموزوم‌های برازنده‌تر شانس بیشتری دارند تا برای تولید مثل انتخاب شوند.

عملگرهای الگوریتم ژنتیک



عملگرهای الگوریتم ژنتیک

Elitist

مناسب‌ترین عضو هر اجتماع انتخاب می‌شود.

Roulette

یک روش انتخاب است که در آن عنصری که برازندگی بهتری داشته باشد، شانس بیشتی برای انتخاب دارد. در واقع به نسبت عدد برازندگی برای هر عنصر، یک احتمال تجمعی نسبت می‌دهیم و با این احتمال است که شانس انتخاب هر عنصر تعیین می‌شود.

عملگرهای الگوریتم ژنتیک

Scaling

این روش زمانی کاربرد دارد که مجموعه پاسخ‌ها دارای عناصری باشد که عدد برازش بزرگی دارند و فقط تفاوت‌های کوچکی آن‌ها را از هم تفکیک می‌کند.

Tournament

یک زیر مجموعه از کروموزوم‌های یک جامعه انتخاب می‌شوند و اعضای آن مجموعه با هم رقابت می‌کنند و سرانجام فقط یک (یا چند) کروموزوم از هر زیر گروه برای تولید انتخاب می‌شوند.

عملگرهای الگوریتم ژنتیک

عملگر آمیزش:

عملگر آمیزش بر روی یک زوج کروموزوم از نسل مولد عمل کرده و یک زوج کروموزوم جدید تولید می‌کند. عملگرهای آمیزش متعددی از قبیل آمیزش تک نقطه‌ای، چند نقطه‌ای و حقیقی وجود دارند که با توجه به خصوصیات مسایل مختلف مورد استفاده قرار می‌گیرند.

عملگرهای الگوریتم ژنتیک

عملگر جهش:

پس از انجام عمل آمیزش، عملگر جهش بر روی کروموزوم‌ها اعمال می‌شود. این عملگر یک ژن از یک کروموزوم را به طور تصادفی انتخاب نموده و سپس محتوای آن ژن را تغییر می‌دهد.

اگر ژن از جنس اعداد دودویی باشد، آن را به وارونش تبدیل می‌کند و چنانچه متعلق به یک مجموعه حقیقی باشد، مقداری جدید و یا عنصر دیگری از آن مجموعه را به جای آن ژن قرار می‌دهد.

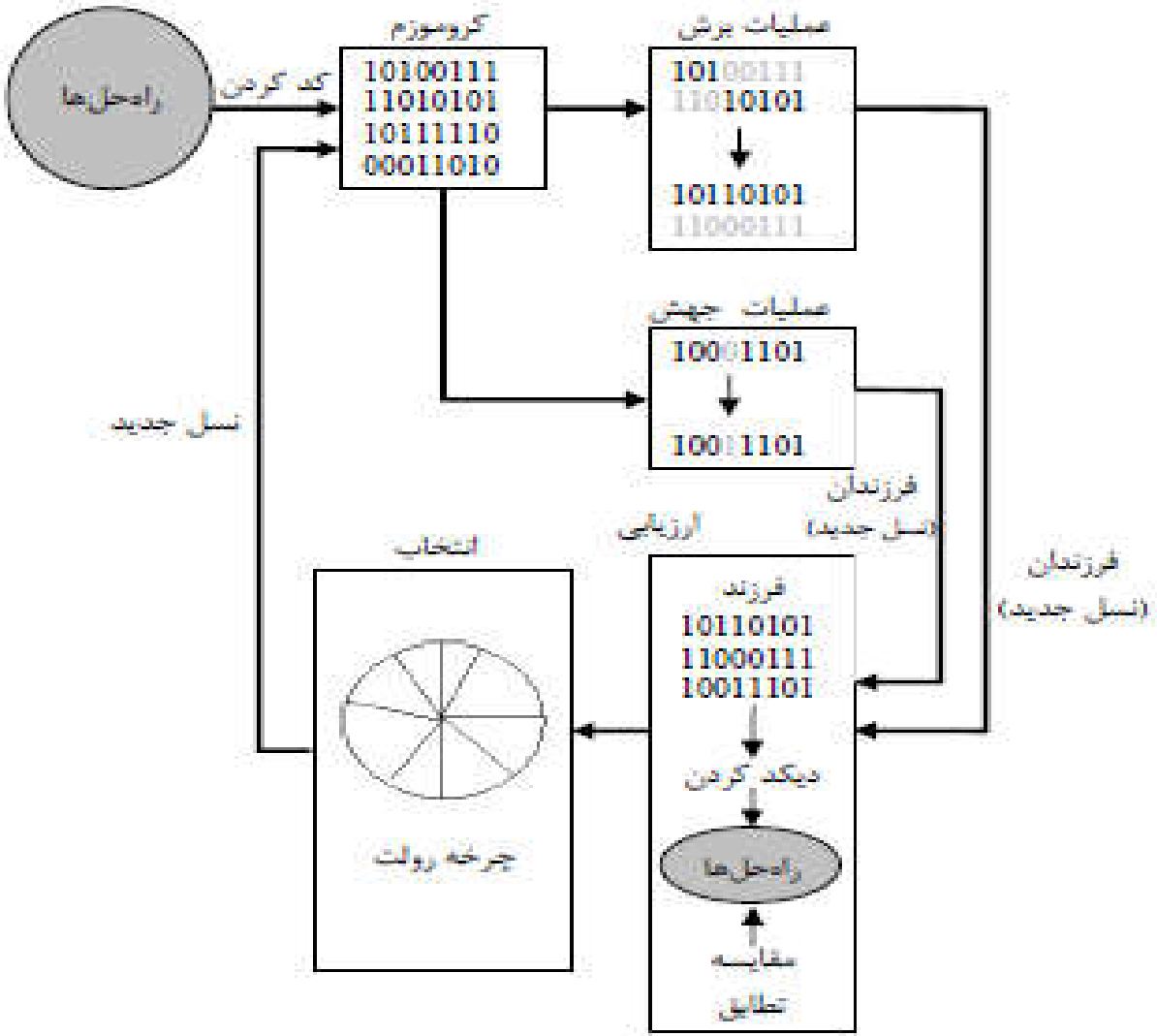
الگوریتم

- Initialize :جمعیت را با تعداد p فرضیه بطور تصادفی مقدار دهی اولیه کنید.
- Evaluate :برای هر فرضیه h در p مقدار تابع Fitness(h) را محاسبه نمایید.
- تا زمانیکه شرط توقف محقق نشده است یک جمعیت جدیدبا استفاده از عملگرهای آمیزش و جهش ایجاد کنید.
- فرضیه ای که دارای بیشترین (کمترین) مقدار Fitness است را برگردانید.

الگوریتم

begin
  t = 0
  initialize P(t)
  evaluate P(t)
while (not termination condition) **do**
  recombine P(t) to yield C(t)
  evaluate C(t)
  select P(t+1) from P(t) and C(t)
  t = t+1
end
end

الگوریتم


راه‌حل‌ها
کد کردن
کروموزم
10100111
11010101
10111110
00011010
عملیات برش
10110101
11000111
عملیات جهش
10011101
نسل جدید
فرزندان
(نسل جدید)
ارزیابی
انتخاب
فرزند
10110101
11000111
10011101
دیکد کردن
راه‌حل‌ها
مقایسه
تطابق
چرخه رولت
فرزندان
(نسل جدید)

الگوریتم

در اصل دو دسته عملگر مختلف در هر الگوریتم ژنتیک قابل تعریف است:

- **عملگرهای ژنتیکی** شامل عملگرهای آمیزش و جهش،
- **عملگر تکامل** که از آن به عنوان عملگر انتخاب یاد می‌شود.

عملگرهای ژنتیکی، از فرآیند ارث بری ژن‌ها به منظور تولید فرزندان در هر نسل پیروی می‌کنند و عملگر تکامل از فرآیند تکامل داروین به منظور تولید جمعیت از نسلی به نسل دیگر تقلید می‌کند.

عملگر آمیزش

همانطور که اشاره شد در هر الگوریتم ژنتیک دو عملگر اصلی به نام‌های آمیزش و جهش وجود دارد.

عملگر آمیزش، اصلی ترین عملگر موجود در الگوریتم ژنتیک است که بر روی دو کروموزوم عمل کرده و فرزندان را با ترکیب خصوصیات دو کروموزوم ایجاد می‌کند.

نکته مهم در هر الگوریتم ژنتیک وابستگی آن به تعریف نوع و نسبت عملگر آمیزش می‌باشد که بر روی کارایی تاثیر بسیار زیادی دارد.

نرخ آمیزش

- نرخ آمیزش که معمولا آن را با Pc نمایش می‌دهند، به عنوان نسبت تعداد فرزندان تولید شده در هر نسل، به اندازه کل جمعیت (که معمولا با PopSize نمایش داده می‌شود) شناخته می‌شود.
- بر این اساس تعداد مورد انتظار Pc×PopSize کروموزوم دستخوش عملیات آمیزش می‌شود.
- نرخ آمیزش بزرگتر، اجازه جستجوی بیشتری را در فضای پاسخ می‌دهد و احتمال افتادن در بهینه محلی را کاهش می‌دهد. اما اگر این نرخ خیلی زیاد باشد، باعث هدر رفتن زیاد زمان در جستجوی مناطقی می‌شود که امید پیدا کردن پاسخ در آن کم است.

عملگر جهش

عملگر جهش، یک عملگر پس زمینه است که در کروموزوم‌های مختلف تغیرات تصادفی تولید می‌کند.

یک راه ساده برای انجام عمل جهش، تغییر یک یا چند ژن است. در الگوریتم‌های ژنتیکی عمل جهش نقش تعیین کننده‌ای ایفا می‌کند، نظیر:

- جانشین کردن ژن‌های گم شده جمعیت در طی انجام عملگر انتخاب
- تولید ژن‌هایی که در جمعیت اولیه حاضر نیستند

نرخ جهش

- نرخ جهش که آن را معمولا با Pm نمایش می‌دهند، به عنوان درصد تعداد کل ژن‌های موجود در یک جمعیت برای عمل جهش است. این درصد، میزان ژن‌های تولید شده برای جمعیت را کنترل می‌کند.
- اگر این نسبت خیلی کم باشد، ژن‌های زیادی که ممکن بود مفید باشند، هیچ‌گاه تولید نمی‌شوند و اگر خیلی زیاد باشد، آشفتگی زیادی در جمعیت تولید شده و فرزندان شباهت کمی با والدین خود دارند و الگوریتم قابلیت آموزش از تاریخچه جستجو را از دست می‌دهد.

مقایسه

- الگوریتم ژنتیک، با مجموعه کد شده پاسخ کار می‌کند نه با خود پاسخ‌ها.
- الگوریتم ژنتیک، با جمعیتی از راه حل‌ها کار می‌کند نه با یک راه حل.
- الگوریتم ژنتیک، از اطلاعات نتیجه نهایی استفاده می‌کند (تابع تطابق) نه از مشتق‌گیری یا دانش کمکی دیگر.
- الگوریتم ژنتیک، از قوانین انتقال احتمالی استفاده می‌کند نه از قوانین قطعی.

اکتشاف و بهره برداری

جستجو یکی از روش‌های بسیار عمومی حل مساله می‌باشد که برای مسائل یک توالی از کارهایی که به پاسخ ختم می‌شود را ارائه می‌دهد. جستجو به دو گونه انجام می شود:

- استراتژی کورکورانه
- استراتژی یافتاری

استراتژی جستجوی کورکورانه، از هیچ اطلاعاتی در مورد دامنه مساله استفاده نمی‌کند اما استراتژی جستجوی یافتاری، از اطلاعات اضافی برای هدایت جستجو در امتداد جهت بهترین جستجو استفاده می‌کند.

اکتشاف و بهره برداری

دو موضوع مهم در استراتژی جستجو وجود دارد:

- اکتشاف فضای جستجو
- بهره برداری بهترین پاسخ

برای این منظور یک مقایسه بین جستجوی تپه نوردی، جستجوی تصادفی و جستجوی ژنتیک انجام می‌دهیم.

جستجوی تپه نوردی، یک مثال از استراتژی است که بهترین پاسخ را بهره برداری می‌کند، در حالی که اکتشاف مناطقی از فضای جستجو که امید یافتن پاسخ در آنها وجود دارد را نادیده می‌گیرد.

اکتشاف و بهره برداری

جستجوی تصادفی، نمونه‌ای از استراتژی است که فضای حالت را اکتشاف می‌کند، در حالی که بهره برداری مناطقی از فضای جستجو که دارای پاسخ مناسب هستند را نادیده می‌گیرد.

الگوریتم‌های ژنتیک، دسته‌ای از روش‌های جستجو همه منظوره هستند که المان‌های جستجوی جهت یافته و تصادفی را ترکیب می‌کنند تا بتواند یک تعادل قابل توجه، بین بهره برداری و اکتشاف فضای جستجو برقرار سازند.

اکتشاف و بهره برداری

در ابتدای رویه جستجوی ژنتیکی، جمعیت تصادفی تولید شده دارای پراکندگی زیادی است و عملگر آمیزش، تمایل به جستجوی گسترده‌ای برای اکتشاف تمام فضای جستجو دارد.

هنگامی که بهترین پاسخ‌ها به دست آمد، عملگر آمیزش اطراف پاسخ‌های به دست آمده به انجام عمل اکتشاف می‌پردازد. به عبارت دیگر، نوع جستجوهایی که عملگر آمیزش انجام می‌دهد، (اکتشافی یا بهره بردارانه) وابسته به محیطی که سیستم ژنتیک در آن قرار دارد (پراکندگی جمعیت) است، و نه صرفا با خود عملگر.

اکتشاف و بهره برداری

علاوه بر این، عملگرهای ساده ژنتیک، به عنوان روش‌های جستجوی همه منظوره طراحی شده‌اند.

این عملگرها، اصولا یک جستجوی کورکورانه انجام می‌دهند و نمی‌توانند حصول فرزندان بهتر را ضمانت کنند.

جستجو مبتنی بر جمعیت

معمولا هر الگوریتمی برای حل مسائل بهینه سازی، شامل یک توالی از کارهای محاسباتی است که به صورت مجانبی به راه حل بهینه همگرا می‌شود.

بسیاری از روش‌های بهینه سازی کلاسیک، یک توالی قطعی از محاسبات، بر اساس مشتق‌گیری از تابع هدف را تولید می‌کنند.

جستجو مبتنی بر جمعیت

این روش‌ها بر روی یک نقطه از فضای جستجو انجام می‌شوند. این نقطه سپس در امتداد جهت کاهش/افزایش شیب، در طی تکرارها بهبود می‌یابد.

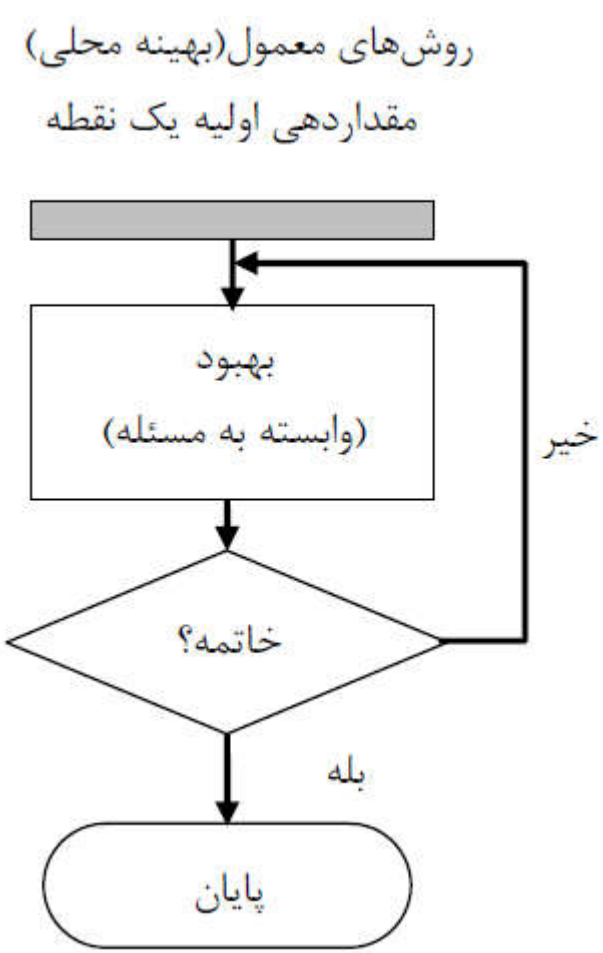
این رهیافت نقطه به نقطه، خطر افتادن به نقطه بهینه محلی را داراست. الگوریتم ژنتیک، یک جستجوی چند بعدی را به وسیله نگهداری یک جمعیت پاسخ بالقوه انجام می‌دهد.

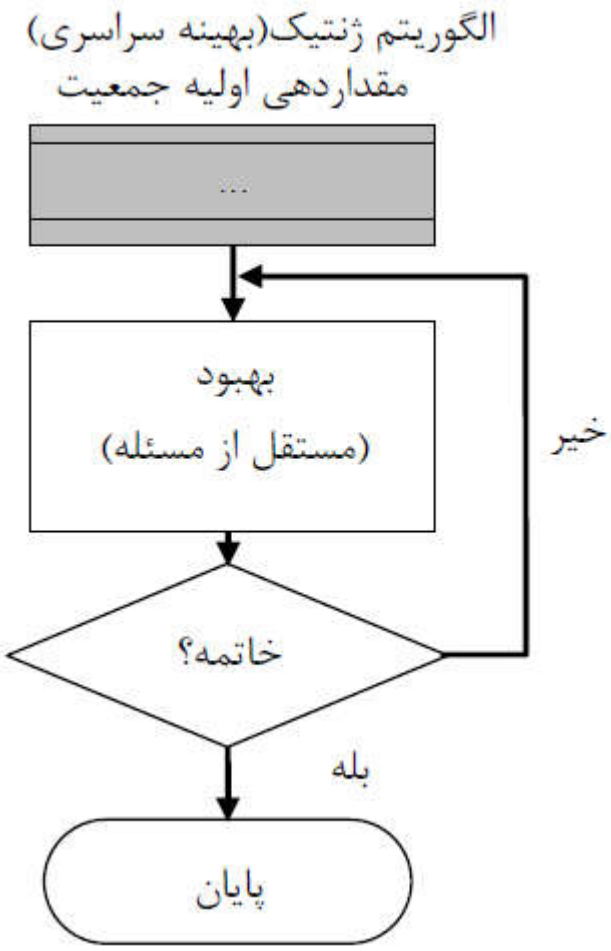
جستجو مبتنی بر جمعیت

رهیافت مذکور تلاش می‌کند که جستجو از افتادن در تله نقاط بهینه محلی فرار کند. در حقیقت در این روش جمعیت دستخوش یک تکامل شبیه سازی شده است.

در هر نسل، پاسخ‌های نسبتا خوب حفظ می‌شوند، در حالی که پاسخ‌های نسبتا بد از بین می‌روند.

جستجو مبتنی بر جمعیت





نمایش فرضیه‌ها

در الگوریتم ژنتیک معمولا فرضیه‌ها به صورت رشته‌ای از بیت‌ها نشان داده می‌شوند تا اعمال عملگرهای ژنتیک برروی آنها ساده‌تر باشد. هر چند در برخی مسائل به منظور جلوگیری از پیچیدگی کارکرد، از کدینگ حقیقی استفاده می‌شود.

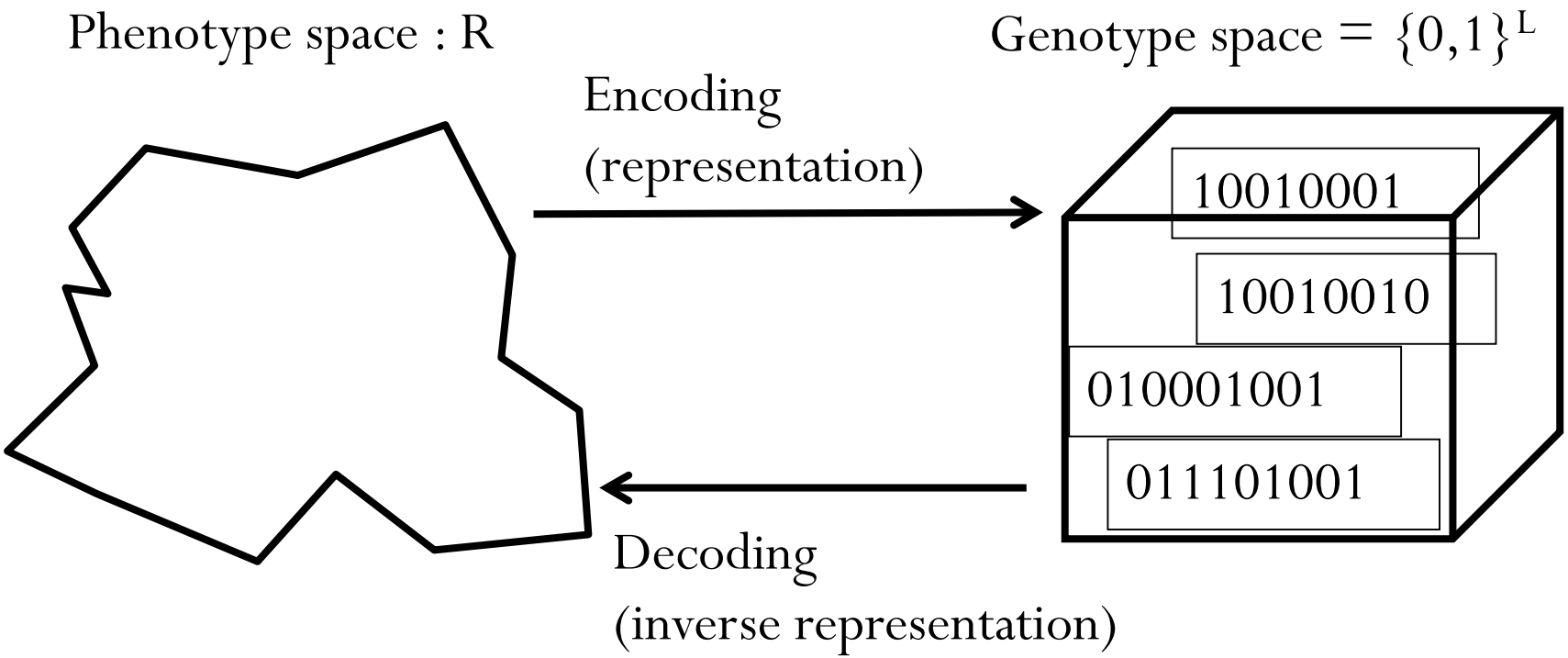
- Phenotype :به مقادیر یا راه حل‌های واقعی گفته می‌شود.
- Genotype :به مقادیر کد شده یا کروموزم‌ها گفته می‌شود که مورد استفاده GA قرار می‌گیرند.
- باید راهی برای تبدیل این دو نحوه نمایش به یکدیگر معرفی شود.

نمایش فرضیه‌ها

Table 1.1 : Explanation of Genetic Algorithm Terms

Genetic Algorithms	Explanation
Chromosome (String , Individual)	Solution (Coding)
Genes (bits)	Part of solution
Locus	Position of gene
Alleles	Values of gene
Phenotype	Decoded solution
Genotype	Encoded solution

نمایش فرضیه‌ها



نمایش فرضیه‌ها :ملاحظات

ممکن است ترکیب بعضی از ژن‌ها منجر به فرضیه‌های بی معنی شود. برای پرهیز از چنین وضعیتی می‌توان:

- از روش کدینگ دیگری استفاده نمود.
- عملگرهای ژنتیک را طوری تعیین نمود که چنین حالت‌هایی را حذف نمایند.
- به این فرضیه‌ها مقدار تطابق خیلی کمی نسبت داه شود.

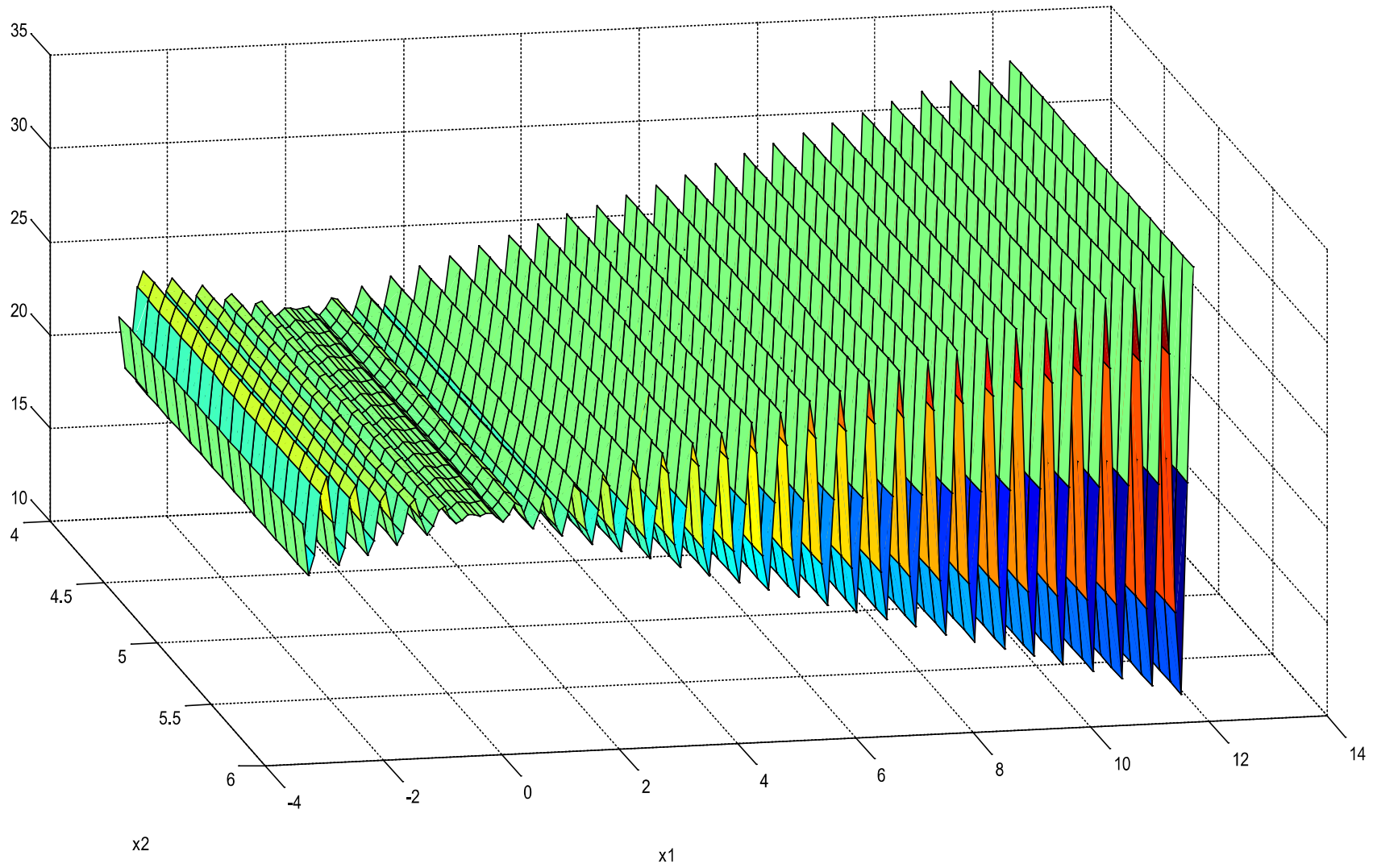
مثال ۱ : بهینه سازی

در این بخش به بررسی حل یک مساله با استفاده از نگارش ساده الگوریتم ژنتیک می‌پردازیم، هدف در حل این مساله انجام عمل بهینه سازی بر روی تابع زیر است:

$$\max f(x_1, x_2) = 21.5 + x_1 \sin(4\pi x_1) + x_2 \sin(20\pi x_2)$$

$$-3.0 \le x_1 \le 12.1 \quad , \quad 4.1 \le x_2 \le 5.8$$

مثال ۱ : بهینه سازی



مثال ۱ : بهینه سازی

در این روش هر متغیر x_j که در فاصله $[a_j,b_j]$ تعریف می‌شود و بر اساس دقتی که برای محاسبات آن در نظر می‌گیریم (n) به صورت زیر کد می‌شود:

$$2^{mj-1} < (b_j - a_j) * 10^n \leq 2^{mj} - 1$$

برای تبدیل مقادیر باینری به فرم حقیقی و به منظور محاسبات نیز از تبدیل زیر استفاده می‌شود:

$$x_j = a_j + decimal(substring) * \frac{b_j - a_j}{2^{mj} - 1}$$

مثال ۱ : بهینه سازی

در این مثال دقت n را برابر با ۴ در نظر می‌گیریم:

$$(12.1-(-3.0))\times 10000 = 151000$$

$$2^{17} < 151000 \le 2^{18} \quad , \quad m_1 = 18$$

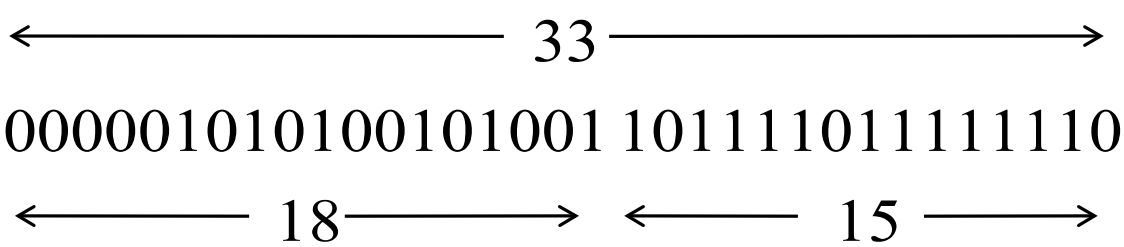
$$(5.8-4.1)\times 10000 = 17000$$

$$2^{14} < 17000 \le 2^{15} \quad , \quad m_2 = 15$$

$$m = m_1 + m_2 = 33$$

مثال ۱ : بهینه سازی

در این حالت طول هر کروموزوم برابر با ۳۳ بیت خواهد بود:



به طور مثال مقادیر x_1 و x_2 از مثال فوق به صورت زیر حساب می‌شوند:

Binary Number	Decimal Number
$x_1 = 0000010101001010 01$	5417
$x_2 = 101111011111110$	24318

مثال ۱ : بهینه سازی

$$x_1 = -3.0 + 5417 \times \frac{12.1 - (-3.0)}{2^{18} - 1} = -2.687969$$

$$x_2 = 4.1 + 24318 \times \frac{5.8 - 4.1}{2^{15} - 1} = 5.361653$$

در این حالت جمعیت اولیه به صورت زیر و تصادفی با مجموعه بیت‌های صفر و یک مقدار دهی می‌شود.

مثال ۱ : بهینه سازی

Initial Population, Initial population is randomly generated as follows:

$V_1 = [00000101010010100110111011111110]$

$V_2 = [00111010111001100000010101001000]$

$V_3 = [11100011100000100001010100100011 0]$

$V_4 = [10011011010010110100000010111001]$

$V_5 = [00001011110110001000111000110100 0]$

$V_6 = [11111010101101100000010110011001]$

$V_7 = [11010001001111100010011001110110 1]$

$V_8 = [00101101010000110001011001100 1100]$

$V_9 = [11111000101110110001110100011110 1]$

$V_{10} = [11110100111010101000010101101010]$

مثال ۱ : بهینه سازی

مقادیر معادل حقیقی برای کروموزوم‌های تولید شده به صورت زیر است:

$$V_1 = [x_1, x_2] = [-2.687969 , 5.361653]$$
$$V_2 = [x_1, x_2] = [0.474101 , 4.170144]$$
$$V_3 = [x_1, x_2] = [10.41945 , 4.661461]$$
$$V_4 = [x_1, x_2] = [6.159951 , 4.109598]$$
$$V_5 = [x_1, x_2] = [-2.301286 , 4.477282]$$
$$V_6 = [x_1, x_2] = [11.788084 , 4.174346]$$
$$V_7 = [x_1, x_2] = [9.342067 , 5.121702]$$
$$V_8 = [x_1, x_2] = [-0.330256 , 4.694977]$$
$$V_9 = [x_1, x_2] = [11.671267 , 4.873501]$$
$$V_{10} = [x_1, x_2] = [11.446273 , 4.171908]$$

مثال ۱ : بهینه سازی

رویه تخمین مقدار تطابق در سه گام زیر اشاره شده است:

Procedure : Evaluation

Step 1. Convert the chromosome's genotype to its phenotype. Here, this means converting binary string into relative real values

$$x^k = (x_1^k, x_2^k)\ ,\ k = 1, 2, \ldots, pop\_size.$$

Step 2. Evaluate the objective function $f(x^k)$

Step 3. Convert the value of objective function into fitness. For the maximization problem, the fitness is simply equal to the value of objective function $eval(v_k) = f(x^k)\ ,\ k = 1, 2, \ldots, pop\_size.$

مثال ۱ : بهینه سازی

بر اساس رویه اشاره شده مقادیر تطابق برای کروموزوم‌های تولید شده به صورت زیر خواهد بود:

$$eval(V_1) = f(-2.687969 , 5.361653) = 19.805119$$
$$eval(V_2) = f(10.41945 , 4.661461) = 17.370896$$
$$eval(V_3) = f(6.159951 , 4.109598) = 9.590546$$
$$eval(V_4) = f(-2.301286 , 4.477282) = 29.406122$$
$$eval(V_5) = f(11.788084 , 4.174346) = 15.686091$$
$$eval(V_6) = f(11.788084 , 4.174346) = 11.900541$$
$$eval(V_7) = f(9.342067 , 5.121702) = 17.958717$$
$$eval(V_8) = f([-0.330256 , 4.694977) = 19.763190$$
$$eval(V_9) = f(11.671267 , 4.873501) = 26.401669$$
$$eval(V_{10}) = f(11.446273 , 4.171908) = 10.252480$$

همان‌طور که مشاهده می‌شود کرومزوم V_4 بهترین در میان جمعیت است.

مثال ۱ : بهینه سازی

Selection

In most practices, a ***Roulette wheel*** approach is adopted as the selection procedure; it belongs to the fitness-proportional selection and can select a new population with respect to the probability distribution based on fitness values. The roulette wheel can be constructed as follows:

1. Calculate the fitness value $eval(v_k)$ for each chromosome v_k:

$$eval(v_k) = f(x) \qquad k = 1, 2, \ldots, pop\_size$$

2. Calculate the total fitness for the population:

$$F = \sum_{k=1}^{pop\_size} eval(v_k)$$

مثال ۱ : بهینه سازی

3. Calculate selection probability p_k for each chromosome v_k:

$$p_k = \frac{eval(v_k)}{F} \qquad k = 1, 2, \ldots, pop\_size$$

3. Calculate cumulative probability q_k for each chromosome v_k:

$$q_k = \sum_{j=1}^{k} p_j \qquad k = 1, 2, \ldots, pop\_size$$

The selection process begins by spinning the roulette wheel *pop_size* times; each time, a single chromosome is selected for a new population in the following way:

مثال ۱ : بهینه سازی

Procedure : Selection

Step 1. Generate a random number r from the range [0, 1].

Step 2. if $r \le q_1$, then select the first chromosome v_1 ; otherwise, select the kth chromosome v_k ($2 \le k \le$ pop_size) such that $q_{k-1} < r \le q_k$.

The total fitness F of the population is

$$F = \sum_{k=1}^{10} eval(v_k) = 178.135372$$

$p_1 = 0.111180$	$p_2 = 0.097515$	$p_3 = 0.053839$
$p_4 = 0.165077$	$p_5 = 0.088057$	$p_6 = 0.066806$
$p_7 = 0.100815$	$p_8 = 0.110945$	$p_9 = 0.148211$
$p_{10} = 0.057554$		

مثال ۱ : بهینه سازی

The cumulative probabilities q_k for each chromosome v_k ($k=1, \ldots, 10$) is :

$q_1 = 0.111180$ $q_2 = 0.208695$ $q_3 = 0.262534$

$q_4 = 0.427611$ $q_5 = 0.515668$ $q_6 = 0.582465$

$q_7 = 0.683290$ $q_8 = 0.794234$ $q_9 = 0.942446$

$q_{10} = 1.000000$

Now we are ready to spin the roulette wheel 10 times, and each time we select a single chromosome for a new population. Let us assume that a random sequence of 10 numbers from the range [0, 1] is as follows:

0.301431	**0.322062**	**0.766503**	**0.881893**
0.350871	**0.583392**	**0.177618**	**0.343242**
0.032685	**0.197577**		

مثال ۱ : بهینه سازی

The first number $r_1 = 0.301431$ is greater than q_3 and smaller than q_4, meaning that the chromosome v4 is selected for the new population.

The second number $r_2 = 0.322062$ is greater than q_3 and smaller than q_4, meaning that the chromosome v4 is again selected for the new population and so on.

Finally the new population consists of the following chromosomes:

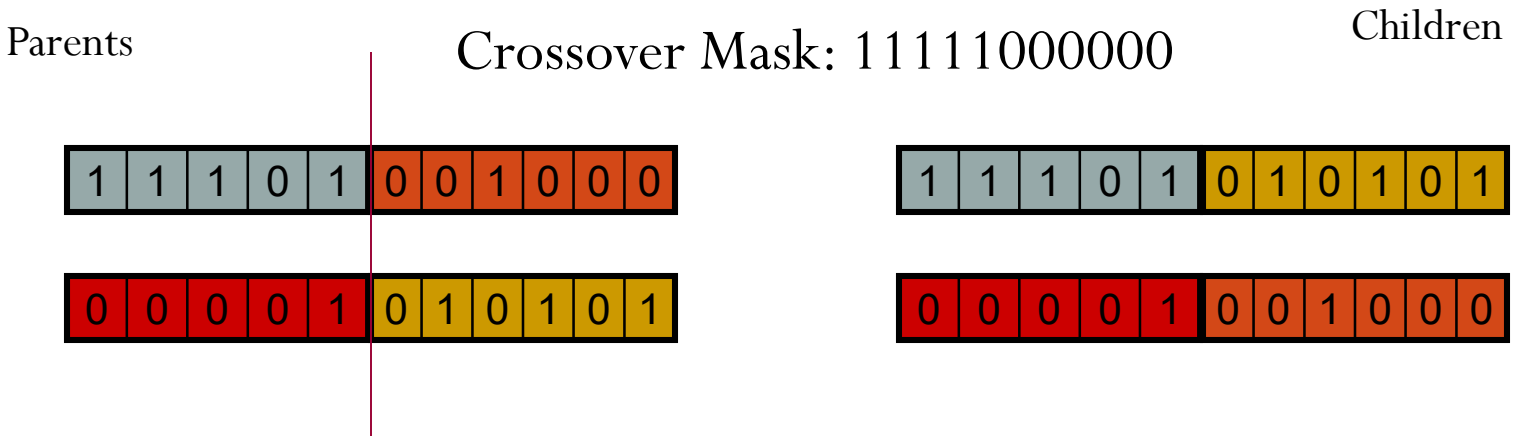
$$\mathbf{V}'_1 = V_4 \qquad \mathbf{V}'_2 = V_4 \qquad \mathbf{V}'_3 = V_8 \qquad \mathbf{V}'_4 = V_9$$
$$\mathbf{V}'_5 = V_4 \qquad \mathbf{V}'_6 = V_7 \qquad \mathbf{V}'_7 = V_2 \qquad \mathbf{V}'_8 = V_4$$
$$\mathbf{V}'_9 = V_1 \qquad \mathbf{V}'_{10} = V_2$$

عملگر آمیزش برای کدینگ باینری

- عملگر آمیزش با استفاده از دو رشته والد، دو رشته فرزند به وجود می‌آورد .
- برای این کار قسمتی از بیت‌های والدین در بیت‌های فرزندان کپی می‌شود .
- انتخاب بیت‌هایی که باید از هر یک از والدین کپی شوند به روش‌های مختلف انجام می‌شود:
  - Single-point crossover
  - Two-point crossover
  - Uniform crossover
- برای تعیین محل بیت‌های کپی شونده از یک رشته به نام Crossover Mask استفاده می‌شود.

Single-point crossover

- یک نقطه تصادفی در طول رشته انتخاب می‌شود .
- والدین در این نقطه به دو قسمت می‌شوند.
- هر فرزند با انتخاب تکه اول از یکی از والدین و تکه دوم از والد دیگر به وجود می‌آید.

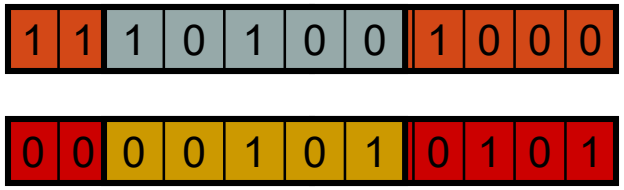


روش‌های دیگر Crossover

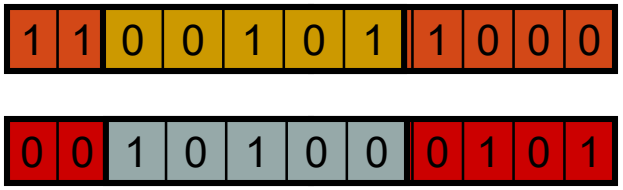
- Two-point crossover

Crossover Mask: 00111110000

Parents

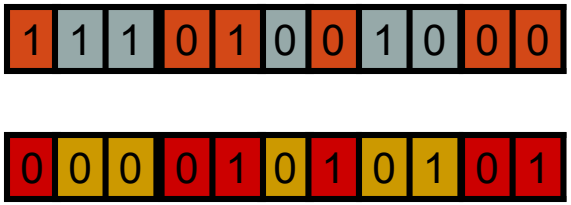


Children



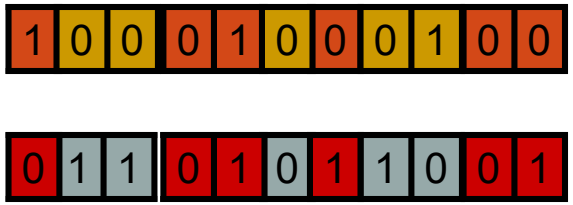
- Uniform crossover

Parents



Crossover Mask: 10011010011

Children



عملگر جهش برای کدینگ باینری

- عملگرجهش برای به وجود آوردن فرزند فقط از یک والد استفاده می‌کند.
- این‌کار با انجام تغییرات کوچکی در رشته اولیه به وقوع می‌پیوندد.
- با استفاده از یک توزیع یکنواخت یک بیت به صورت تصادفی انتخاب و مقدار آن تغییر پیدا می‌کند.
- معمولا جهش بعد از انجام آمیزش اعمال می‌شود.

Parent

1	1	1	0	1	0	0	1	0	0	0

Child

1	1	1	0	1	1	0	1	0	0	0

مثال ۱ : بهینه سازی ادامه ...

$$V_1 = [1001101101001011010000000101110 01]$$
$$V_2 = [0010110101000011000101100 11001100]$$

The resulting offspring by exchanging the right parts of their parents would be as follows:

$$V'_1 = [1001101101001011000101100 11001100]$$
$$V'_2 = [0010110101000011010000000101110 01]$$

The probability of crossover is set as pc = 0.25 , so we expect that, on average, 25% of chromosomes undergo crossover. Crossover is performed in the following way:

مثال ۱ : بهینه سازی ادامه ...

Procedure Crossover

begin
  k = 1
  while (k ≤ 10) **do**
  r_k = random number from [0,1]
  if (r_k <0.25) **then**
    select v_k as one parents for crossover
  end
  k = k+1
  end
end

مثال ۱ : بهینه سازی ادامه ...

Assume that the sequence of random number is

0.625721	**0.266823**	**0.288644**	**0.295114**
0.163274	**0.567461**	**0.085940**	**0.392865**
0.770714	**0.548656**		

This means that the chromosomes v'_5 and v'_7 were selected for crossover. We generate a random integer number **Position** from the range [1, 32] (because 33 is the total length of a chromosome) as cutting point or in other word, the position of the crossover point.

مثال ۱ : بهینه سازی ادامه ...

Assume that the generated number **Position** equal 1, the two chromosomes are cut after the first bit, offspring are generated by exchanging the right parts of them as follows:

$$V'_5 = [1001101101001011010000001011 1001]$$
$$V'_7 = [0011101011100110000000101010 01000]$$

$$\downarrow$$

$$V'_5 = [1001101101001011010000001011 1001]$$
$$V'_7 = [0011101011100110000000101010 01000]$$

مثال ۱ : بهینه سازی ادامه ...

Mutation

Mutation alters one or more genes with a probability equal to the mutation rate.

Assume that the 18^{th} gene of the following chromosome v_1 is selected for a mutation.

Since the gene is **1**, it would be flipped into **0**. thus the chromosome after mutation would be

$$v_1 = [1001101101001011010000000101110 01]$$

$$v'_1 = [0011101011100110000001010100 1000]$$

مثال ۱ : بهینه سازی ادامه...

در این مثال نرخ جهش برابر با ۰.۰۱ در نظر گرفته شده است، لذا انتظار داریم به صورت متوسط ۱٪ از بیت‌های جمعیت در عمل جهش شرکت کنند، با توجه به وجود ۳۳۰ ژن در کل جمعیت، انتظار داریم حدود ۳.۳ جهش در هر تولید داشته باشیم.

Random Number	Bit position	Chromosome	Gene
0.009857	105	4	6
0.003113	164	5	32
0.000946	199	7	1
0.001282	329	10	32

مثال ۱ : بهینه سازی ادامه ...

The corresponding decimal values of variables $[x_1, x_2]$ and fitness are as follows:

$$f(6.159951 , 4.109598) = 29.406122$$
$$f(6.159951 , 4.109598) = 29.406122$$
$$f(-0.330256 , 4.694977) = 19.763190$$
$$f(11.907206 , 4.873501) = 5.702781$$
$$f(8.024130 , 4.170248) = 19.91025$$
$$f(9.342067 , 5.121702) = 17.958717$$
$$f(6.159951 , 4.109598) = 29.406122$$
$$f(6.159951 , 4.109598) = 29.406122$$
$$f(-2.687969 , 5.361653) = 19.805119$$
$$f(0.474101 , 4.170248) = 17.370896$$

مثال ۱ : بهینه سازی ادامه ...

Now we just complete one iteration (generation) of genetic algorithm. The test run is terminated after 1000 generations.

We have obtained the best chromosome in the 419^{th} generation as follows:

$$v^* = [11111000000011100011110100101011 0]$$

$$eval(v^*) = f(11.631407, 5.724824) = 38.818208$$

$$x_1^* = 11.631407$$
$$x_2^* = 5.724824$$
$$f(x_1^*, x_2^*) = 38.818208$$

مثال ۲ : Word Matching

در این مثال به بررسی قابلیت الگوریتم ژنتیک در حل مسائل Word Matching می‌پردازیم.

در این مساله هدف یافتن عبارت "to be or not to be" با استفاده از الگوریتم ژنتیک می‌باشد.

با توجه به وجود ۲۶ کاراکتر مختلف در زبان انگلیسی و تعداد ۱۳ حرف در عبارت فوق، احتمال رسیدن به عبارت مورد نظر به صورت کاملا تصادفی برابر با $(1/26)^{13}=4.03038*10^{-19}$ می‌باشد!

مثال ۲ : Word Matching

برای حل این مساله از روش کدینگ اسکی استفاده می‌کنیم تا دنباله کاراکترها را نمایش دهیم.

دنباله کاراکترها به صورت Lower Case با مجموعه اعداد ۹۷ الی ۱۲۲ نمایش داده می‌شوند. (از نمایش فاصله صرف نظر می‌شود)

عبارت مورد نظر به صورت زیر در کد اسکی نمایش داده می‌شود:

[116, 111, 98, 101, 111, 114, 110, 111, 116, 116, 111, 98, 101]

Word Matching : مثال ٢

Generate an initial population of 10 random phrase as follows:

[114, 122, 102, 113, 100, 104, 117, 106, 97, 114, 100, 98, 101]

[110, 105, 101, 100, 119, 118, 121, 118, 106, 97, 104, 102, 106]

[115, 99, 121, 117, 101, 105, 115, 111, 115, 113, 118, 99, 98]

[102, 98 , 102, 118, 114, 97, 109, 116, 101, 107, 117, 118, 115]

[107, 98, 117, 113, 114, 116, 106, 116, 106, 101, 110, 115, 98]

[102, 119, 121, 113, 121, 107, 107, 116, 122, 121, 111, 106, 104]

[116, 98, 120, 98, 108, 115, 111, 105, 122, 103, 103, 119, 109]

[101, 111, 111, 117, 114, 104, 100, 120, 98, 118, 116, 120, 97]

[100, 116, 114, 105, 117, 111, 115, 114, 103, 107, 109, 98, 103]

[106, 118, 112, 98, 103, 101, 109, 116, 112, 106, 97, 108, 113]

Word Matching : مثال ٢

Now, we convert this population to string to see what they look like:

rzfqdhujardbe

niedwvyvjahfj

scyueisosqvcb

fbfvramtekuvs

……

……

jvpbgemtpjalq

مثال ۲ : Word Matcing

به منظور حل مساله، تابع تطابق برابر با تعداد صحیح حروفی در نظر گرفته می‌شود که در مکان صحیح خود قرار گرفته باشند. به طور مثال برای عبارت rzfqdhujardbe مقدار این تطابق برابر با ۲ خواهد بود.

در این مساله فقط از عملگر جهش برای این منظور استفاده شده است. در جدول زیر تعداد ۳۰ تکرار از الگوریتم برای بهترین پاسخ را خواهید دید.

Word Matching : مثال ٢

Table 1.2 The Best String for Each Generation

Generation	String	Fitness	Generation	String	Fitness
1	rzfqdhujardbe	2	16	rzbwornottobe	10
2	rzfqdhuoardbe	3	17	rzbwornottobe	10
3	rzfqghuoatdbe	4	18	rzbwornottobe	10
4	rzfqghuoztobe	5	19	rzbwornottobe	10
5	rzfqghhottobe	6	20	robwornottobe	11
6	rzfqohhottobe	7	21	tobwornottobe	12
7	rzfqohnottobe	8	22	tobwornottobe	12
8	rzfqohnottobe	8	23	tobeornottobe	13
9	rzfqohnottobe	8	24	tobeornottobe	13
10	rzfqohnottobe	8	25	tobeornottobe	13
11	rzfqornottobe	9	26	tobeornottobe	13
12	rzfqornottobe	9	27	tobeornottobe	13
13	rwfwornottobe	9	28	tobeornottobe	13
14	rwcwornottobe	9	29	tobeornottobe	13
15	rzcwornottobe	9	30	tobeornottobe	13