Soft Computing

H. Nezamabadi-pour
Electrical Eng. Dept.,
University of Kerman, Kerman, Iran.
nezam@mail.uk.ac.ir

LECTURE2: Binary GA

- فرض کنید که هدف از بهینه‌سازي، پیدا کردن بیشینه تابع f در یک دامنه مشخص باشد:

$$f\left(x_1, x_2, \ldots, x_m\right)$$

$$x_i^{lo} \le x_i \le x_i^{hi} \quad \text{for i} = 1,2,\ldots,\text{m}$$

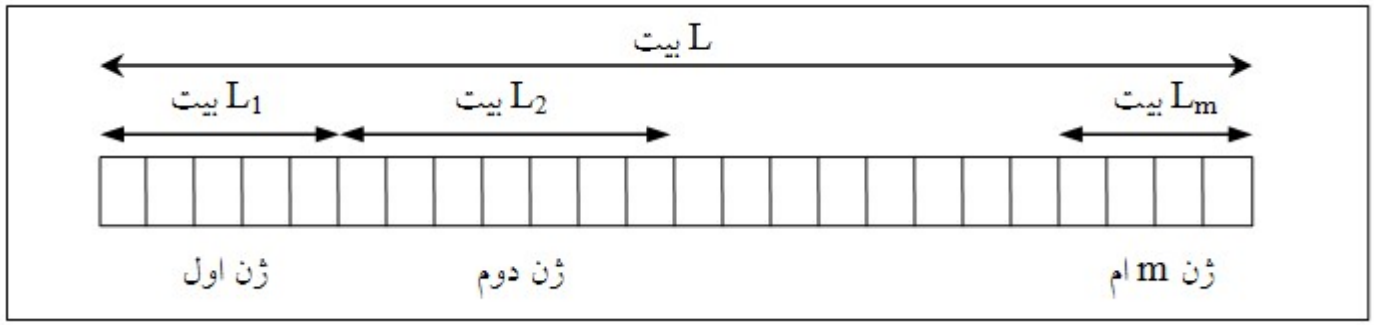
- در این وضعیت، پیدا کردن مقادیري براي متغیرهاي حقیقی x_1 تا x_m مد نظر است که تابع f، به ازاي آنها بیشترین مقدار را به خود بگیرد.

- به عبارتي هدف از بیشنه‌سازي، یافتن X^* است به گونه‌اي که

$$f(X^*) \ge f(X) \quad , \quad \forall X \in D_f$$

LECTURE2: Basic Concepts

- در حل مساله با الگوريتم وراثتي، هر يک از متغيرهاي اين مساله بصورت يک ژن در وراثت طبيعي در نظر گرفته مي‌شوند.
- از کنار هم قرار گرفتن تمام متغيرهاي يک مساله (ژنها)، يک کروموزوم ساخته مي‌شود.
- *هلند* براي اولين بار از رشته‌هاي بيتي براي بيان اطلاعات کروموزومها استفاده کرد
- مثال: 3 متغير و هر متغير 10 بيت



تعاريف

- **كروموزوم**
  - در الگوريتم وراثتي هر كروموزوم بيانگر يک جواب مساله بصورت رمز شده است و يک نقطه در فضاي جستجو را نشان مي‌دهد. هر كروموزوم از تعداد مشخصي ژن تشكيل شده است كه همان متغيرهاي مساله هستند. در الگوريتم وراثتي بايناري، ژنها از تعداد معيني بيت تشكيل مي‌شوند. هر ژن مي‌تواند مقادير متفاوتي به خود بگيرد كه اين مقادير آللهاي يكديگر براي يک متغير در يک مساله هستند.
- **جمعيت**
  - مجموعه‌اي از كروموزومها، يک جمعيت را مي‌سازند. از هر جمعيت با استفاده از عملگرهاي وراثتي، يک جمعيت جديد ساخته مي‌شود.

تعاريف

تابع شايستگي (برازندگي)

- براي حل هر مساله بهينه‌سازي، بايد يک تابع شايستگي طراحي شود. تابع شايستگي براي هر کروموزوم(يک مجموعه از متغيرهاي ورودي)، يک عدد نامنفي برمي-گرداند که نشاندهنده کارآيي يا توانايي آن کروموزوم در حل مساله است. درحل بسياري از مسائل، به خصوص بهينه‌سازي توابع، تابع شايستگي مقدار آن تابع را به ازاي متغيرهاي ورودي برمي‌گرداند. به عبارت ديگر، تابع شايستگي همان تابع هدف است. اما در حل مسائل مهندسي در دنياي واقعي، اغلب بايد يک تابع شايستگي مناسب ابداع و طراحي شود.

توليد مثل

- در مرحله توليد مثل، تعدادي از افراد جمعيت نسل قبل انتخاب مي‌شوند. اين اعضا ضمن ترکيب با يکديگر نسل جديد را مي‌سازند. هر يک از اعضاي انتخاب شده براي توليد مثل، والد و هر يک از اعضاي توليد شده، فرزند ناميده مي‌شوند. براي توليد مثل و ساختن نسل بعد از عملگرهاي وراثتي استفاده می‌شود که مهمترين آنها، عملگرهاي انتخاب، همبري و جهش هستند.-

تعاريف

عملگر انتخاب

- با استفاده از اين عملگر از بين كروموزومهاي موجود در يک جمعيت تعدادي براي توليد مثل انتخاب شده و به حوضچه ازدواج منتقل مي‌شوند. انتخاب كروموزومها بصورت اتفاقي است اما فرايند انتخاب به گونه‌اي است كه كروموزومهاي با شايستگي بيشتر از شانس بيشتري براي انتخاب و توليد مثل برخوردار مي‌شوند.

عملگر همبري

- عملگر همبري مهمترين عملگر در الگوريتم وراثتي است. اين عملگر بر روي دو (چند) كروموزوم والد عمل همبري را اجرا كرده و دو فرزند براي نسل جديد توليد مي‌كند.

تعاريف

عملگر جهش

- اين عملگر، يک ژن از يک کروموزوم را بصورت تصادفي انتخاب کرده و محتويات آن را اندکي تغيير مي‌دهد.

همگرايي

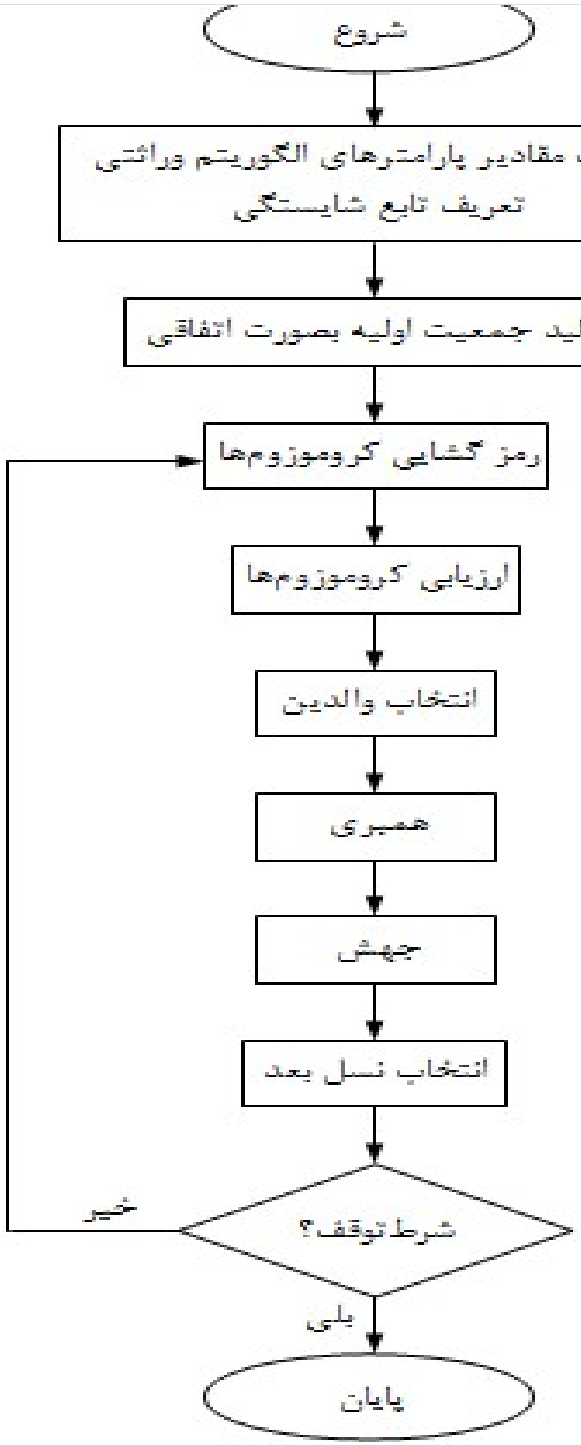
- همگرايي، پيشرفت به سوي يکنواختي است. چنانچه الگوريتم وراثتي به طرز صحيحي پياده‌سازي شود، جمعيت در نسلهاي متمادي تکامل پيدا مي‌کند و متوسط برازندگي جمعيت نسل به نسل به سمت برازندگي بهترين عضو آن جمعيت نزديکتر شده و به سمت بهينه کلي سوق پيدا مي‌کند. يک جمعيت زماني همگرا ناميده مي‌شود که همه ژنهاي آن همگرا شده باشند و ژني همگرا گفته مي‌شود که 95 درصد افراد جمعيت مقدار يکساني در آن ژن داشته باشند.-

تعاریف

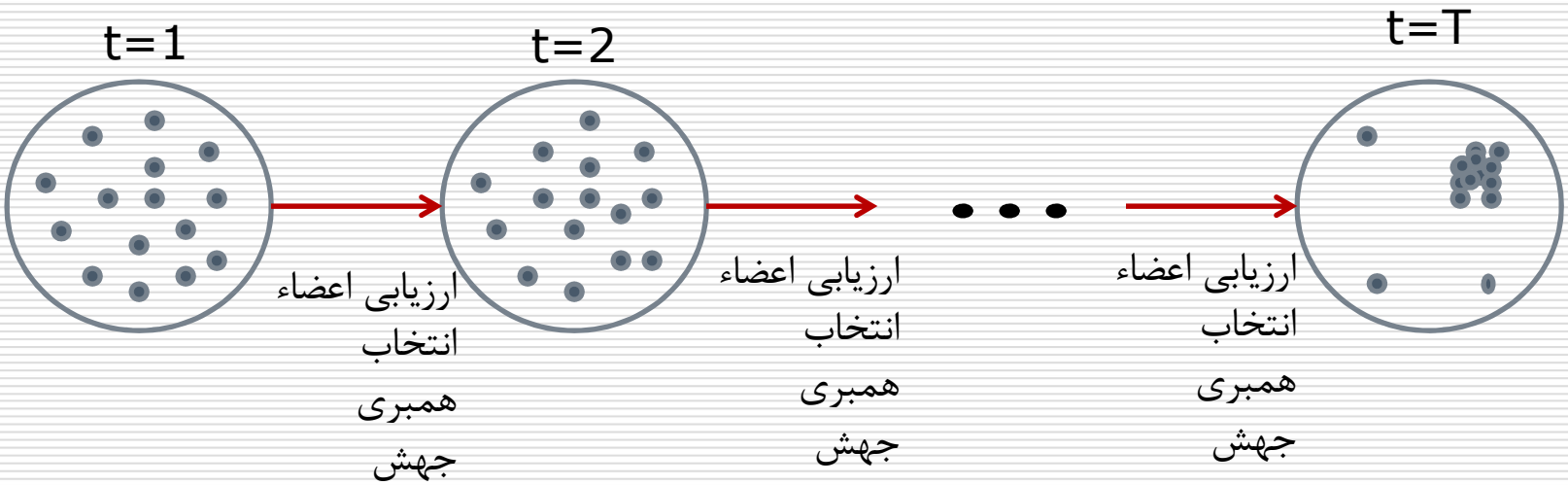
جدول۲-۱ انطباق الگوریتم وراثتی و وراثت طبیعی

وراثت طبیعی	الگوریتم وراثتی
فرد یا عضو	یک جواب برای مساله
جمعیت	مجموعه‌ای از جوابها
شایستگی	کیفیت یک جواب
کروموزوم	یک جواب رمز شده به شکل رشته بیتی
ژن	بخشی از یک جواب بصورت رمز
همبری و جهش	عملگرهای جستجو
انتخاب طبیعی	عملگر انتخاب(استفاده از جوابهای خوب در ادامه کار الگوریتم)
آلل	مقادیر مختلف برای یک متغیر

ساختار كلي الگوريتم وراثتي



قالب کلی



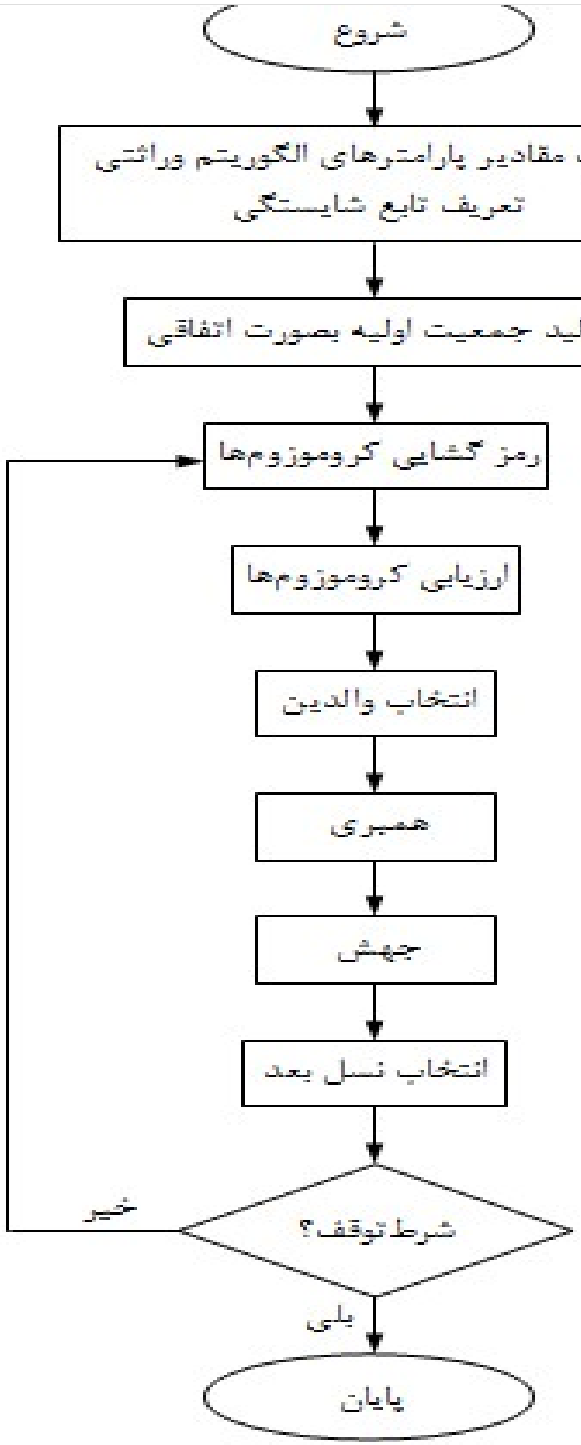
عملگرهای الگوریتم شامل انتخاب- همبری – جهش هستند.

مقدار دهي پارامترها

- تعيين تعداد اعضاي جمعيت
- نرخ همبري
- نرخ جهش
- تعداد متغيرها
- طول هر متغير
- طول كروموزوم
- تعيين محدوده هر متغير و دامنه جستجو
- نحوه خاتمه الگوريتم

$$L = \sum_{i=1}^{m} L_i$$

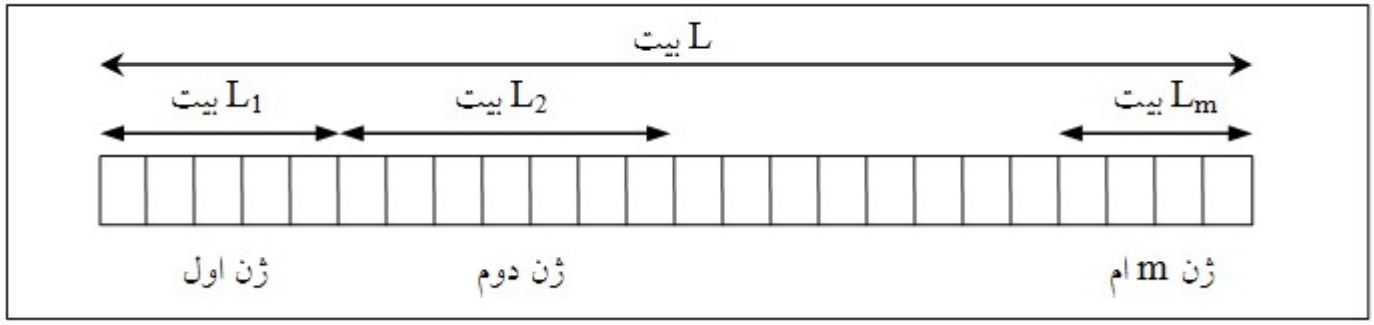
ساختار كلي الگوريتم وراثتي



تولید جمعیت اولیه

- براي ساختن جمعيت اوليه كافي است كه يک ماتريس N*L (شامل Nسطر و L ستون) از صفر و يک بصورت اتفاقي با تابع توزيع يكنواخت ساخته شود. در اين شرايط هر سطر اين ماتريس اطلاعات مربوط به يک كروموزوم را با خود دارد.

يک كروموزوم

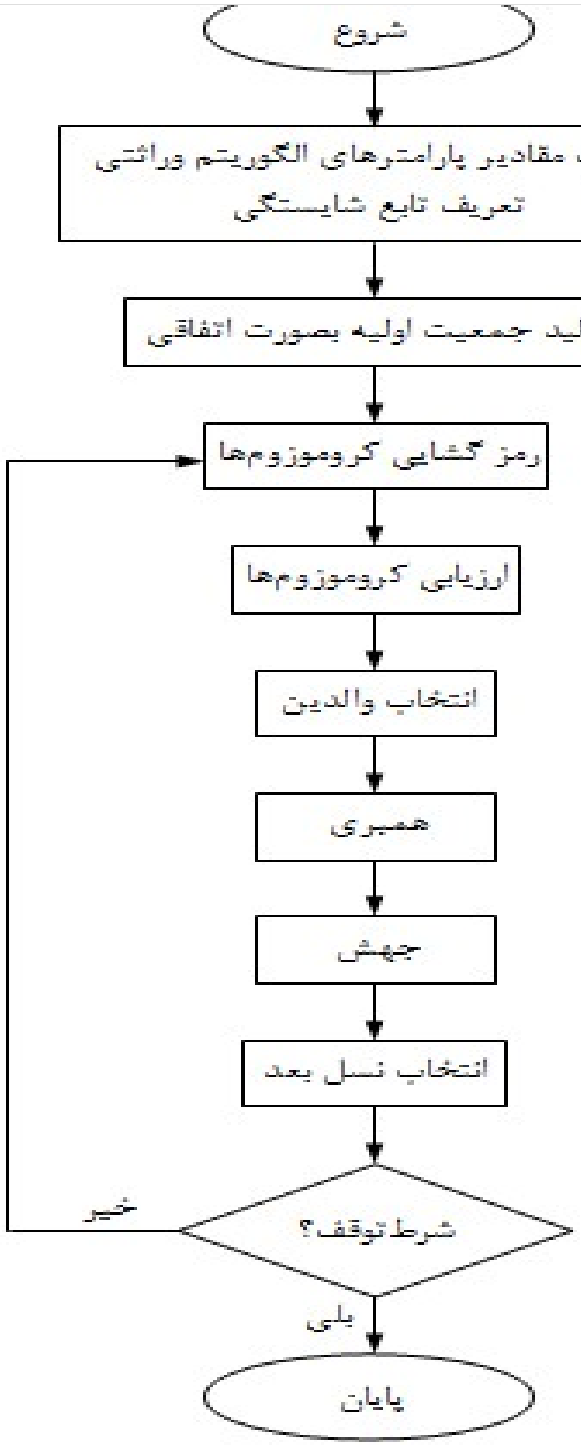


Population

- هر کروموزوم یک بردار L بیتی است. با مقادیر 0 و 1

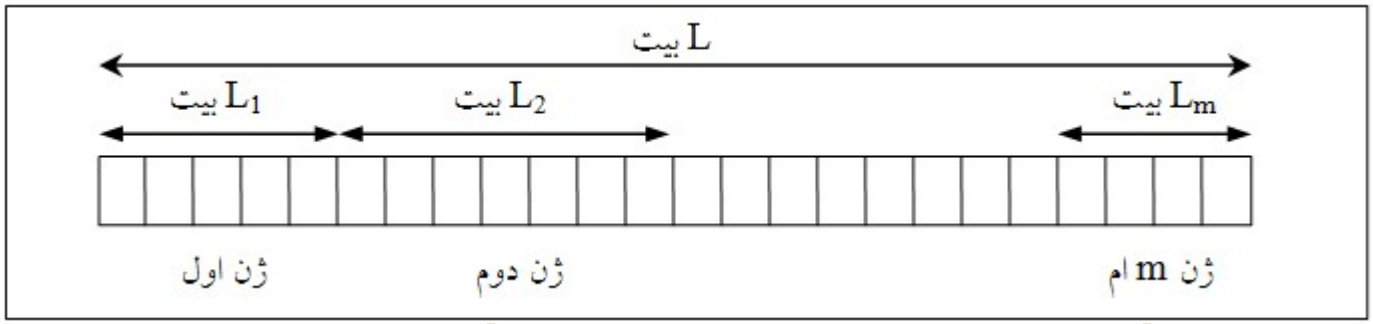
$$X = \begin{bmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ \cdot \\ X_N \end{bmatrix} = \begin{bmatrix} x_1^1 & x_1^2 \dots\dots\dots x_1^L \\ x_2^1 & x_2^2 \dots\dots\dots x_2^L \\ & \cdot \\ & \cdot \\ & \cdot \\ x_N^1 & x_N^2 \dots\dots\dots x_N^L \end{bmatrix} \qquad X = \begin{bmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ \cdot \\ X_N \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \dots\dots 1 & 0 \\ 1 & 1 & 0 \dots\dots 0 & 0 \\ & & \cdot & \\ & & \cdot & \\ & & \cdot & \\ 0 & 0 & 1 \dots\dots 1 & 1 \end{bmatrix}$$

ساختار كلي الگوريتم وراثتي



رمز گشايي كروموزومها

ابتدا هر كروموزوم بايد به ژنها شكسته شود و هر ژن جداگانه رمزگشايي می شود.



Decoding

$[x_1$ x_2 $x_m]$

رمز گشايي كروموزومها

- رمزگشايي ژنها (متغيرها)

L_{i-1}	L_{i-2}	...	۲	۱	۰

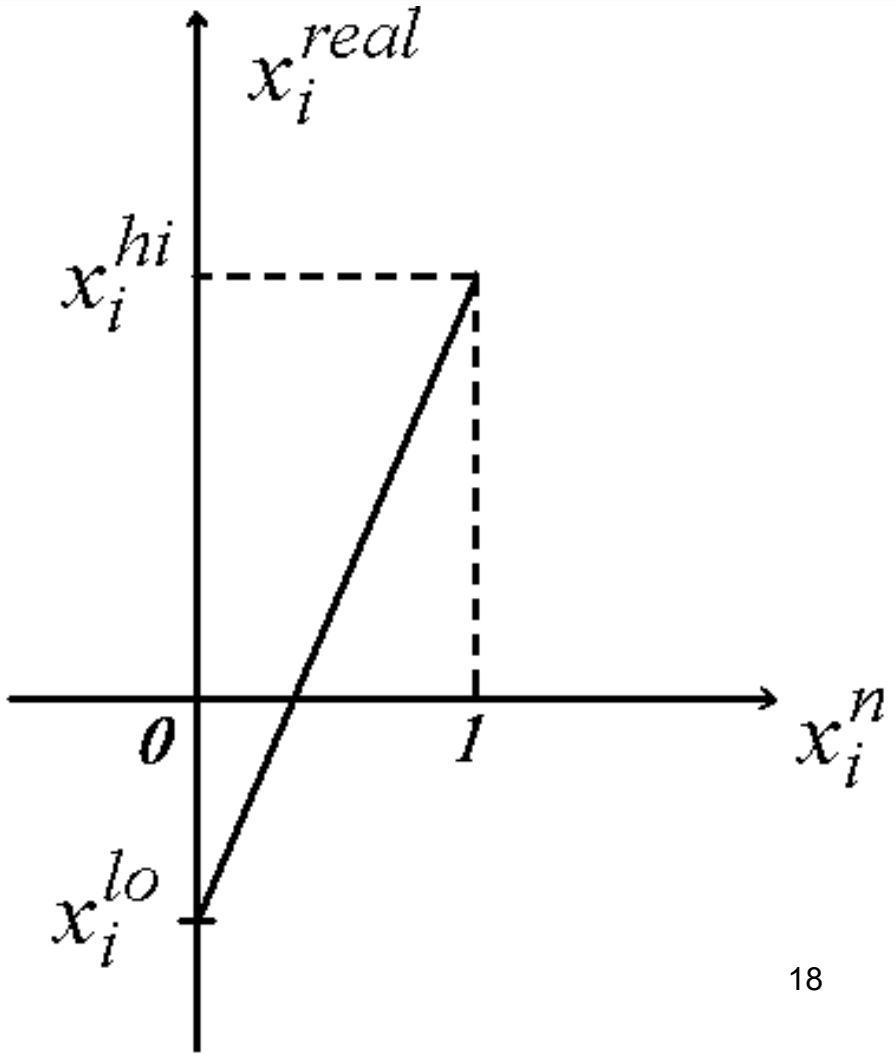
نحوه شماره گذاری بیتهای متغیرi ام (x_i)، کم ارزشترین بیت با شماره صفر و با ارزشترین بیت با شماره L_{i-1} نامگذاری می‌شود

$$(x_i)_{10} = bit(0)\times 2^0 + bit(1)\times 2^1 + bit(2)\times 2^2 + ... + bit(L_i - 1)\times 2^{(Li-1)} = \sum_{k=0}^{Li-1} bit(k)\times 2^k$$

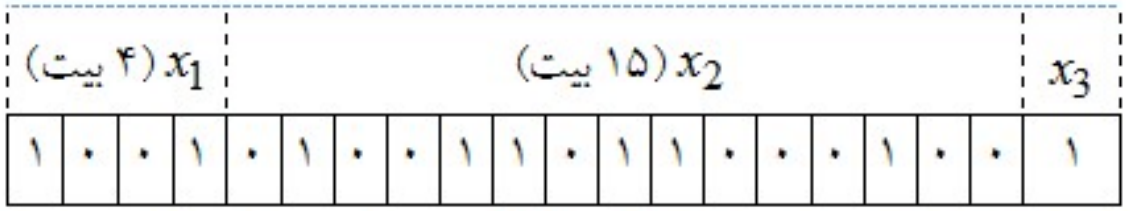
$$x_i^n = \frac{(x_i)_{10}}{2^{Li} - 1}$$

رمز گشايي كروموزومها

$$x_i^{real} = x_i^{lo} + x_i^n \times \left(x_i^{hi} - x_i^{lo} \right)$$



رمز گشايي كروموزومها: يک مثال



شکستن یک کروموزوم به متغیرهایش، در این مثال متغیرهای x_1، x_2 و x_3 به ترتیب ۴، ۱۵ و ۱ بیت طول دارند.

$$x_2 = (0100110110000100)_2 = (9924)_{10}$$

$$x_2^n = \frac{9924}{2^{15}-1} = \frac{9924}{32767} = 0.3029$$

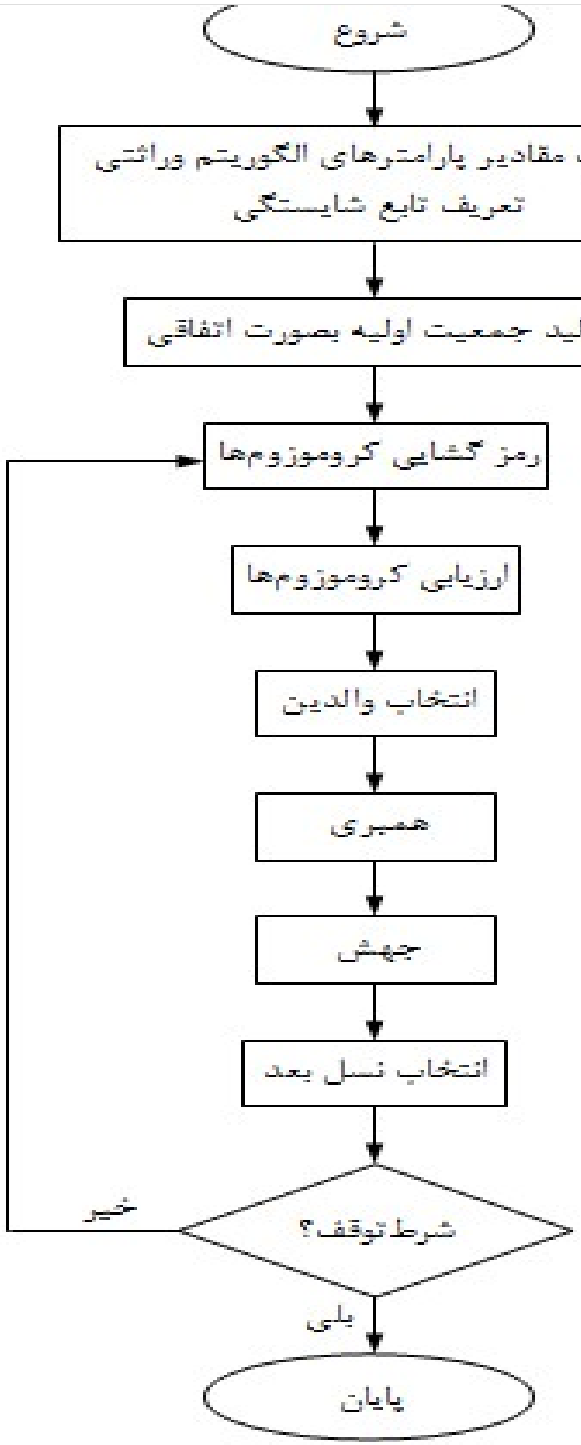
$$x_2^{real} = x_2^{lo} + \left(x_2^{hi} - x_2^{lo}\right) \times x_2^n = -2.1 + (4 + 2.1) \times 0.3029 = -0.2523$$

فرض بر این است که این متغیر در بازه [4، 2/1-] تعریف شده است.

رمز گشايي كروموزومها: خطاي چنديسازي

$$\Delta_j = \frac{x_j^{hi} - x_j^{lo}}{2^{Lj+1}}$$

ساختار كلي الگوريتم وراثتي



ارزيابي كروموزومها

- در اين مرحله به ازاي هر يک از بردارهاي ورودي، مقدار تابع شايستگي محاسبه مي شود. براي اين كار، مقادير بدست آمده براي متغيرها در تابع شايستگي قرار مي گيرد. خروجي تابع شايستگي به ازاي هر دسته از متغيرهاي ورودي به عنوان شايستگي كروموزوم مربوط به آن در نظر گرفته مي شود.

تابع شايستگي Fitness function

- اين تابع بايد به ازاي هر جواب كه توسط كروموزومها ارائه مي شود، عددي نامنفي و متناسب با كارآيي و شايستگي آن جواب برگرداند.
- به عبارت ديگر، اين تابع بايد به گونه‌اي طراحي شود كه هر چه جواب ارائه شده به جواب بهينه نزديكتر باشد، عدد شايستگي بزرگتري برگرداند كه نشان دهنده شايستگي بيشتر آن كروموزوم(جواب) است.
- فراموش نشود كه تابع شايستگي با توجه به تابع هدف طراحي مي‌شود.

نگاشت تابع هدف به تابع شايستگي

- توابع هدف مثبت حداكثر شونده

$Fit(X)=f(X)$

- توابع هدف با مقادير منفي

$$fit(X)=f(X)+\left|f^{\min}\right| \quad , \quad \text{for} \quad f^{\min}<0$$

- كمينه كردن توابع هدف

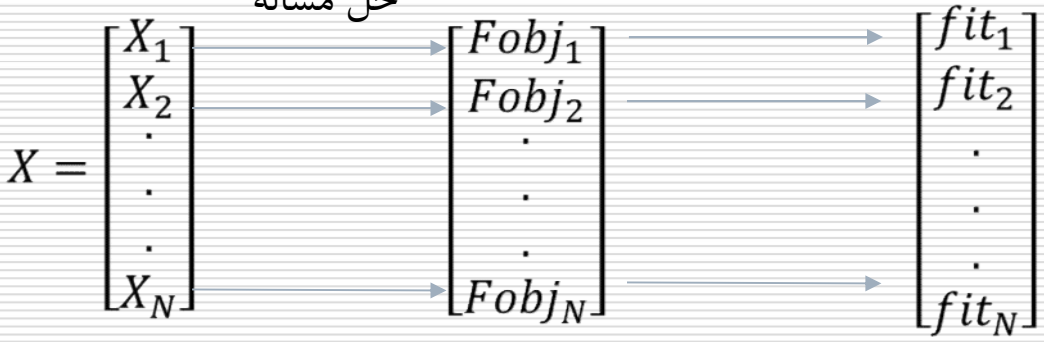
$$fit(X)=f^{\max}-f(X)$$

$$fit(X)=\frac{1}{f(X)}$$

ارزیابی Evaluation

- در هر تکرار، به ازای هر عضو یکبار مساله حل می شود.

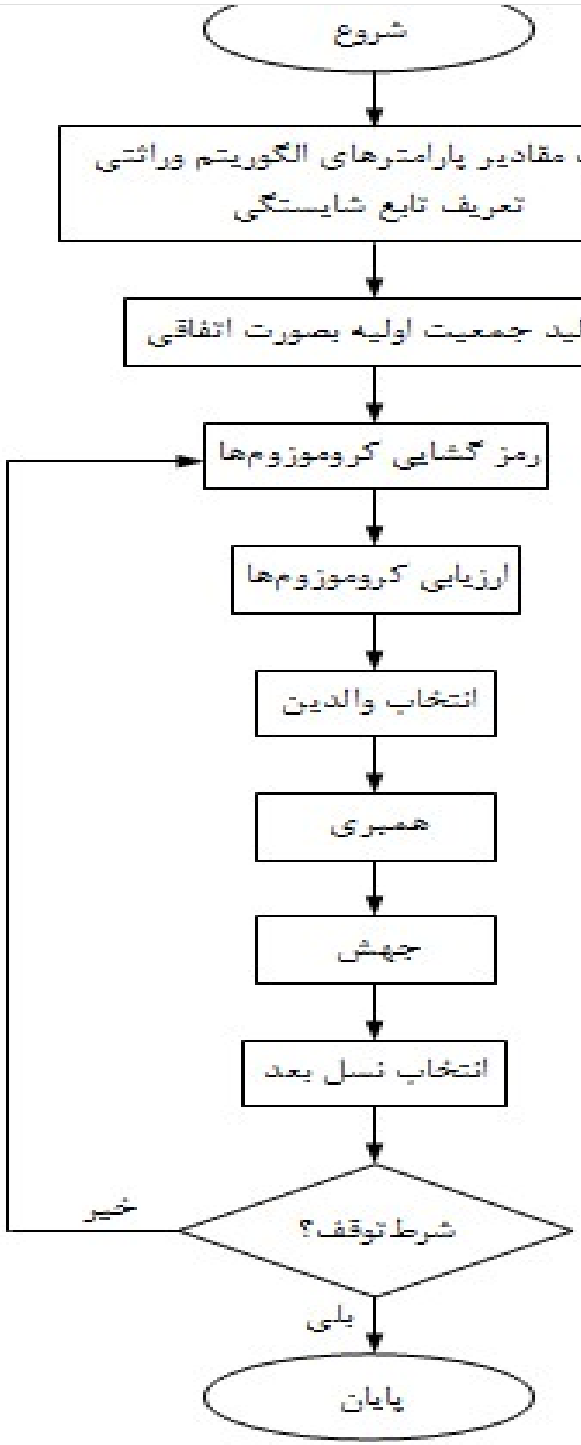
حل مساله



Objective function
تابع هدف

Fitness function
تابع برازندگی (شایستگی)

ساختار كلي الگوريتم وراثتي



عملگر انتخاب Selection

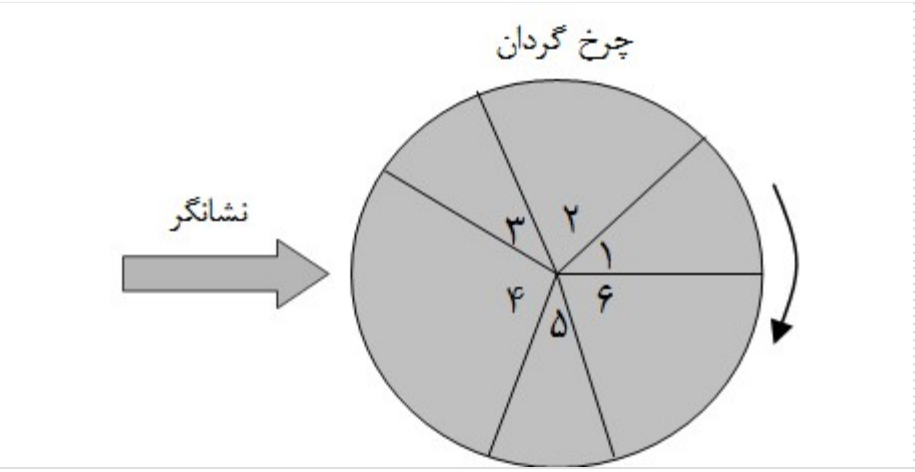
- عملگر انتخاب در الگوريتم وراثتي، برداشتي از انتخاب طبيعي در وراثت طبيعي است.
- هدف از اعمال اين عملگر، انتخاب بعضي از افراد جمعيت براي زاد و ولد و ايجاد نسل بعد است
- مشهورترين عملگر انتخاب، انتخاب متناسب با شايستگي است كه براي پياده سازي آن از چرخ گردان استفاده مي شود
- اين روش به انتخاب چرخ گردان شهرت پيدا كرده است.

عملگر انتخاب

- در انتخاب متناسب با برازندگي به كروموزومهاي شايسته‌تر(جوابهاي بهتر)، شانس بيشتر و به كروموزومهاي ضعيفتر(جوابهاي بدتر) شانس كمتري براي بقا و توليد مثل داده مي‌شود.
- در الگوريتم وراثتي عملگرها بر پايه احتمالات عمل مي‌كنند و قطعيتي در كار نيست. اين موضوع به اين معني است كه به هر يک از كروموزومها متناسب با شايستگي‌شان شانسي براي انتخاب تعلق مي‌گيرد، ولي انتخاب بر مبناي احتمال صورت مي‌پذيرد.

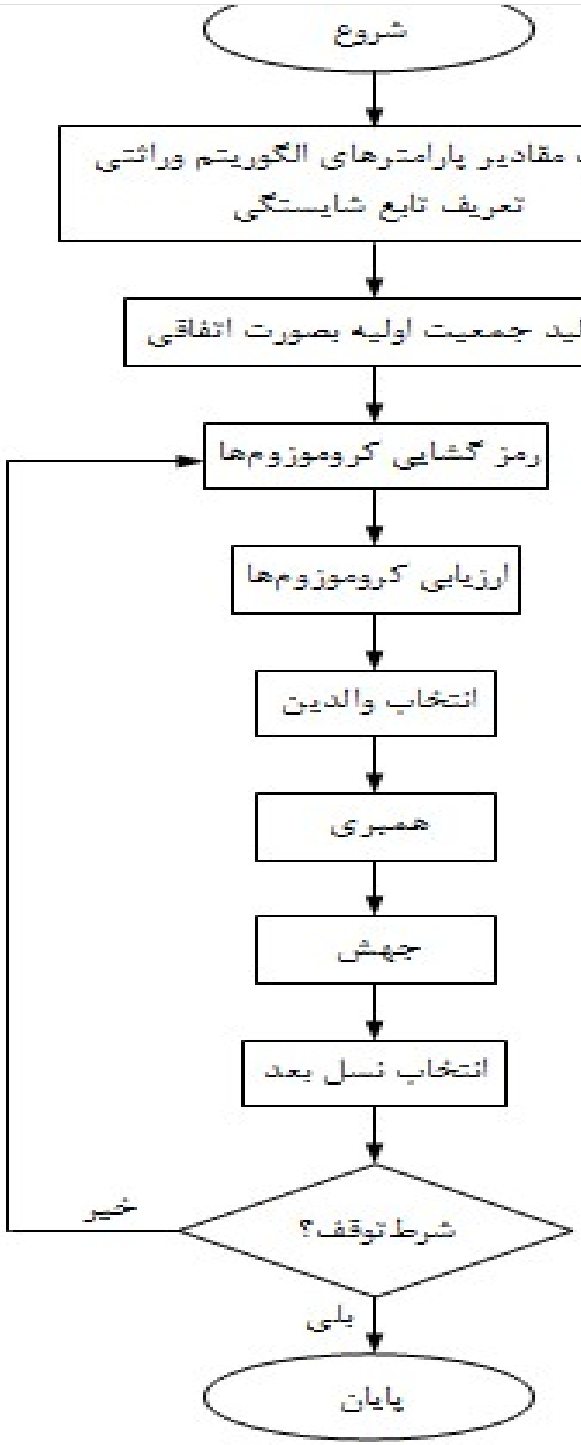
عملگر انتخاب: چرخ گردان

$$P_i = \frac{fit_i}{\sum_{j=1}^{N} fit_j}$$



$$EV_i = P_i \times N$$

ساختار كلي الگوريتم وراثتي

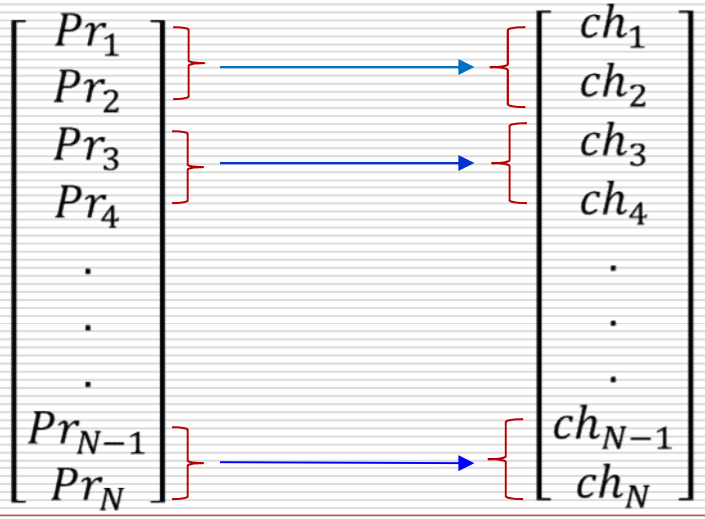


عملگر همبري Cross over

- اين عملگر مهمترين عملگر در الگوريتم وراثتي است
- براي ساختن نسل بعد، دو كروموزوم از حوضچه ازدواج به عنوان والد انتخاب شده و با عمل همبري دو فرزند بوجود ميآيد.
- كه عملگر همبري بطور قطعي روي تمام والدين اجرا نمي‌شود (احتمال همبري P_C معمولاً عددي بين 0.6 و 0.9)

عملگر همبري

- عملگر همبري در دو گام انجام مي شود:
- **گام اول:** از بين افراد موجود در حوضچه تزويج، دو نفر بصورت اتفاقي براي همبري انتخاب مي شوند.
- **گام دوم:** دو كروموزوم منتخب به عنوان والدين، طي يكي از روشهاي همبري فرزند يا انتقال زوج فرزندان توليد مي كنند.

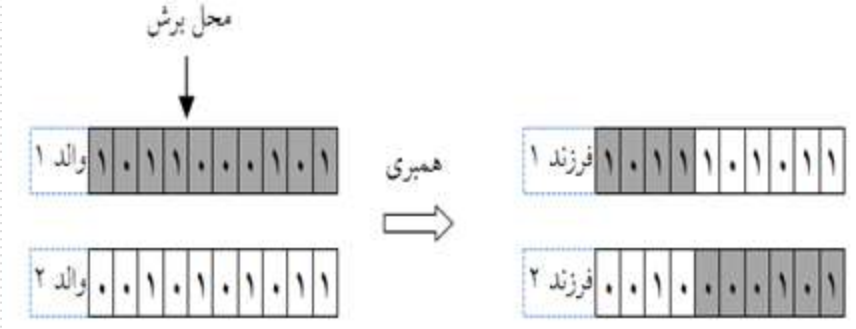


عملگر همبري

برای هر زوج

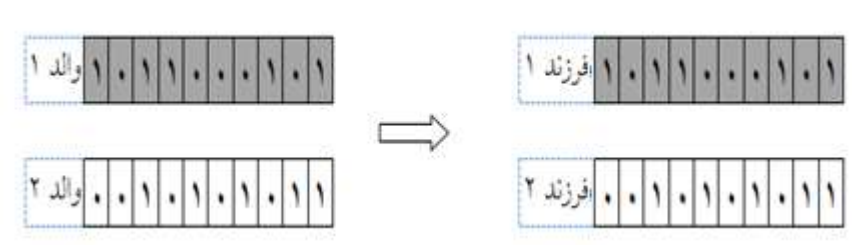
If rand<Pc

همبری



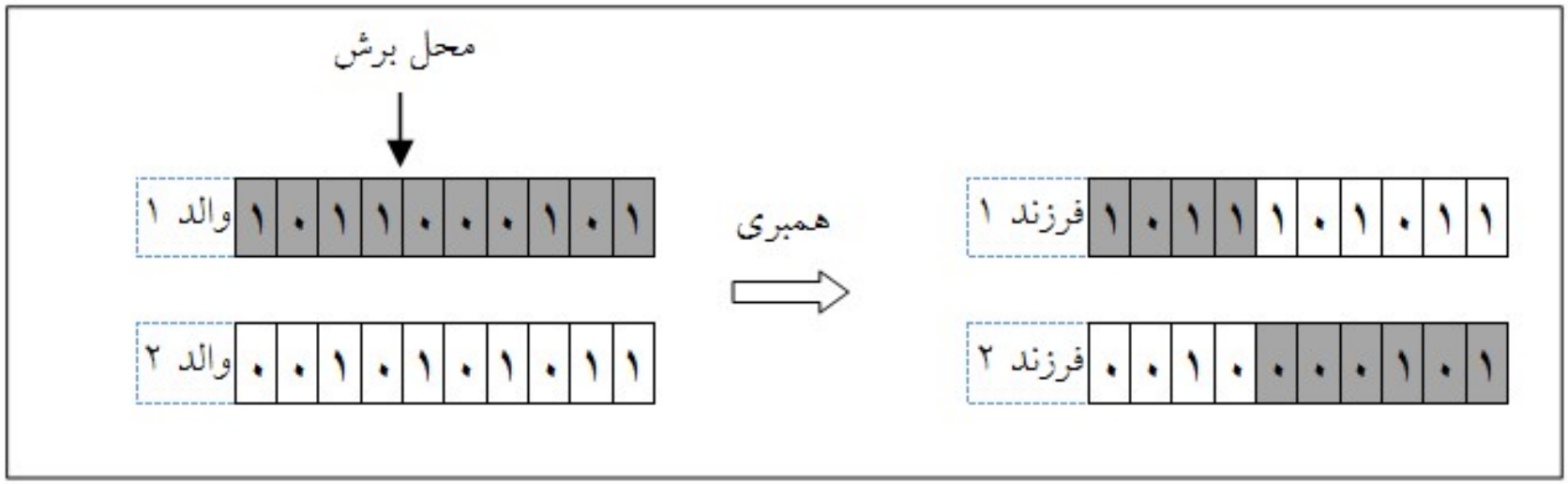
Else

انتقال



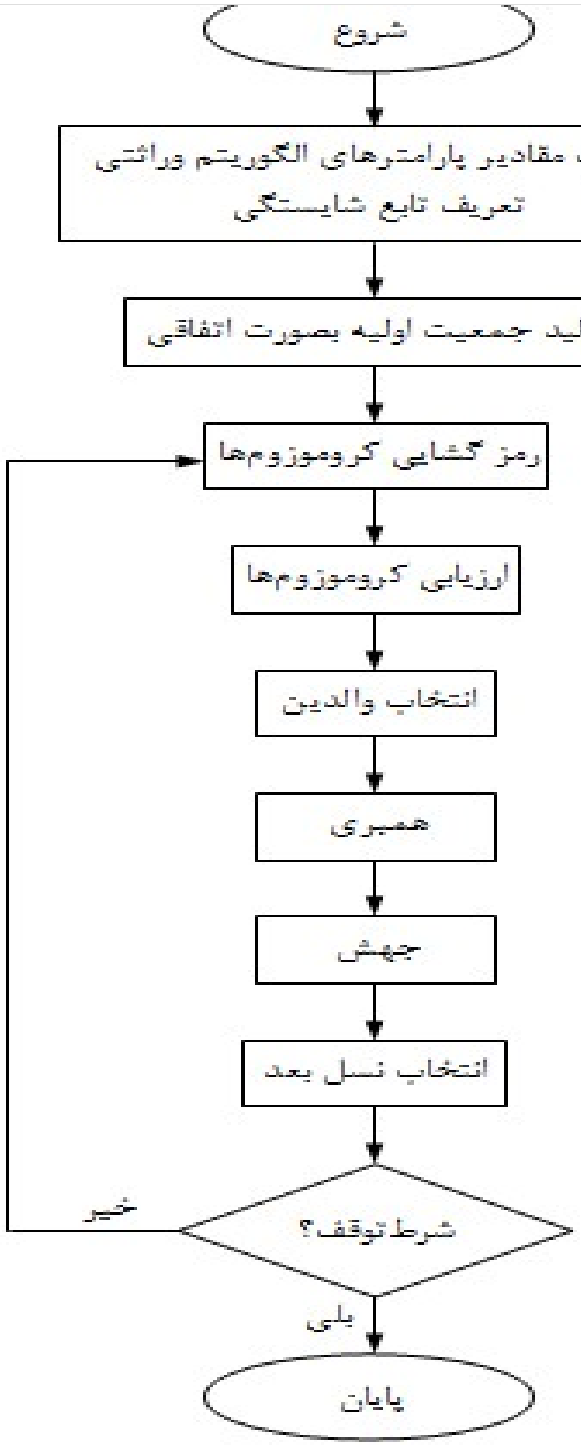
end

عملگر همبري تک نقطه اي



Cross point

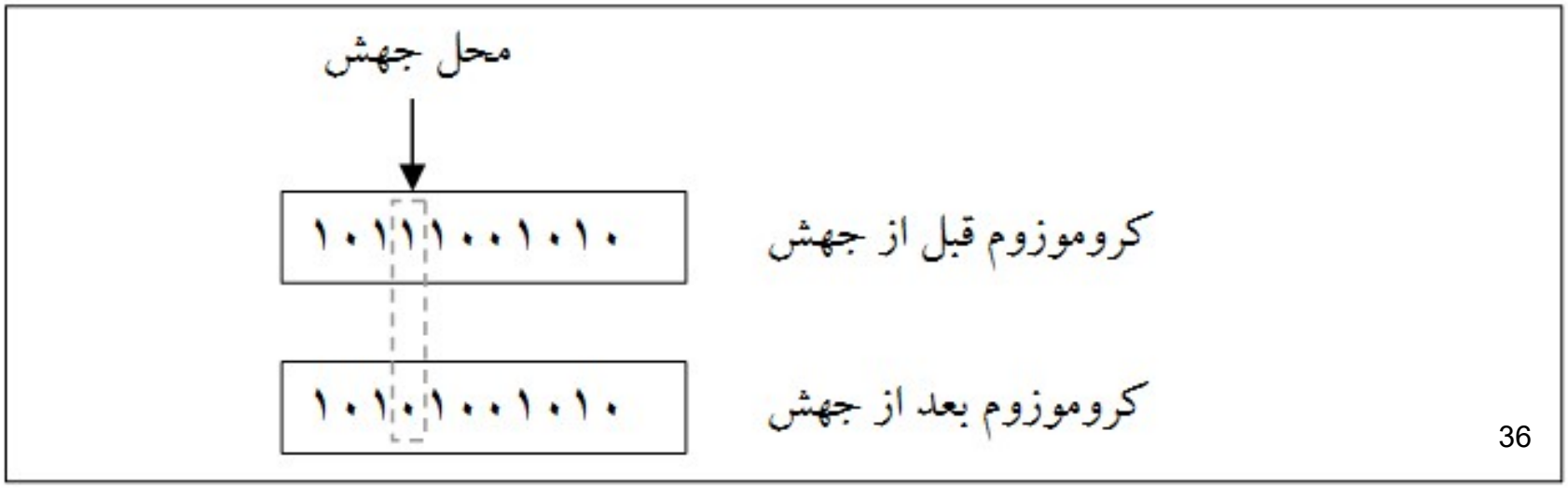
ساختار كلي الگوريتم وراثتي



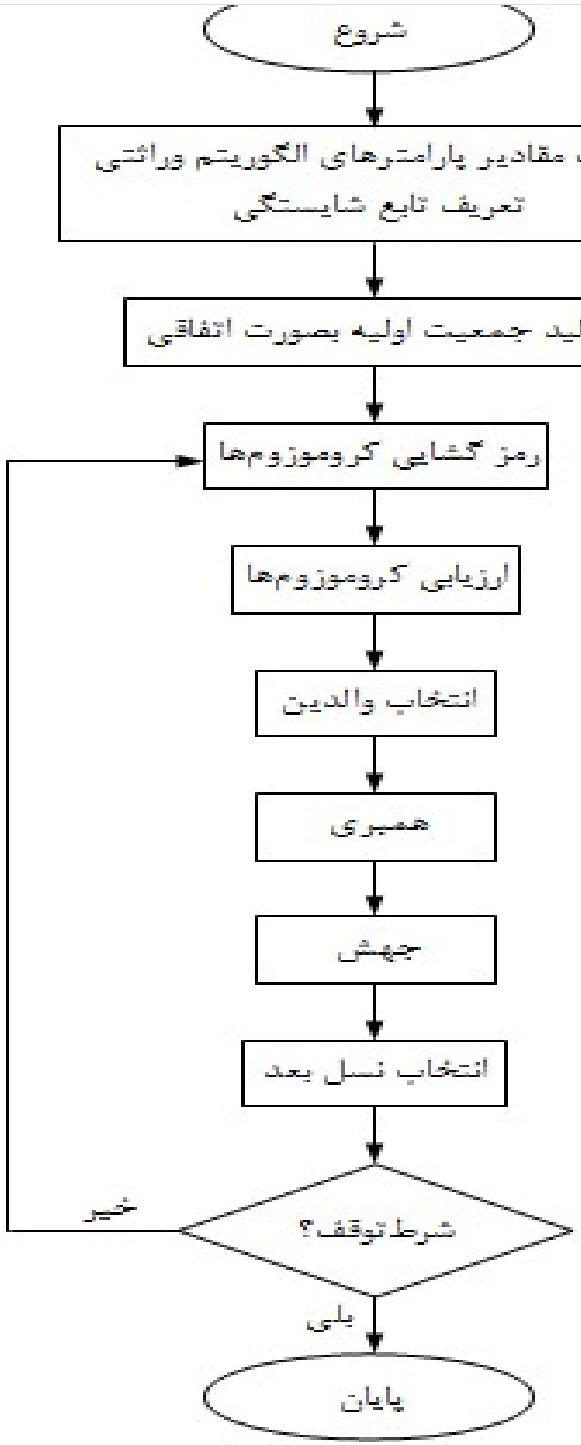
عملگر جهش Mutation

- این عملگر بطور عام بعد از عملگر همبري اعمال مي شود.
- احتمال رخداد جهش در الگوريتم وراثتي با P_m نمايش داده مي شود كه معمولاً بين 0.001 تا 0.01 انتخاب مي شود.
- عملگر جهش به هر يک از افراد جمعيت به تنهايي اعمال مي شود.

$$P_m \times L \times N$$



ساختار كلي الگوريتم وراثتي



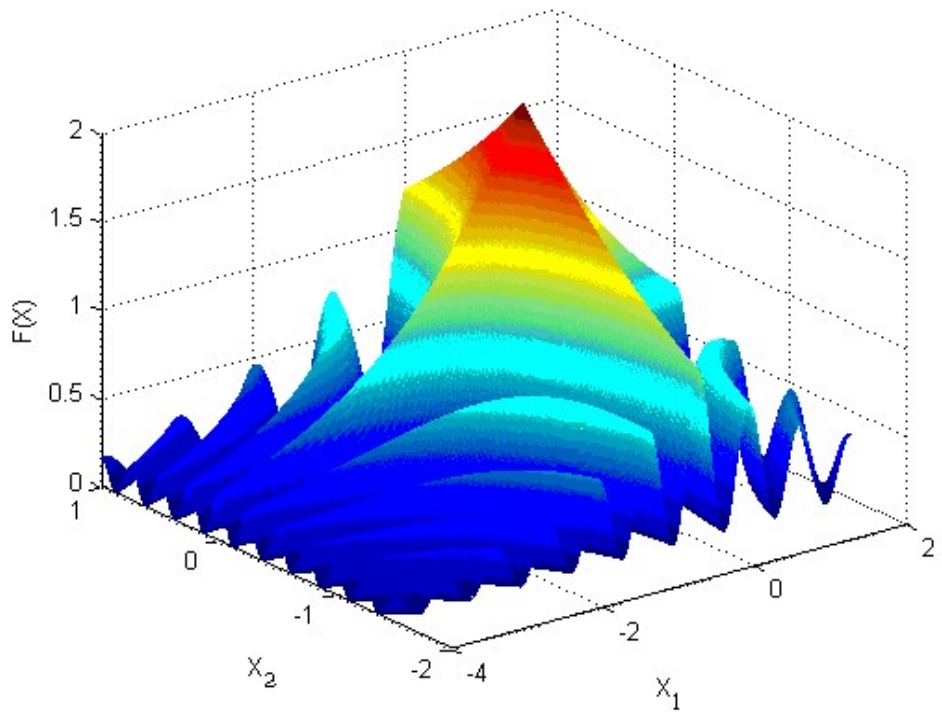
سایر مراحل

- جایگزینی نسل بعد
- چک کردن شرط توقف **Stopping criteria**

حل یک مسئله نمونه

$$f(X) = f(x_1, x_2) = \left(1 + \cos\left(2\pi\, x_1 x_2\right)\right) \times \exp\left[-\left(|x_1| + |x_2|\right)/2\right]$$

$$-4 \le x_1 \le 2\ ,\ -1.5 \le x_2 \le 1 \quad ,\quad f^{opt} = 2 \quad for \quad X^* = (0, 0)$$



مقدار دهي پارامترها و تعريف تابع شايستگي

عنوان	مقدار
P_C	۰/۹
P_m	۰/۰۰۵
N	۱۰
m	۲
L_i , i=1,2	۸

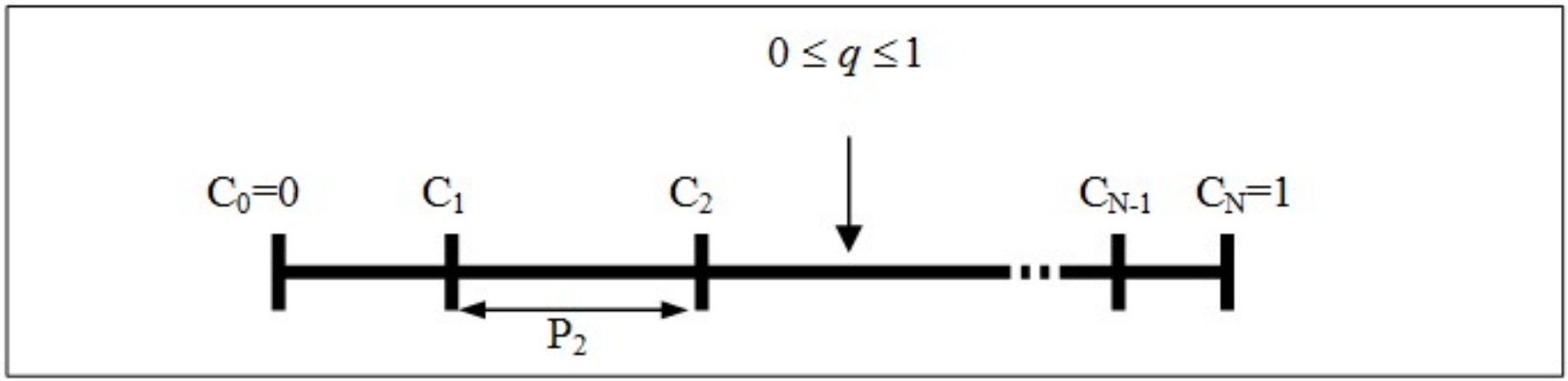
حل یک مسئله نمونه

شماره	جمعیت اولیه	متغیرها (نسل اول)	شایستگی	احتمال انتخاب	نرخ انتظار	احتمال تجمعی	انتخاب چرخ گردان
i	x_1 x_2	$\{x_1,x_2\}$	fit_i	P_i	$P_i \times N$	$C_i = \sum_{j=1}^{i} P_j$	
۱	۱۱۱۰۱۰۰۱۰۰۱۰۱۰۰۱	{۱/۴۸۲۲ ،−۱/۰۹۸۰}	۰/۰۸۳۶	۰/۰۱۰۸	۰/۱	۰/۰۱۰۸	۱
۲	۱۰۰۱۱۰۰۰۱۱۰۰۱۱۰۱	{−۰/۴۲۳۴ ، ۰/۵۰۹۷}	۰/۷۶۰۸	۰/۰۹۷۵	۰/۹	۰/۱۰۸۳	۰
۳	۱۰۱۰۰۱۱۱۱۱۱۰۰۱۰۰	{−۰/۰۷۰۶ ، ۰/۷۳۵۳}	۱/۳۰۱۵	۰/۱۶۶۹	۱/۷	۰/۲۷۵۲	۲
۴	۰۰۱۰۰۰۱۱۰۱۱۱۰۰۱۱	{−۳/۱۷۶۲ ،−۰/۳۷۲۵}	۰/۲۳۸۸	۰/۰۳۰۶	۰/۳	۰/۳۰۵۸	۰
۵	۱۰۱۰۱۰۱۰۰۱۱۱۰۰۰۱	{−۰/۰۰۰۲ ،−۰/۳۹۲۲}	۱/۶۴۳۷	۰/۲۱۰۸	۲/۱	۰/۵۱۶۶	۳
۶	۱۱۰۱۱۰۱۰۰۰۱۰۰۰۱۱	{۱/۱۲۹۴ ،−۱/۱۵۶۷}	۰/۲۰۸۱	۰/۰۲۰۶	۰/۲	۰/۵۳۷۲	۰
۷	۱۰۱۱۱۱۰۱۱۰۱۱۰۰۰۱	{۰/۴۴۷۲ ، ۰/۲۳۵۳}	۱/۲۷۲۱	۰/۱۶۳۱	۱/۶	۰/۷۰۰۳	۱
۸	۱۰۰۰۰۱۰۰۰۰۱۱۰۰۰۰	{−۰/۸۹۴۴ ،−۱/۰۲۹۵}	۰/۷۱۷۹	۰/۰۹۲۰	۰/۹	۰/۷۹۲۳	۱
۹	۱۱۰۰۰۰۰۱۰۰۰۱۱۱۰۱	{۰/۵۴۱۴ ،−۱/۲۱۵۷}	۰/۱۸۸۹	۰/۰۲۴۱	۰/۲	۰/۸۱۶۴	۰
۱۰	۱۰۱۱۱۱۰۰۱۰۱۰۱۱۰۰	{۰/۴۲۳۸ ، ۰/۱۸۶۳}	۱/۳۸۵۴	۰/۱۷۷۷	۱/۸	۱	۲
جمع	--	--	۷/۸۰۰۷	۱	۱۰	---	۱۰

احتمال تجمعي و چرخ گردان

$$C_i = \sum_{j=1}^{i} P_j$$

$$C_{i-1} \le q < C_i \quad , \quad C_0 = 0 \quad , \quad i = 1,2,....N$$

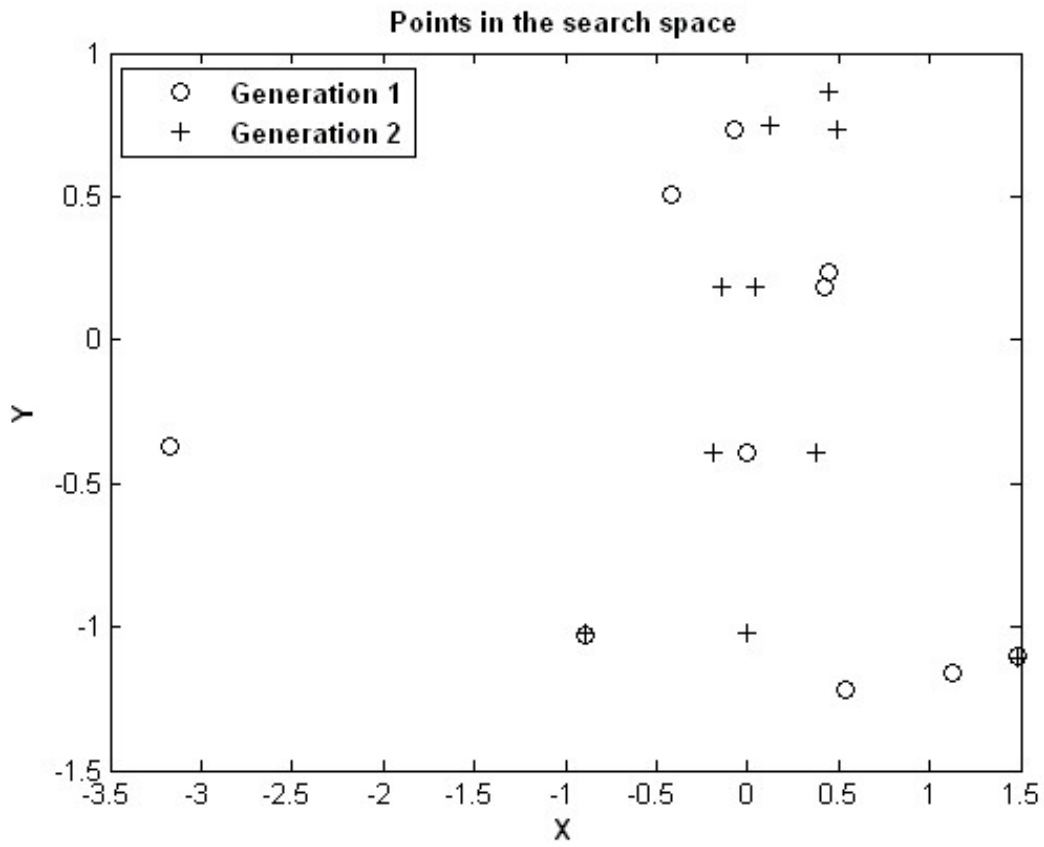


پیاده سازی نرم افزاری چرخ گردان

حل یک مسئله نمونه

شماره	حوضچه ازدواج		شماره جفت	نقطه برش	عملگر همبری		عملگر جهش (جمعیت جدید)		متغیرها (نسل دوم)
i	x_1	x_2			x_1	x_2	x_1	x_2	$\{x_1,x_2\}$
۱	۱۱۱۰۱۰۰۱۰۰۱۰۱۰۰۱		۸	۱۵	۱۱۱۰۱۰۰۱۰۰۱۰۱۰۰۰		۱۱۱۰۱۰۰۱۰۰۱۰۱۰۰۰		{۱/۴۸۲۲ ،−۱/۱۰۷۸}
۲	۱۰۱۰۰۱۱۱۱۱۱۰۰۱۰۰		۱۰	۶	۱۰۰۰۰۱۰۰۰۰۱۱۰۰۰۱		۱۰۰۰۰۱۰۰۰۰۱۱۰۰۰۱		{−۰/۸۹۴۴ ،−۱/۰۱۹۵}
۳	۱۰۱۰۰۱۱۱۱۱۱۰۰۱۰۰		۶	۵	۱۰۱۰۰۱۰۰۱۰۱۰۱۱۰۰		۱۰۱۰۰۱۰۰۱۰۱۰۱۱۰۰		{−۰/۱۴۱۴ ،۰/۱۸۶۳}
۴	۱۰۱۰۱۰۱۰۰۱۱۱۰۰۰۱		۷	۹	۱۰۱۱۱۱۱۱۱۱۱۱۰۰۱۰۰		۱۰۱۱۱۱۱۱۱۱۱۰۰۱۰۰		{۰/۴۹۴۰ ،۰/۷۳۵۳}
۵	۱۰۱۰۱۰۱۰۰۱۱۱۰۰۰۱		۹	۵	۱۰۱۰۰۰۱۰۰۱۱۱۰۰۰۱		۱۰۱۰۰۰۱۰۰۱۱۱۰۰۰۱		{−۰/۱۸۸۲ ،−۰/۳۹۲۲}
۶	۱۰۱۰۱۰۱۰۰۱۱۱۰۰۰۱		۳	۵	۱۰۱۰۱۱۱۱۱۱۱۱۰۰۱۰۰		۱۰۱۰۱۱۱۱۱۱۱۰۰۱۰۱		{۰/۱۱۷۸ ،۰/۷۴۵۰}
۷	۱۰۱۱۱۱۰۱۱۰۱۱۰۰۰۱		۴	۹	۱۰۱۰۱۰۱۰۰۰۱۱۰۰۰۱		۱۰۱۰۱۰۱۰۰۰۱۱۰۰۰۱		{۰/۰۰۰۲ ،۱/۰۱۹۵}
۸	۱۰۰۰۰۱۰۰۰۰۱۱۰۰۰۰		۱	۱۵	۱۰۱۱۱۱۰۱۱۱۱۱۰۰۰۱		۱۰۱۱۱۱۰۱۱۱۱۱۰۰۰۱		{۰/۴۴۷۲ ،۰/۸۶۲۸}
۹	۱۰۱۱۱۱۰۰۱۰۱۰۱۱۰۰		۵	۵	۱۰۱۰۱۱۰۰۱۰۱۰۱۱۰۰		۱۰۱۰۱۱۰۰۱۰۱۰۱۱۰۰		{۰/۰۴۷۰ ،۰/۱۸۶۳}
۱۰	۱۰۱۱۱۱۰۰۱۰۱۰۱۱۰۰		۲	۶	۱۰۱۱۱۰۱۰۰۱۱۱۰۰۰۱		۱۰۱۱۱۰۱۰۰۱۱۱۰۰۰۱		{۰/۳۷۶۴ ،−۰/۳۹۲۲}

حل یک مسئله نمونه



ارزيابي كارآيي الگوريتم وراثتي

- ارزيابي كارآيي الگوريتم وراثتي يا هر الگوريتم ابتكاري ديگر، از طريق تحليل همگرايي و زمان رسيدن به پاسخ بهينه بررسي مي‌شود. براي مشاهده وضعيت پيشرفت الگوريتم و پايش معيارهاي فوق از روشهاي ترسيمي كمک گرفته مي‌شود
- متوسط شايستگي جمعيت mean-fitness
- بهترين جواب ديده شده تا آن لحظه best-so-far

ارزیابی کارایی الگوریتم

t=1 t=2 ..... t=T

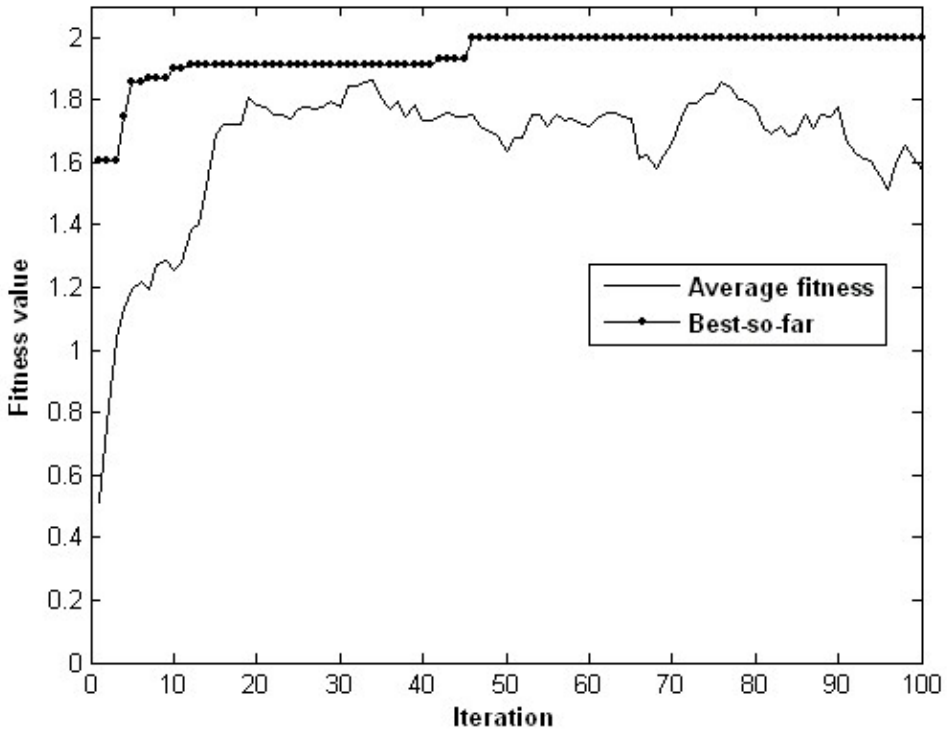
$$\begin{bmatrix} fit_1(1) \\ fit_2(1) \\ fit_3(1) \\ fit_4(1) \\ . \\ . \\ . \\ fit_{N-1}(1) \\ fit_N(1) \end{bmatrix} \quad \begin{bmatrix} fit_1(2) \\ fit_2(2) \\ fit_3(2) \\ fit_4(2) \\ . \\ . \\ . \\ fit_{N-1}(2) \\ fit_N(2) \end{bmatrix} \quad \cdots\cdots\cdots\cdots \quad \begin{bmatrix} fit_1(T) \\ fit_2(T) \\ fit_3(T) \\ fit_4(T) \\ . \\ . \\ . \\ fit_{N-1}(T) \\ fit_N(T) \end{bmatrix}$$

mean
Fitness=[mf(1) mf(2) ..... mf(T)] میانگین هر تکرار

best
So far =[bsf(1) bsf(2) ..... bsf(T)] بهترین تا هر تکرار

ارزيابي كارآيي الگوريتم وراثتي

- در الگوريتم پياده سازي شده، از جمعيتي شامل 30 كروموزوم با طول 16 بيت استفاده شده است. نرخ همبري و جهش به ترتيب 0.9 و 0.005 در نظر گرفته شده است و الگوريتم براي 100 تكرار اجرا شده است.



آیا مقایسه الگوریتم ها با یک اجرا عادلانه است؟

ارزيابي و مقايسه كارآيي

- الگوریتم ها تصادفی هستند. و با هر بار اجرا جواب های متفاوتی می دهند.
- برای مقایسه عادلانه، هر الگوریتم را چند بار اجرا کرده و میانگین نتایج را گزارش می کنند.

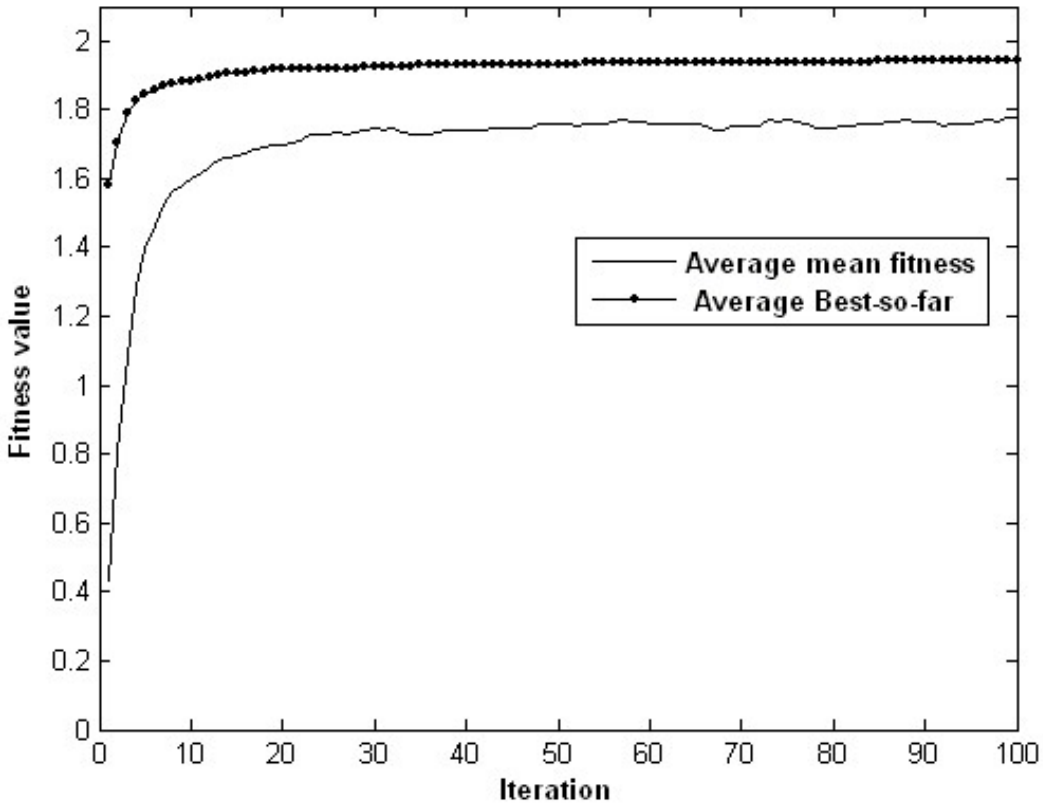
ارزیابی در چند اجرای مستقل الگوریتم

Run1	[bsf(1)	bsf(2)	……	bsf(T)]
Run2	[bsf(1)	bsf(2)	……	bsf(T)]
Run3	[bsf(1)	bsf(2)	……	bsf(T)]
.				
.				
.				
Run20	[bsf(1)	bsf(2)	……	bsf(T)]

Average best
So far =[absf(1) absf(2) …… absf(T)]

ارزيابي كارآيي الگوريتم وراثتي

- متوسط معيارهاي قبلي



ارزيابي كارآيي الگوريتم وراثتي

- معيار برخط
- معيار برون خط
- معيار سرعت همگرايي

$$f_{on}(T) = \frac{1}{T}\sum_{t=1}^{T}\left[\frac{1}{N}\sum_{i=1}^{N} fit_i(t)\right]$$

$$f_{off}(T) = \frac{1}{T}\sum_{t=1}^{T} fit^{\max}(t)$$

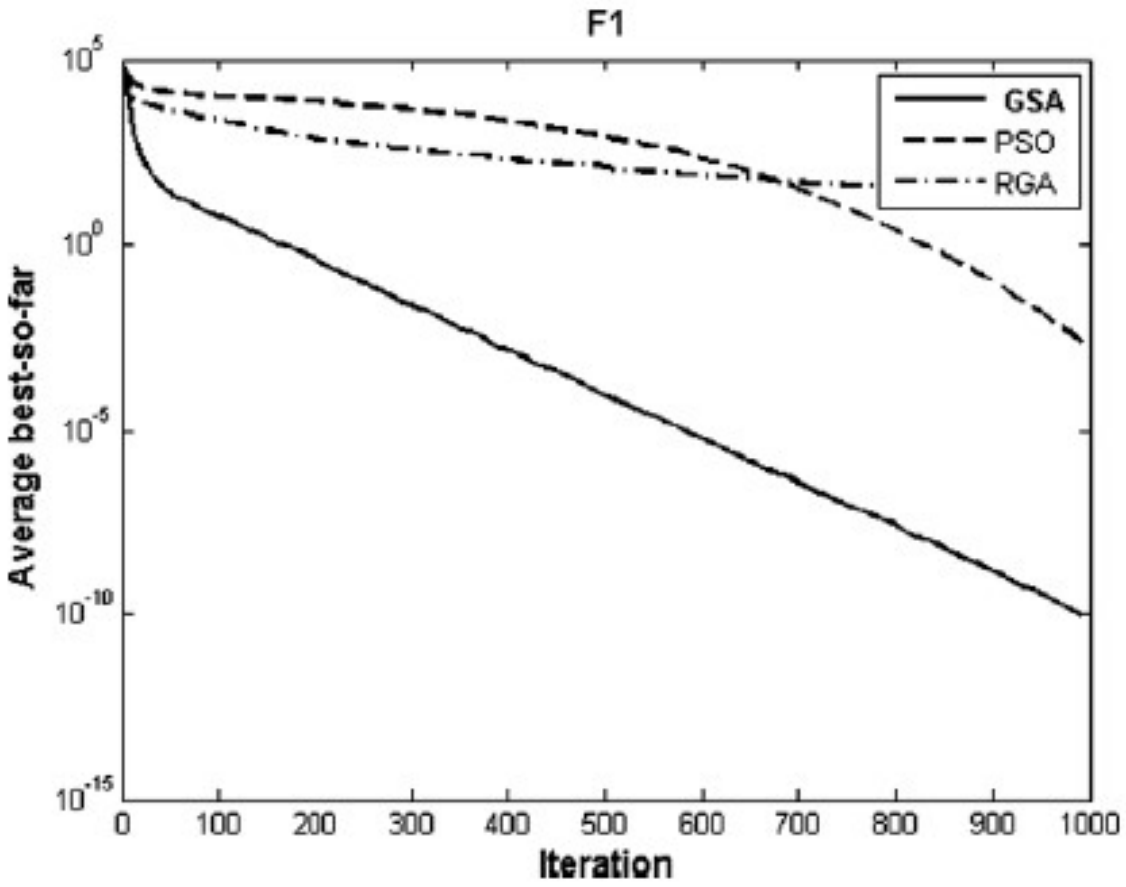
$$V = Ln\sqrt{\frac{f^{\max}(T)}{f^{\max}(1)}}$$

مقایسه کارآیي الگوریتم های مختلف

- رسم کردن نمودارهای الگوریتم ها در یک شکل
- استفاده از جداول

مقایسه کارآیي الگوریتم های مختلف

- مقایسه با رسم کردن نمودارهای الگوریتم ها در یک شکل



مقایسه کارآیی الگوریتم های مختلف

- مقایسه با استفاده از جداول

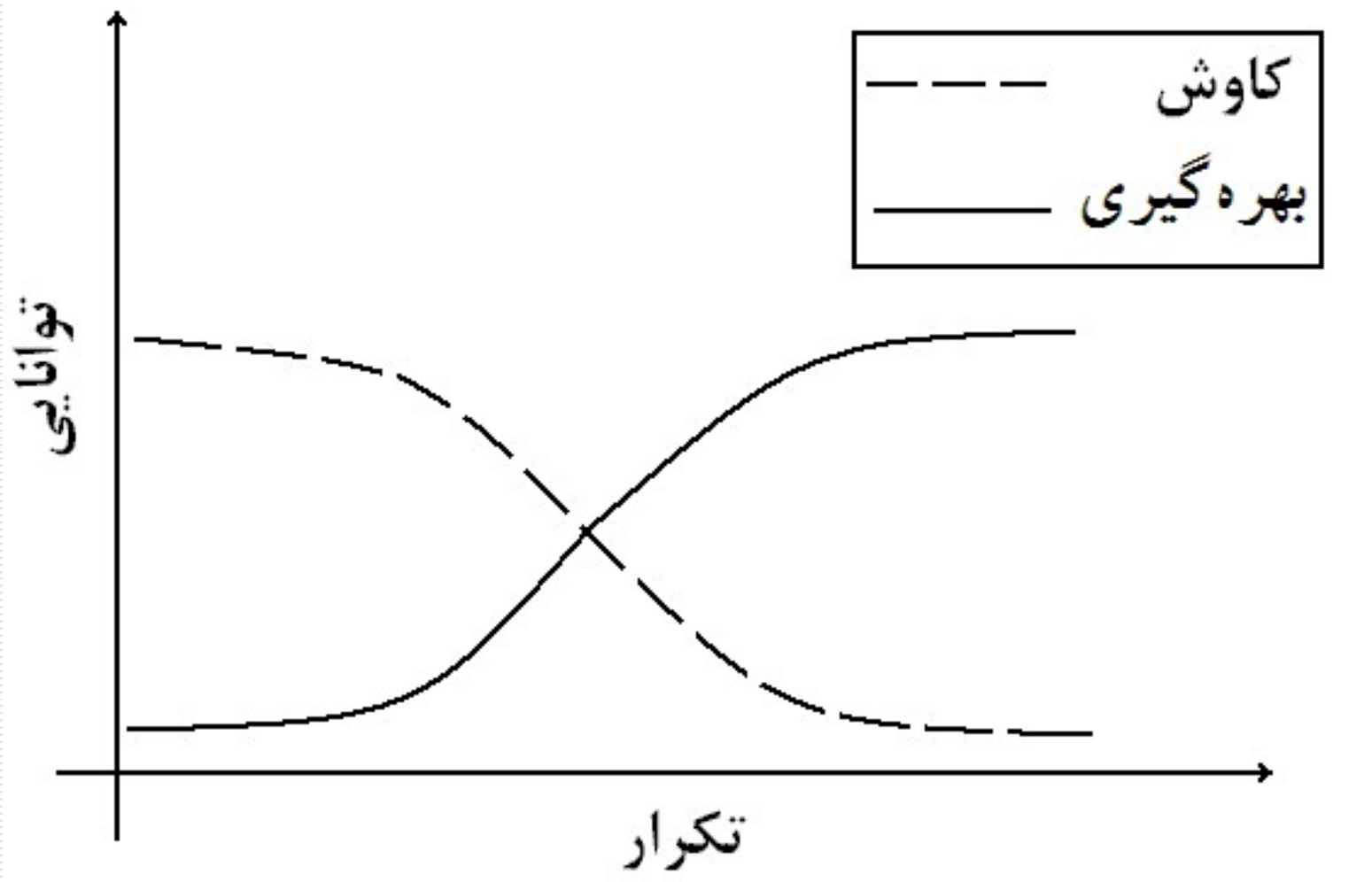
Run1	[bsf(1)	bsf(2)	.....	bsf(T)]
Run2	[bsf(1)	bsf(2)	.....	bsf(T)]
Run3	[bsf(1)	bsf(2)	.....	bsf(T)]
.				
.				
.				
Run20	[bsf(1)	bsf(2)	.....	bsf(T)]

مقایسه کارآیي الگوریتم های مختلف

- مقایسه با استفاده از جداول

Algorithm	Mean ± Var	Best	Time (sec)
A	1.0 ± 0.2	1.22	2
B	1.7 ± 0.1	1.72	1.5
C	0.9 ± 0.05	0.95	2.5

کاوش و بهره گيري



- Convergence
- Exploration / diversification
- Exploitation / intensification

نوشتن یک برنامه وراثتي باینري ساده

```
%%%% MyGenetic.m %%%%  نام برنامه اصلی
clc; %  صفحه نمایش را پاک کن
clear all; %  کلیه متغیرهای موجود در حافظه را پاک کن
%%%% Initialization %%%%در این قسمت مقدار دهی اولیه پارامترهای الگوریتم انجام می‌شود
N=50; %  تعداد کروموزومها
Pc=0.9;%  نرخ همبری
Pm=0.005;%  نرخ جهش
ITER=100;%  تعداد دفعات تکرار الگوریتم
m=2;%  تعداد متغیرها
BS=[10 10];%  برداری با طول تعداد متغیرها که در آن تعداد بیت مورد نیاز هر متغیر لحاظ شده است
L=sum(BS);%  تعداد کل بیتهای در نظر گرفته شده برای یک کروموزوم که برابر مجموع بیتهای متغیرهای
%  مساله است
Lo=[-4 -1.5];%  برداری با طول تعداد متغیرها که در آن حد پایین هر یک از متغیرها لحاظ شده است
Hi=[2  1];%  برداری با طول تعداد متغیرها که در آن حد بالای هر یک از متغیرها لحاظ شده است
Population=round(rand(N,L));% تولید جمعیت اولیه بصورت اتفاقی با تابع توزیع احتمال یکنواخت
```

نوشتن یک برنامه وراثتي باینري ساده

```
best_so_far=[];
Average_fitness=[];
for it=1: ITER  %          الگوریتم تا تعداد دفعات از پیش تعیین شده، تکرار می‌شود
[real_val]=chrom_decode(Population,N,L,BS,m,Lo,Hi);
 [selection_probability,fit,ave_fit,max_fit, opt_sol]=fit_eval(real_val,N,m);

if it==1
   best_so_far(it)=max_fit;
   final_sol=opt_sol;
elseif max_fit>best_so_far(it-1)
   best_so_far(it)=max_fit;
   final_sol=opt_sol;
else
   best_so_far(it)= best_so_far(it-1);
end
Average_fitness(it)=ave_fit;
%%%%
[mating_pool]=g_roulette_wheel(Population,N,selection_probability);
 [new_pop]=g_crossover(mating_pool,Pc,N,L);
 [population]=g_mutation(new_pop,Pm,N,L);
end
display('final Solution      optimum fitness');
result=[final_sol,best_so_far(end)]
x=1:ITER;
figure,plot(x,best_so_far,'k',x,Average_fitness,'.-k');
xlabel('Generation');
ylabel('Fitness Function')
legend('Best-so-far','Average fitness')
```

نوشتن یک برنامه وراثتي باینري ساده

```
%%%% chrom_decode.m %%%%
% این تابع مقادیر رمز شده باینری کروموزومها را گرفته و مقادیر واقعی رمزگشایی شده آنها را بر می‌گرداند
function [real_val]=chrom_decode(Population,N,L,BS,m,Lo,Hi)
% ابتدا باید برنامه‌ای نوشته شود که قسمت ابتدایی و انتهایی هر متغیر(ژن) در کروموزوم مشخص شود. در ادامه
% ابتدا این قسمت از برنامه نوشته می‌شود
real_val=[];
STED(1)=1;
for i=2:m+1
   STED(i)=STED(i-1)+BS(i-1);
end
%%%%%%%%%%%%%%%
for j=1:m
  x=BS(j)-1:-1:0;
  Pow2x=2.^x; % ساختن یک رشته عددی از توانهای ۲ برای دهدهی کردن عدد باینری
   for i=1:N
      gene=Population(i,STED(j): STED(j+1)-1);% جدا کردن یک متغیر از یک کروموزوم
      Var_norm=sum(Pow2x.*gene)/(2^BS(j)-1);%مقدار نرمال متغیر
      real_val(i,j)=Lo(j)+(Hi(j)-Lo(j))*Var_norm;%مقدار حقیقی متغیر
   end
end
return;
%%%%%%%%%%%%%%%
```

نوشتن یک برنامه وراثتي باینري ساده

```
%%%%% fit_eval %%%%%
این تابع مقادیر واقعی متغیرها را گرفته، هر راه حل را ارزیابی کرده و مقادیر شایستگی آن را بر می گرداند%
function[selection_probability,fit,ave_fit,max_fit, opt_sol]=fit_eval(real_val,N,m)
for i=1:N
   x(1)=real_val(i,1);
   x(2)=real_val(i,2);
   fit(i)=(1+cos(2*pi*x(1)*x(2)))*exp(-(abs(x(1))+abs(x(2)))/2);%محاسبه مقدار شایستگی
end
selection_probability=fit/sum(fit);%                    محاسبه احتمال انتخاب
ave_fit=mean(fit);%                                     محاسبه متوسط شایستگی جمعیت
[max_fit,max_loc]=max(fit);%                            محاسبه مقدار بیشینه شایستگی
opt_sol=real_val(max_loc,:);
return;
%%%%%%%%%%%%%%%%%%
```

نوشتن یک برنامه وراثتي باینري ساده

```
%%%% g_roulette_wheel.m %%%%
این زیر برنامه والدین را برای زاد و ولد و تولید مثل بر مبنای شایستگی و روش چرخ گردان انتخاب می‌کند%
function [mating_pool]=g_roulette_wheel(Population,N,selection_probability)
cdf(1)=selection_probability(1); محاسبه احتمال تجمعی جمعیت%
for i=2:N
  cdf(i)=cdf(i-1)+selection_probability(i);
end
 %%%%% پیاده سازی چرخ گردان
for i=1:N
   r=rand;
   for j=1:N
      if r<=cdf(j)
         mating_pool(i, :)=Population(j, :);
         break;
      end
   end
end
return;
%%%%%%%%%%%%%%%%
```

نوشتن یک برنامه وراثتي باینري ساده

```
%%%%% g_crossover.m %%%%%
این تابع عملگر همبری یک نقطه‌ای را روی والدین انتخاب شده اعمال می‌کند%
function [new_pop]=g_crossover(mating_pool,Pc,N,L);
parent_num=randperm(N);
for j=1:2:N
    pointer1= parent_num (j);%                    شمارنده مربوط به والد ۱
    pointer2= parent_num (j+1);%                  شمارنده مربوط به والد۲
    cut_point= round(1+(L-2)*rand);%              انتخاب محل برش بصورت اتفاقی
    off1=mating_pool(pointer1,:);%                کپی والد اول را در فرزند اول قرار بده
    off2=mating_pool(pointer2,:); %               کپی والد دوم را در فرزند دوم قرار بده
  if rand <Pc    %                              با نرخ احتمال Pc عمل همبری را انجام بده
        temp=off2;
        off2(cut_point+1:end)=off1(cut_point+1:end);
        off1(cut_point+1:end)=temp(cut_point+1:end);
   end
new_pop(j,:)=off1;%                             فرزندان تولید شده را به جمعیت جدید منتقل کن
new_pop(j+1,:)=off2;
end
return;
%%%%%%%%%%%%%%%%%%
```

نوشتن یک برنامه وراثتي باینري ساده

```
%%%% g_mutaion.m %%%%
این تابع، عملگر جهش را روی جمعیت تولیدشده، اعمال می‌کند%
function [Population]=g_mutation(new_pop,Pm,N,L);
mask=rand(N,L)<=Pm;
Population=xor(new_pop,mask);
return;
%%%%%%%%%%%%%%%%%%
```