بسم الله الرحمن الرحیم

هوش مصنوعی پیشرفته

هوش تطوری

Evolutionary Intelligence

کاظم فولادی
دانشکده مهندسی برق و کامپیوتر
دانشگاه تهران

http://courses.fouladi.ir/aai

هوش مصنوعی

هوش تطوری

۱

مقدمه

محاسبات تطوری

EVOLUTIONARY COMPUTATION

محاسبات تطوری، فرآیند **تطور** را بر روی یک کامپیوتر شبیه‌سازی می‌کند.

نتیجه‌ی یک چنین شبیه‌سازی،
گردایه‌ای از الگوریتم‌های بهینه‌سازی بر اساس مجموعه‌ی ساده‌ای از قواعد است.

بهینه‌سازی به صورت تکراری کیفیت راه‌حل‌ها را بهبود می‌دهد
تا اینکه یک راه‌حل بهینه یا حداقل امکان‌پذیر پیدا شود.

تطور در طبیعت

EVOLUTION IN NATURE

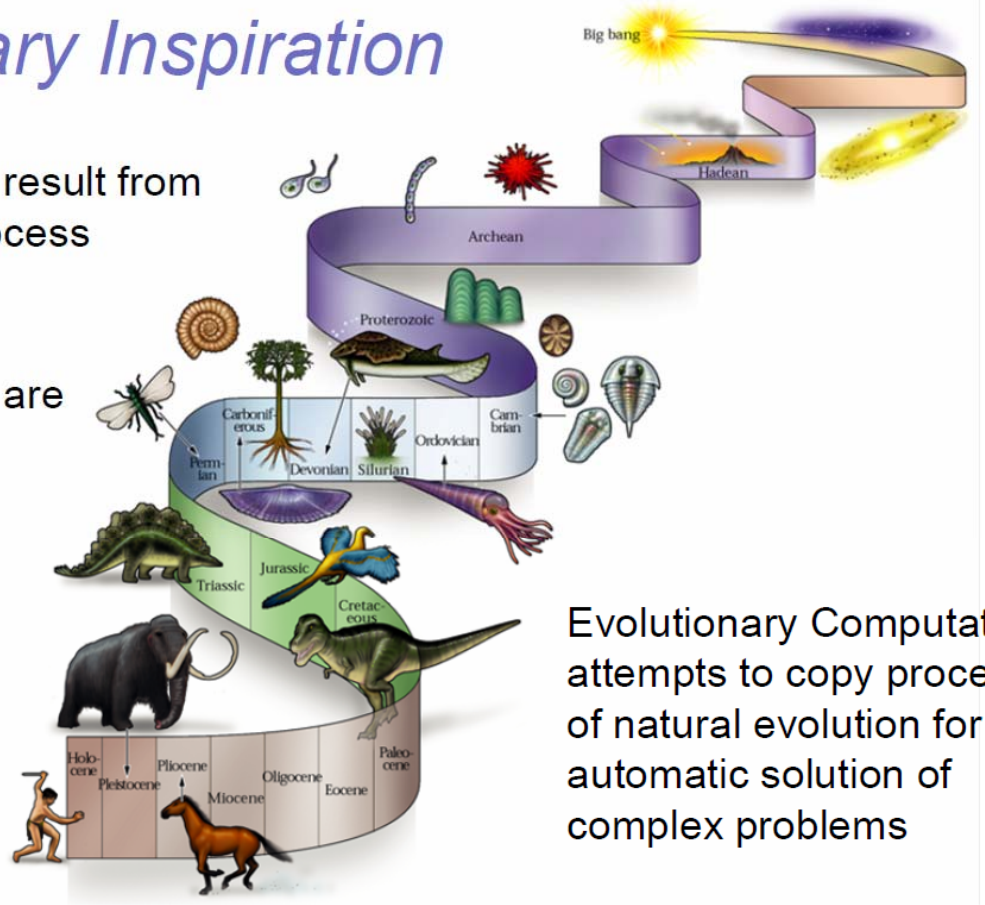
رفتار یک ارگانیسم فردی،
یک استنتاج استقرائی در مورد برخی جنبه‌های هنوز ناشناخته‌ی محیط آن است.

اگر در طول نسل‌های پی‌درپی، آن ارگانیسم بقا پیدا کند
می‌توانیم بگوییم:
آن ارگانیسم قادر به یادگیری برای پیش‌بینی تغییرات در محیط خودش است.

Evolutionary Inspiration

Biological systems result from an evolutionary process

Biological systems are
- robust
- complex
- adaptive



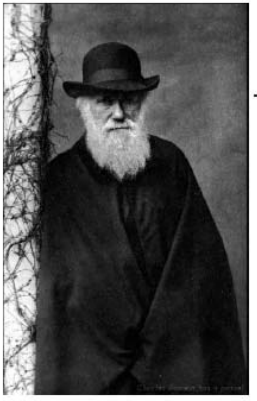
Evolutionary Computation attempts to copy process of natural evolution for automatic solution of complex problems

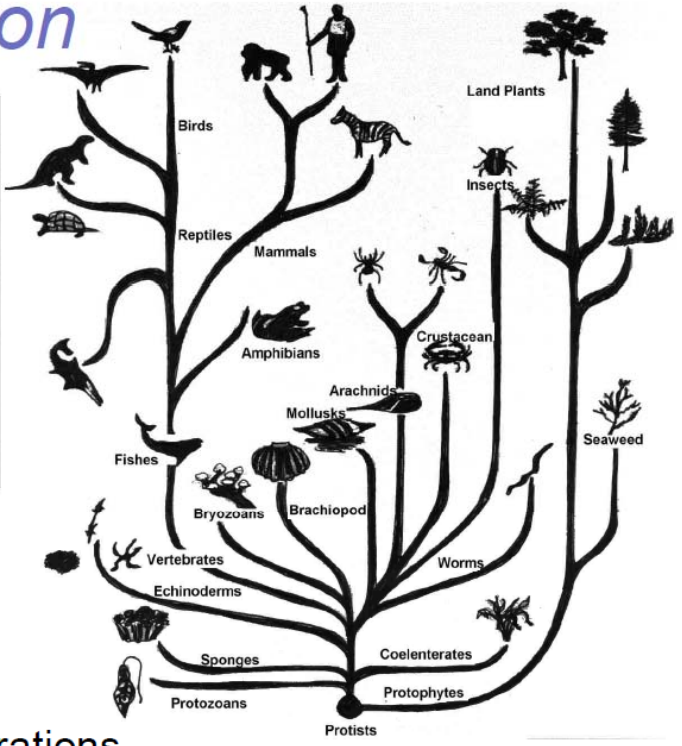
Does natural evolution generate increasingly complex systems?

The 4 Pillars of Evolution

All species derive from common ancestor

Charles Darwin, 1859
On the Origins of Species





Population
Group of several individuals
Diversity
Individuals have different characteristics
Heredity
Characteristics are transmitted over generations
Selection
- Individuals make more offspring than the environment can support
- Better at food gathering = better at surviving = make more offspring

روی‌کرد تطوری

EVOLUTIONARY APPROACH

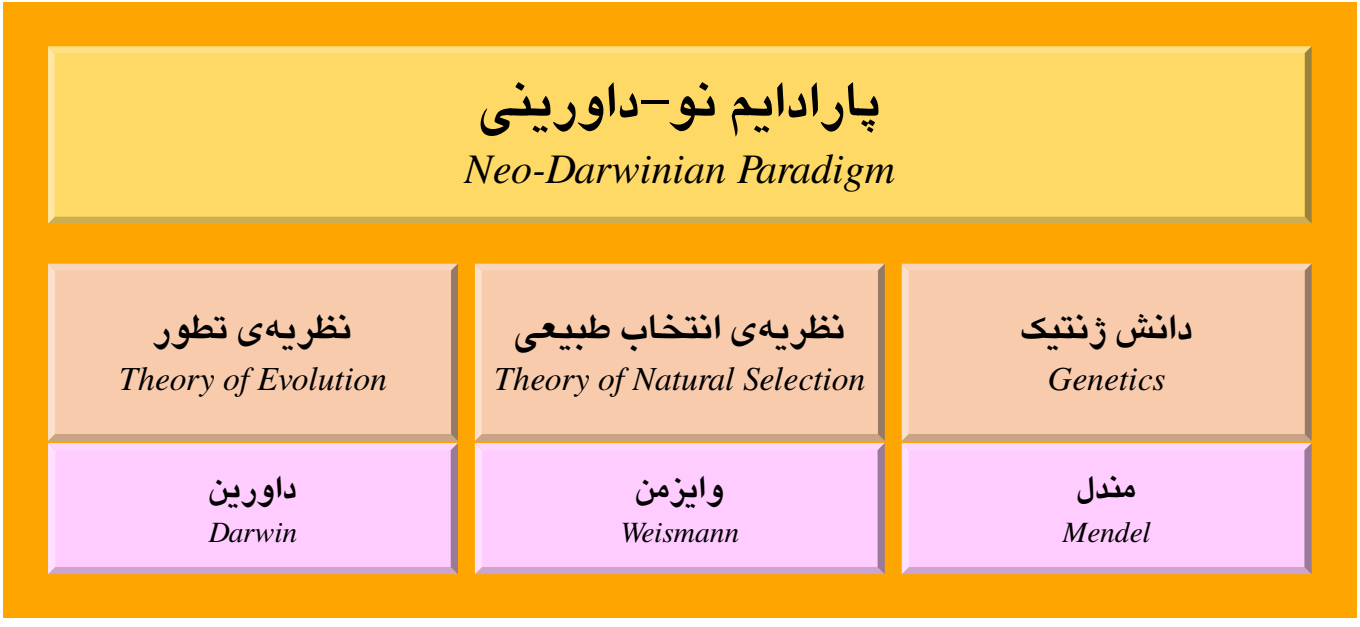
روی‌کرد تطوری، بر اساس مدل‌های محاسباتی انتخاب طبیعی و ژنتیک است.

محاسبات تطوری
Evolutionary Computation

استراتژی‌های تطور *Evolution Strategy*	**برنامه‌نویسی ژنتیک** *Genetic Programming*	**الگوریتم‌های ژنتیک** *Genetic Algorithms*
تطور تفاضلی *Differential Evolution*	**برنامه‌نویسی تطوری** *Evolutionary Programming*	**الگوریتم‌های فرهنگی** *Cultural Algorithms*

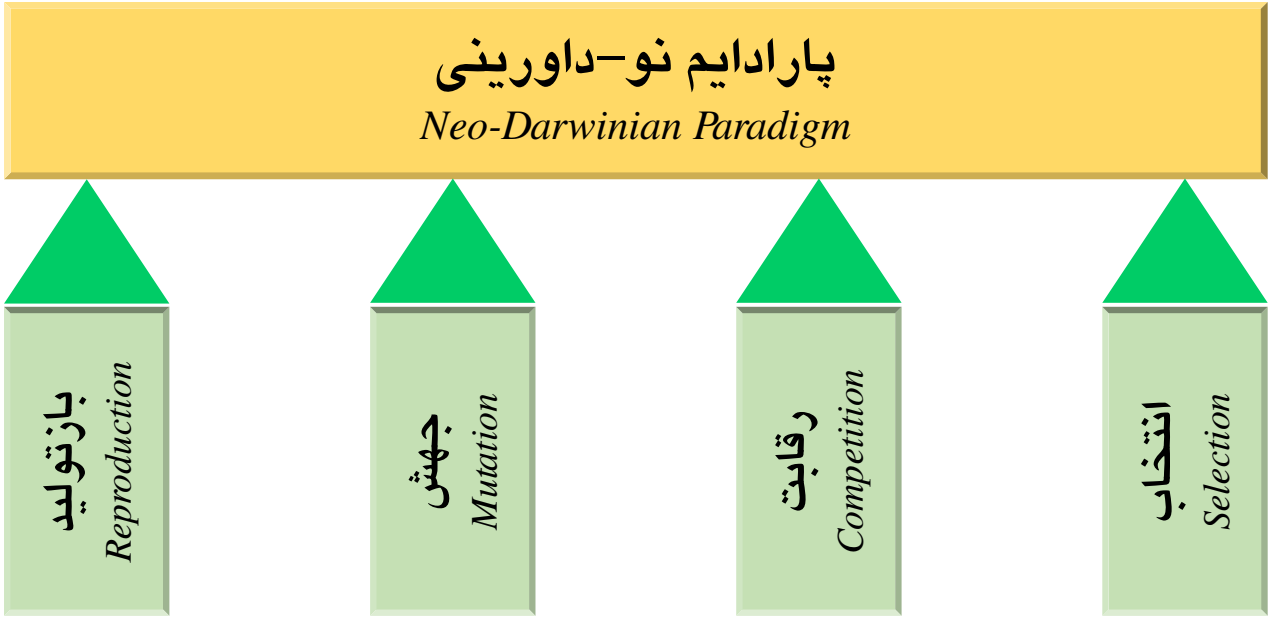
پارادایم نو–داروینی

NEO-DARWINIAN PARADIGM



پارادایم نو–داروینی

مبناها



برازش و بقا

FITNESS AND SURVIVAL

تطور می‌تواند به‌عنوان فرآیندی دیده شود
که منجر به حفظ قابلیت بقای یک جمعیت و بازتولید در یک محیط خاص می‌شود.
(قابلیت برازش تطوری)

برازش تطوری می‌تواند به‌عنوان معیاری از قابلیت یک ارگانیسم
برای پیش‌دستی در برابر تغییرات در محیط آن دیده شود.

برازش، یا معیار کمی قابلیت پیش‌بینی تغییرات محیطی و پاسخ مناسب به آن،
می‌تواند به‌عنوان کیفیتی که در حیات طبیعی بهینه می‌شود، دیده شود.

تولید جمعیت دارای برازش صعودی

GENERATING A POPULATION WITH INCREASING FITNESS

جمعیتی از خرگوش‌ها را در نظر بگیرید.

برخی خرگوش‌ها سریع‌تر از بقیه هستند،
و ممکن است بگوییم این خرگوش‌ها از برازش بالاتری برخوردار هستند:
زیرا آنها شانس بیشتری برای فرار از روباه‌ها، بقا و سپس تولید مثل دارند.

اگر دو والد برازش بالاتری داشته باشند،
شانس بهتری وجود دارد که ترکیبی از ژن‌های آنها فرزندی با برازش حتی بالاتر تولید کند.

در طول زمان، کل جمعیت خرگوش‌ها
برای برخورد با چالش‌های محیطی خود در مواجهه با روباه‌ها سریع‌تر می‌شود.

شبیه‌سازی تطور طبیعی

SIMULATION OF NATURAL EVOLUTION

همه‌ی روش‌های محاسبات تطوری،
تطور طبیعی را با

* ایجاد جمعیتی از افراد
* ارزیابی برازش آنها
* تولید یک جمعیت جدید از طریق عملیات ژنتیکی و
* تکرار این فرآیند به دفعات

شبیه‌سازی می‌کنند.

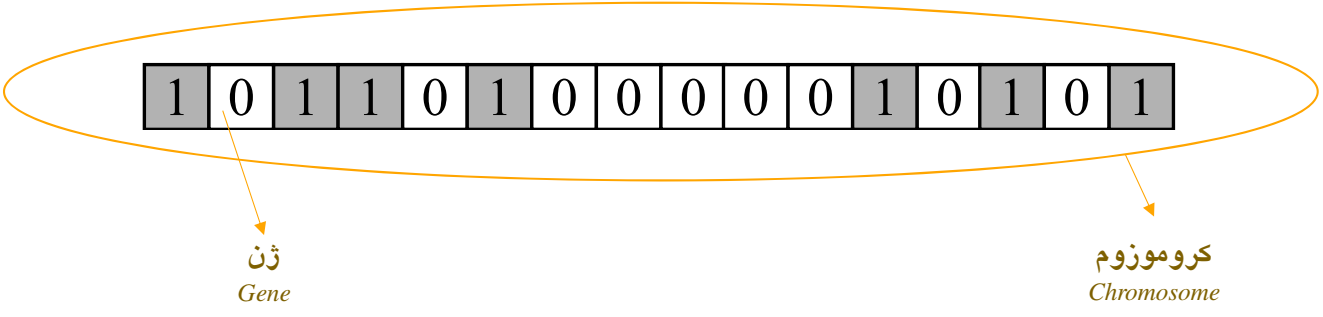
الگوریتم‌های ژنتیک

GENETIC ALGORITHMS (GAS)

معرفی رسمی توسط John Holland (1970s)

هدف هلند: وادار کردن کامپیوترها به انجام کاری که طبیعت انجام می‌دهد.

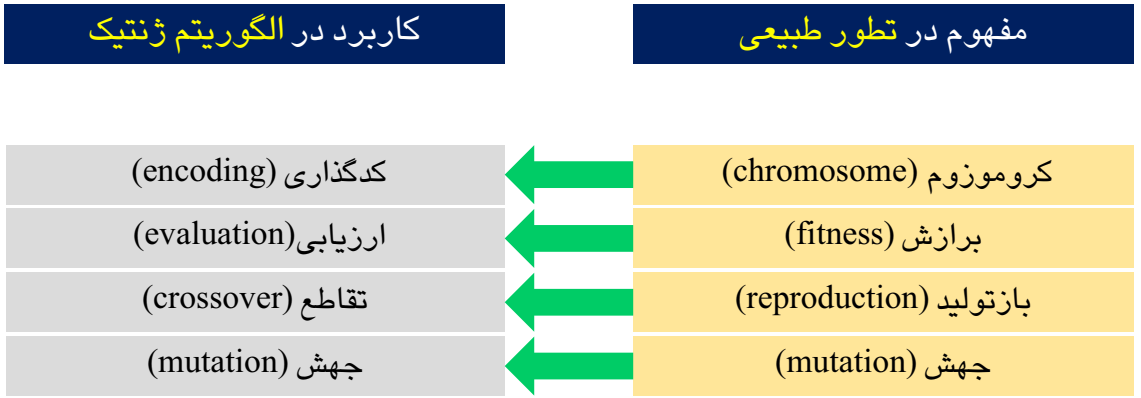
هلند، درگیر الگوریتم‌هایی برای دستکاری رشته‌هایی از ارقام دودویی بود.
این رشته از ارقام دودویی، با مفهوم **کروموزم** شناخته می‌شود؛
که هر عنصر آن یک **ژن** نامیده می‌شود.



الگوریتم‌های ژنتیک

مقایسه با تطور طبیعی

GENETIC ALGORITHMS (GAS)



الگوریتم ژنتیک پایه

گام‌های الگوریتم ژنتیک پایه

۱) بازنمایی متغیر مسئله (کروموزوم)؛ انتخاب اندازه‌ی جمعیت N ، احتمال تقاطع p_c و احتمال جهش p_m

↓

۲) تعریف یک تابع برازش (برای اندازه‌گیری کارآیی/برازش یک کروموزوم فرد)

↓

۳) ایجاد یک جمعیت از کروموزوم‌ها به اندازه‌ی N

↓

۴) محاسبه‌ی برازش هر کروموزم فردی

↓

۵) انتخاب **یک جفت** کروموزوم از جمعیت فعلی برای **تزویج** (احتمال انتخاب متناسب با برازش)

↓

۶) ایجاد یک جفت از کروموزوم‌های فرزند با به‌کارگیری عملگرهای ژنتیکی: **تقاطع** و **جهش**

↓

۷) قرار دادن کروموزوم‌های فرزند ایجاد شده در جمعیت جدید

↓

۸) تکرار از (۵) تا رسیدن اندازه‌ی جمعیت جدید به N.

↓

۹) جایگزینی جمعیت فعلی با جمعیت جدید

↓

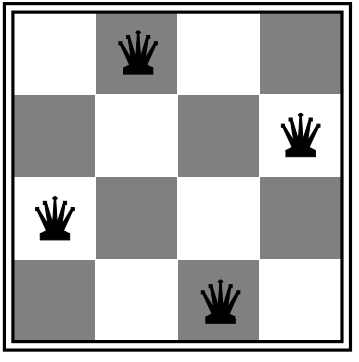
۱۰) تکرار فرآیند از (۴) تا ارضا شدن ضابطه‌ی خاتمه

الگوریتم‌های ژنتیک

کروموزوم (ژنوم)

کروموزوم *Chromosome*	یک ساختمان‌داده برای کدگذاری راه‌حل‌های مسئله (حالت‌ها) برای ذخیره‌سازی در کامپیوتر (معمولاً **رشته/بردار**)
ژنوم *Genome*	
فرد *Individual*	

مثال



$(3, 1, 4, 2)$

الگوریتم‌های ژنتیک

جمعیت

جمعیت *Population*	مجموعه‌ای از افراد (کروموزوم‌ها)

الگوریتم‌های ژنتیک

انتخاب

انتخاب *Selection*	انتخاب افراد (کروموزوم‌ها) برای جفت شدن

ضوابط متنوعی برای **گزینش بهترین افراد برای جفت شدن** وجود دارد.

الگوریتم‌های ژنتیک

تقاطع

تقاطع *Crossover*	ترکیب ضربدری دو کروموزوم والد برای تولید دو یا چند فرزند

تقاطع می‌تواند به صورت (**جنسی** یا **غیرجنسی**) / و حتی **تک‌فرزندی** تعریف شود.

الگوریتم‌های ژنتیک

جهش

جهش *Mutation*	تغییری تصادفی در یک کروموزوم

- جهش مقدار مشخصی **تصادفی بودن** به جستجو اضافه می‌کند.
- جهش به جستجو کمک می‌کند تا راه‌حل‌هایی که *تقاطع* به تنهایی به آنها برخورد نمی‌کند پیدا شوند.

الگوریتم‌های ژنتیک

تابع برازش (تابع هدف)

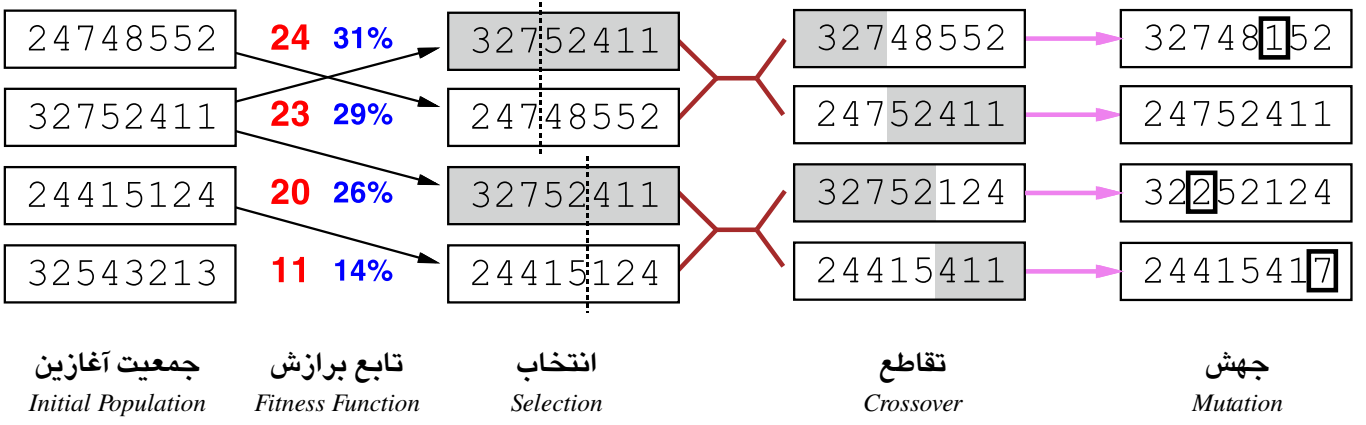
تابع برازش *Fitness Function*	تابعی که میزان خوب بودن یک فرد (کروموزوم) را نشان می‌دهد.
تابع هدف *Objective Function*	

هدف، ماکزیمم‌سازی تابع برازش است.

الگوریتم‌های ژنتیک

مثال

الگوریتم ژنتیک = جستجوی پرتوی محلی اتفاقی + تولید مابعدها از روی **جفت‌هایی** از حالت‌ها



نمونه‌ی الگوریتم ژنتیک، ارائه شده برای بازنمایی حالت‌های مسئله‌ی ۸-وزیر در قالب رشته‌ی عددی
هر عدد برای یک ستون صفحه‌ی شطرنج و عدد مربوطه شماره‌ی سطر است.

الگوریتم ژنتیک

شبه کد

function GENETIC-ALGORITHM(*population*, FITNESS-FN) **returns** an individual
  inputs: *population*, a set of individuals
    FITNESS-FN, a function that measures the fitness of an individual

  repeat
    new_population ← empty set
    for $i = 1$ **to** SIZE(*population*) **do**
      $x \leftarrow$ RANDOM-SELECTION(*population*, FITNESS-FN)
      $y \leftarrow$ RANDOM-SELECTION(*population*, FITNESS-FN)
      child ← REPRODUCE(x, y)
      if (small random probability) **then** *child* ← MUTATE(*child*)
      add *child* to *new_population*
    population ← *new_population*
  until some individual is fit enough, or enough time has elapsed
  return the best individual in *population*, according to FITNESS-FN

---

function REPRODUCE(x, y) **returns** an individual
  inputs: x, y, parent individuals

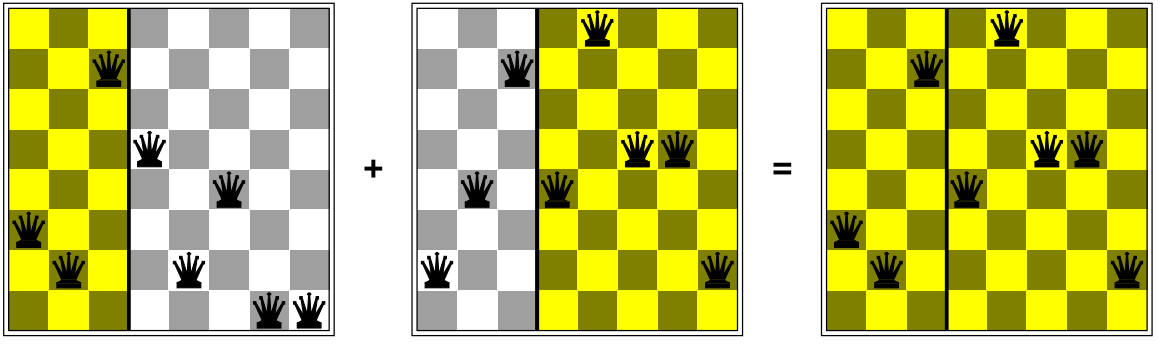
  $n \leftarrow$ LENGTH(x); $c \leftarrow$ random number from 1 to n
  return APPEND(SUBSTRING($x, 1, c$), SUBSTRING($y, c + 1, n$))

الگوریتم ژنتیک

بازنمایی حالت‌ها

حالت‌ها (کروموزوم‌ها) در الگوریتم ژنتیک به صورت **رشته‌ها** کدگذاری می‌شوند.

تقاطع (crossover) در صورتی کمک می‌کند که **زیررشته‌ها** اجزای معناداری باشند.



مثال: در مسئله‌ی ۸-وزیر، رشته‌ی عددی دارای زیررشته‌های معنادار است.

الگوریتم ژنتیک

مفهوم «نسل»

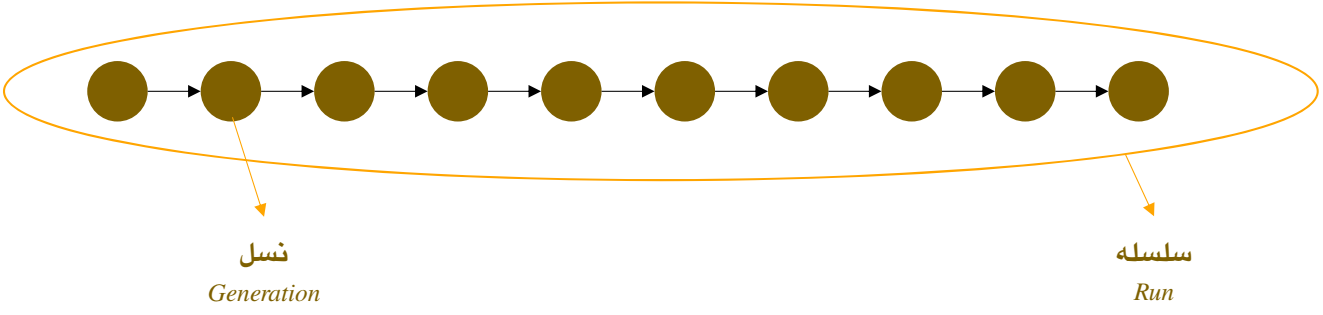
GENERATION

GA یک فرآیند تکراری را نمایش می‌دهد.

هر تکرار الگوریتم ژنتیک، یک **نسل** نامیده می‌شود.

(برای یک الگوریتم ژنتیک ساده، اندازه‌ی نسل می‌تواند بین ۵۰ تا بیش از ۵۰۰ گسترش یابد.)

کل مجموعه‌ی نسل‌ها یک **سلسله** نام دارد.



الگوریتم ژنتیک

شرط خاتمه‌ی الگوریتم و شروع مجدد

TERMINATION AND RESTARTING

GA از یک روش جستجوی اتفاقی استفاده می‌کند.

⇓

برازش یک جمعیت ممکن است پیش از ظاهر شدن کروموزوم برتر،
برای تعدادی از نسل‌ها پایدار (ثابت) بماند.

تجربه می‌گوید: معمولاً بهتر است GA پس از تعداد مشخصی نسل خاتمه پیدا کند
و بهترین کروموزوم‌ها در آن جمعیت بررسی شوند.
اگر راه‌حل راضی‌کننده‌ای پیدا نشد،
GA مجدداً شروع می‌شود.

الگوریتم ژنتیک

مراحل استفاده از الگوریتم ژنتیک برای حل مسئله

شکل‌های متنوعی از GA وجود دارد، اما اگر تابع برازش، کروموزوم‌ها و عملگرهای ژنتیکی به خوبی تعریف شده باشند، تغییر شکل الگوریتم معمولاً بهبودهای اندکی ایجاد می‌کند.

الگوریتم‌های ژنتیک

GENETIC ALGORITHMS (GAS)

مشخصه‌های الگوریتم‌های ژنتیک	
✓ روش جستجوی اتفاقی	✗ روش جستجوی قطعی
✓ کار بر روی یک جمعیت از راه‌حل‌ها	✗ کار بر روی یک راه‌حل
✓ روش جستجوی آگاهانه	✗ روش جستجوی ناآگاهانه
✓ تفکیک الگوریتم از بازنمایی	✗ یکپارچگی الگوریتم با بازنمایی

الگوریتم‌های ژنتیک

مزایا و معایب

مزایا و معایب الگوریتم‌های ژنتیک	
مزایا	معایب
سادگی بالا	سرعت پایین
عملکرد خوب روی انواع مختلفی از مسائل	محاسبات سنگین
عملکرد خوب روی فضاهای هیبرید (پیوسته/گسسته)	عدم وفق‌یابی مناسب با موقعیت‌های جدید
شک کمتر به گیر افتادن در بهینه‌های محلی	

الگوریتم ژنتیک

مورد مطالعاتی: یافتن مقدار بهینه‌ی یک تابع (۱ از ۹)

یک مثال ساده

که به ما کمک می‌کند بفهمیم GA چگونه کار می‌کند؟

می‌خواهیم مقدار ماکزیمم تابع زیر را بیابیم:

$$f(x) = 15x - x^2$$

$$x \in \{0, 1, 2, \ldots, 15\}$$

الگوریتم ژنتیک

مورد مطالعاتی: یافتن مقدار بهینه‌ی یک تابع (۲ از ۹)

$$\text{maximize } f(x), \quad f(x) = 15x - x^2 \quad , x \in \{0,1,2,\ldots,15\}$$

کروموزوم‌ها می‌توانند با چهار بیت (ژن) ساخته شوند:

Integer	*Binary code*	*Integer*	*Binary code*	*Integer*	*Binary code*
1	0 0 0 1	6	0 1 1 0	11	1 0 1 1
2	0 0 1 0	7	0 1 1 1	12	1 1 0 0
3	0 0 1 1	8	1 0 0 0	13	1 1 0 1
4	0 1 0 0	9	1 0 0 1	14	1 1 1 0
5	0 1 0 1	10	1 0 1 0	15	1 1 1 1

الگوریتم ژنتیک

مورد مطالعاتی: یافتن مقدار بهینه‌ی یک تابع (۳ از ۹)

اندازه‌ی جمعیت $N = 6$

احتمال تقاطع $p_c = 0.7$

احتمال جهش $p_m = 0.001$

تابع برازش $f(x) = 15x - x^2$

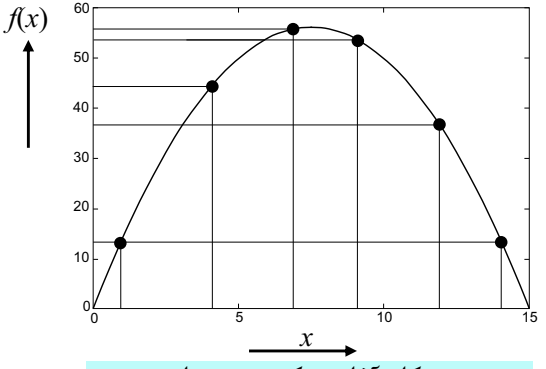
Chromosome label	*Chromosome string*	*Decoded integer*	*Chromosome fitness*	*Fitness ratio, %*
X1	1 1 0 0	12	36	16.5
X2	0 1 0 0	4	44	20.2
X3	0 0 0 1	1	14	6.4
X4	1 1 1 0	14	14	6.4
X5	0 1 1 1	7	56	25.7
X6	1 0 0 1	9	54	24.8

الگوریتم ژنتیک

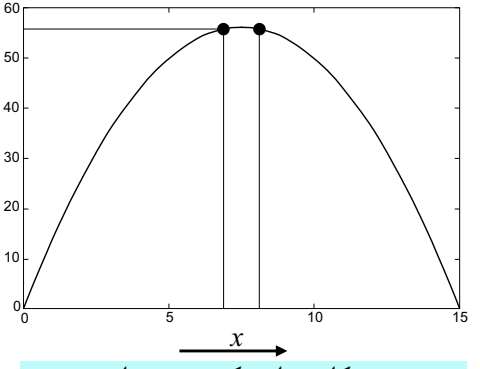
مورد مطالعاتی: یافتن مقدار بهینه‌ی یک تابع (۴ از ۹)

تابع برازش و مکان کروموزوم‌ها

Chromosome label	*Chromosome string*	*Decoded integer*	*Chromosome fitness*	*Fitness ratio, %*
X1	1 1 0 0	12	36	16.5
X2	0 1 0 0	4	44	20.2
X3	0 0 0 1	1	14	6.4
X4	1 1 1 0	14	14	6.4
X5	0 1 1 1	7	56	25.7
X6	1 0 0 1	9	54	24.8



مکان آغازین کروموزم‌ها



مکان نهایی کروموزم‌ها

الگوریتم ژنتیک

مورد مطالعاتی: یافتن مقدار بهینه‌ی یک تابع (۵ از ۹)

در انتخاب طبیعی، تنها برازنده‌ترین گونه‌ها می‌توانند
بقا داشته باشند، تولید مثل کنند و از این طریق ژن‌هایشان را به نسل بعدی منتقل کنند.

الگوریتم‌های ژنتیک نیز از روی‌کرد مشابهی استفاده می‌کنند،
اما برخلاف طبیعت اندازه‌ی جمعیت کروموزوم‌ها از یک نسل به نسل دیگر بدون تغییر باقی می‌ماند.

Chromosome label	*Chromosome string*	*Decoded integer*	*Chromosome fitness*	*Fitness ratio, %*
X1	1 1 0 0	12	36	16.5
X2	0 1 0 0	4	44	20.2
X3	0 0 0 1	1	14	6.4
X4	1 1 1 0	14	14	6.4
X5	0 1 1 1	7	56	25.7
X6	1 0 0 1	9	54	24.8

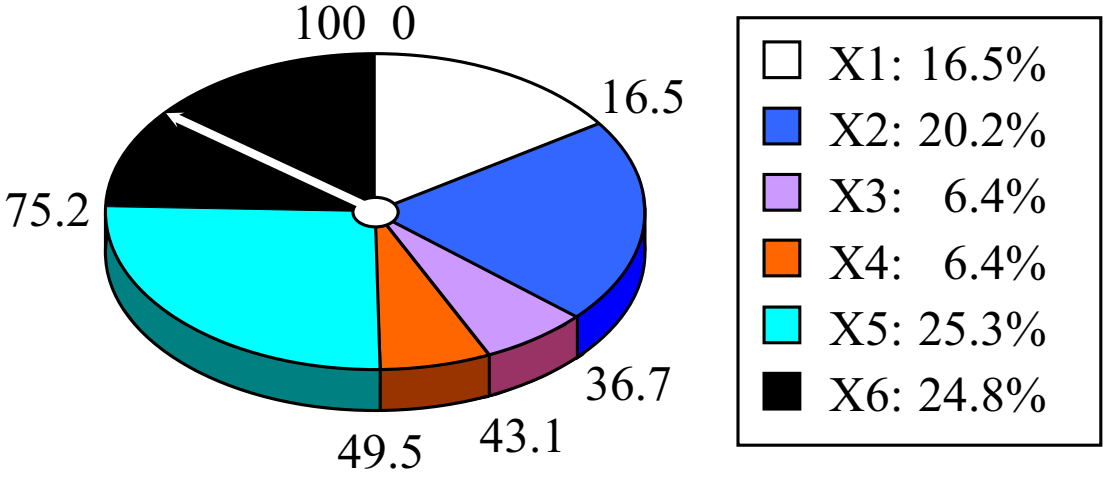
ستون آخر جدول، نسبت برازش هر کروموزوم را به برازش کل جمعیت نشان می‌دهد.
این نسبت شانس انتخاب شدن برای تزویج را نشان می‌دهد.
برازش متوسط کروموزوم‌ها از یک نسل به نسل بعدی بهتر می‌شود.

الگوریتم ژنتیک

مورد مطالعاتی: یافتن مقدار بهینه‌ی یک تابع (۶ از ۹)

انتخاب چرخ گردان
Roulette Wheel Selection

متداول‌ترین تکنیک انتخاب کروموزوم، انتخاب با چرخ گردان است.
چرخ به گردش در می‌آید، فلش روی هر ناحیه‌ای ایستاد، آن ناحیه انتخاب می‌شود.
احتمال انتخاب متناسب با مساحت ناحیه است.



الگوریتم ژنتیک

مورد مطالعاتی: یافتن مقدار بهینه‌ی یک تابع (۷ از ۹)

تقاطع تک‌نقطه‌ای
Single-Point Crossover

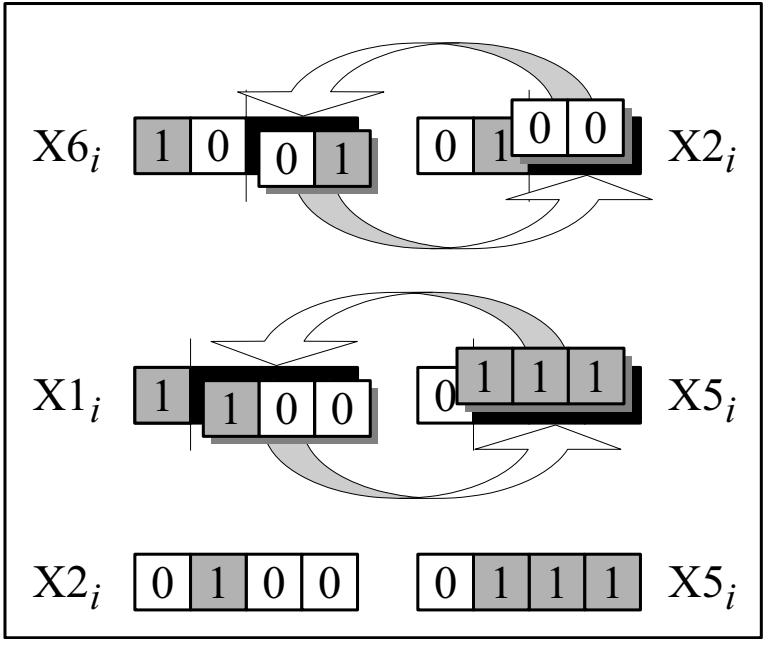
وقتی کروموزوم‌های والد انتخاب شدند، عملگر **تقاطع** اعمال می‌شود.

عملگر تقاطع، ابتدا یک نقطه‌ی تقاطع را انتخاب می‌کند که از آنجا دو کروموزوم والد شکسته می‌شوند و دو بخش این دو کروزموم از این نقطه مبادله می‌شود.

⇓

دو فرزند جدید ساخته می‌شود.

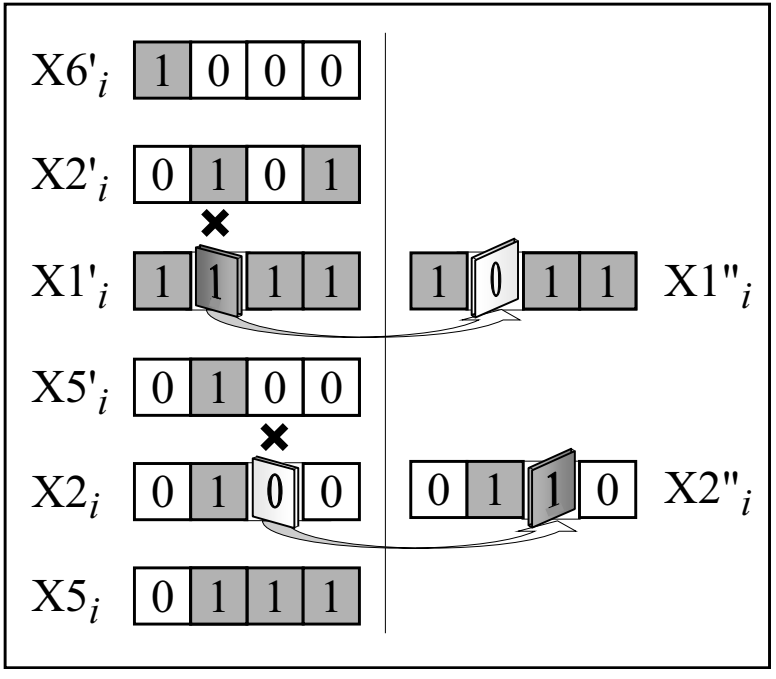
اگر یک زوج از کروموزم‌ها در تقاطع شرکت نکنند، آن‌گاه تولید مثل غیرجنسی اتفاق می‌افتد: فرزندان به‌صورت کپی‌های کامل هر والد ایجاد می‌شوند.



الگوریتم ژنتیک

مورد مطالعاتی: یافتن مقدار بهینه‌ی یک تابع (۸ از ۹)

جهش
Mutation



جهش، یک تغییر در ژن را بازنمایی می‌کند.

جهش یک عملگر پس‌زمینه‌ای است.
نقش جهش تضمین کردن این است که الگوریتم جستجو در بهینه‌ی محلی گیر نمی‌افتد.

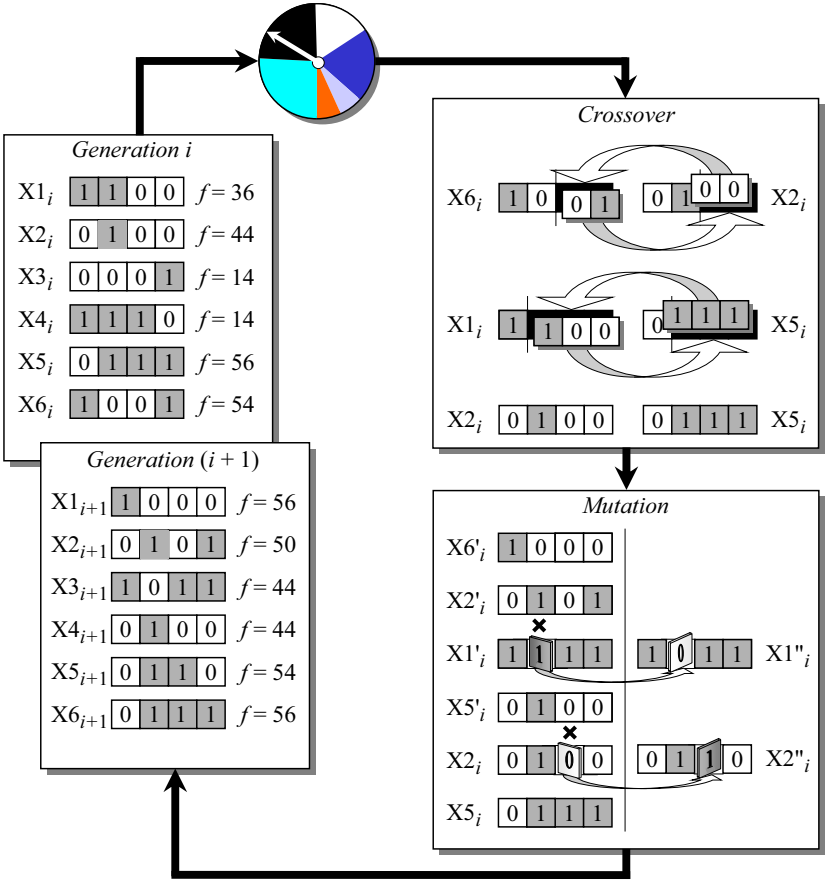
عملگر جهش، یک ژن انتخاب شده به صورت تصادفی از کروموزوم را وارون می‌کند.

احتمال جهش در طبیعت بسیار کوچک است و برای GA هم کوچک گرفته می‌شود (معمولاً در بازه‌ی 0.001 تا 0.1).

الگوریتم ژنتیک

مورد مطالعاتی: یافتن مقدار بهینه‌ی یک تابع (۹ از ۹)

چرخه‌ی الگوریتم ژنتیک
The Genetic Algorithm Cycle



الگوریتم ژنتیک

مورد مطالعاتی: یافتن مقدار ماکزیمم یک تابع دومتغیری (۱ از ۱۱)

می‌خواهیم ماکزیمم «قله»ی تابع دومتغیری زیر را بیابیم:

$$f(x,y) = (1-x)^2 e^{-x^2-(y+1)^2} - (x - x^3 - y^3)\, e^{-x^2-y^2}$$

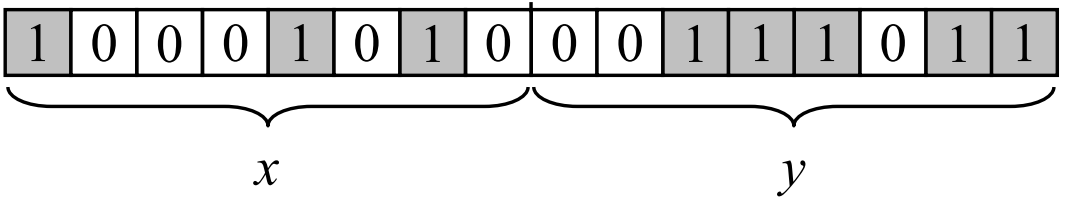
$$-3 \le x, y \le 3$$

الگوریتم ژنتیک

مورد مطالعاتی: یافتن مقدار ماکزیمم یک تابع دومتغیری (۲ از ۱۱)

$$f(x,y) = (1-x)^2 e^{-x^2-(y+1)^2} - (x - x^3 - y^3) e^{-x^2-y^2}$$
$$-3 \le x, y \le 3$$

در گام اول، متغیرهای مسئله را به‌صورت یک کروموزوم ۱۶ بیتی بازنمایی می‌کنیم:



پارامترهای x و y به صورت یک رشته‌ی دودویی الحاق شده بازنمایی می‌شوند.

اندازه‌ی جمعیت برای نمونه $N = 6$ انتخاب می‌شود،
و جمعیت آغازین به‌صورت تصادفی تولید می‌شود.

الگوریتم ژنتیک

مورد مطالعاتی: یافتن مقدار ماکزیمم یک تابع دومتغیری (۳ از ۱۱)

$$f(x,y) = (1-x)^2 e^{-x^2-(y+1)^2} - (x - x^3 - y^3) e^{-x^2-y^2}$$
$$-3 \le x, y \le 3$$

در گام بعدی، **برازش** هر کروموزوم محاسبه می‌شود:

1	0	0	0	1	0	1	0

and

0	0	1	1	1	0	1	1

هر نیمه‌ی کروموزوم یک عدد ۸ بیتی است که به‌عنوان یک عدد حقیقی بین 3 و 3- تعبیر می‌شود:

$$(10001010)_2 = 1\times2^7 + 0\times2^6 + 0\times2^5 + 0\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 = (138)_{10}$$

and

$$(00111011)_2 = 0\times2^7 + 0\times2^6 + 1\times2^5 + 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = (59)_{10}$$

$$\frac{3-(-3)}{2^8-1} = \frac{6}{256-1} = 0.0235294 \quad \Rightarrow$$

$$x = (138)_{10} \times 0.0235294 - 3 = 0.2470588$$

and

$$y = (59)_{10} \times 0.0235294 - 3 = -1.6117647$$

الگوریتم ژنتیک

مورد مطالعاتی: یافتن مقدار ماکزیمم یک تابع دومتغیری (۴ از ۱۱)

$$f(x,y) = (1-x)^2 e^{-x^2-(y+1)^2} - (x - x^3 - y^3) e^{-x^2-y^2}$$
$$-3 \le x, y \le 3$$

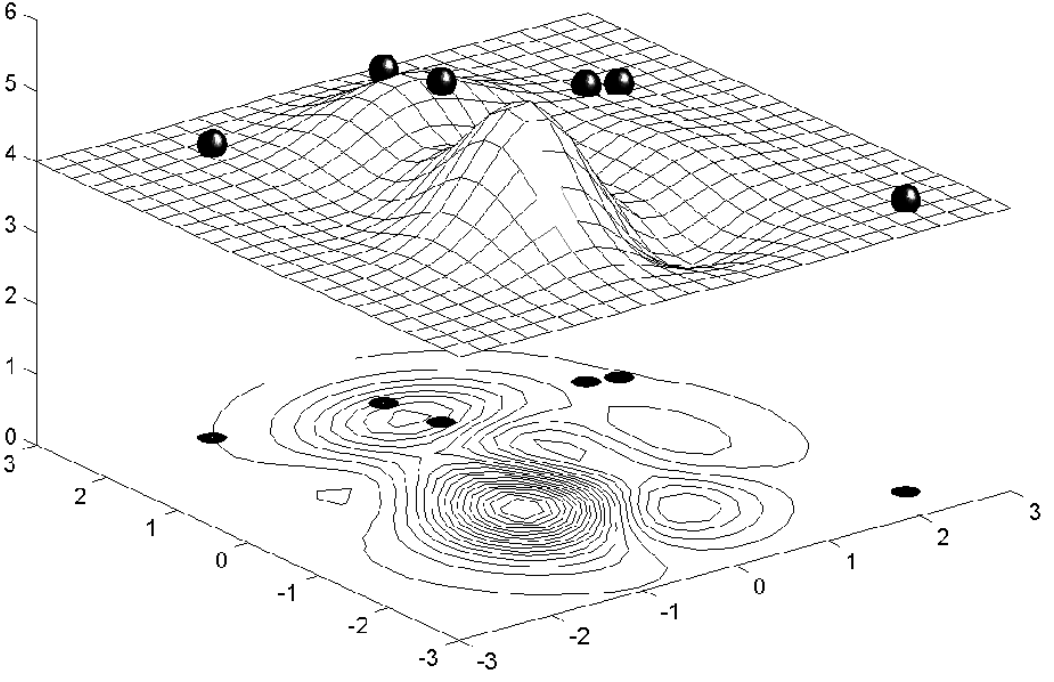
اندازه‌ی جمعیت	$N = 6$
احتمال تقاطع	$p_c = 0.7$
احتمال جهش	$p_m = 0.001$
تابع برازش	$f(x,y)$

تعداد نسل‌ها را 100 در نظر می‌گیریم.
پس GA ۱۰۰ نسل از شش کروموزوم را پیش از توقف ایجاد می‌کند.

الگوریتم ژنتیک

مورد مطالعاتی: یافتن مقدار ماکزیمم یک تابع دومتغیری (۵ از ۱۱)

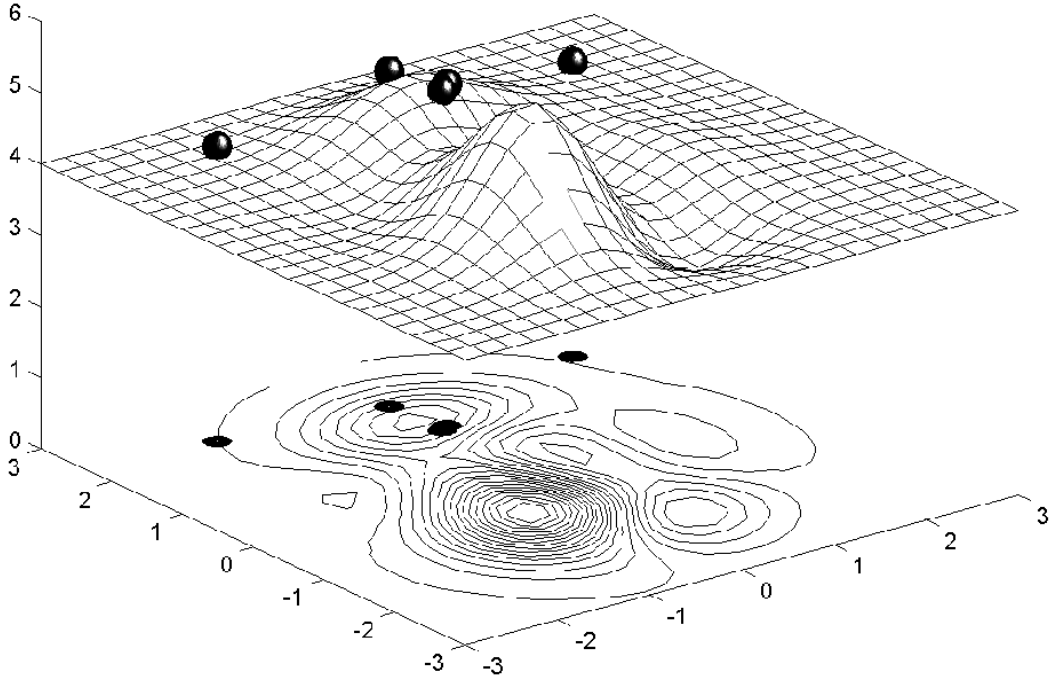
مکان کروموزوم‌ها بر روی رویه‌ی تابع «قله»: **جمعیت آغازین**



الگوریتم ژنتیک

مورد مطالعاتی: یافتن مقدار ماکزیمم یک تابع دومتغیری (۶ از ۱۱)

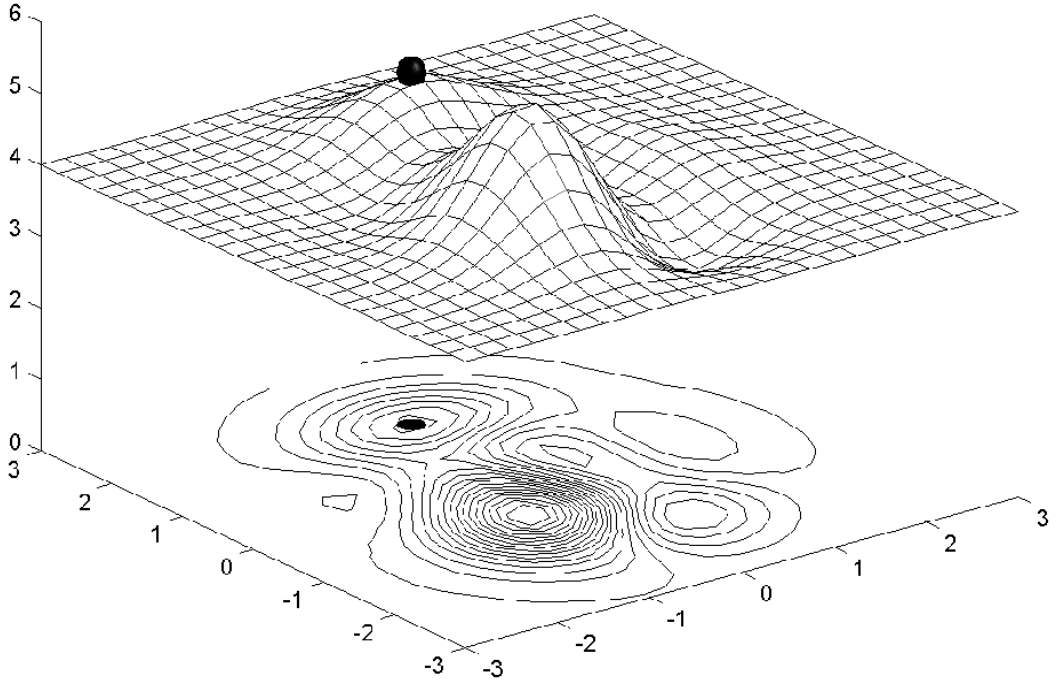
مکان کروموزوم‌ها بر روی رویه‌ی تابع «قله»: **نسل اول**



الگوریتم ژنتیک

مورد مطالعاتی: یافتن مقدار ماکزیمم یک تابع دومتغیری (۷ از ۱۱)

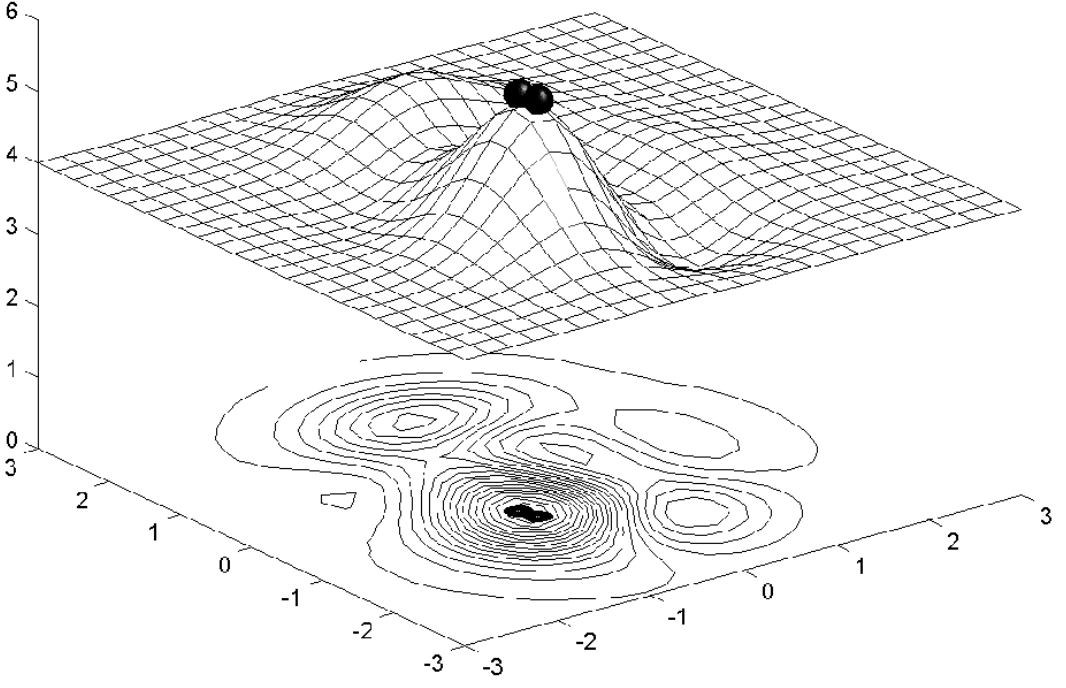
مکان کروموزوم‌ها بر روی رویه‌ی تابع «قله»: **ماکزیمم محلی**



الگوریتم ژنتیک

مورد مطالعاتی: یافتن مقدار ماکزیمم یک تابع دومتغیری (۸ از ۱۱)

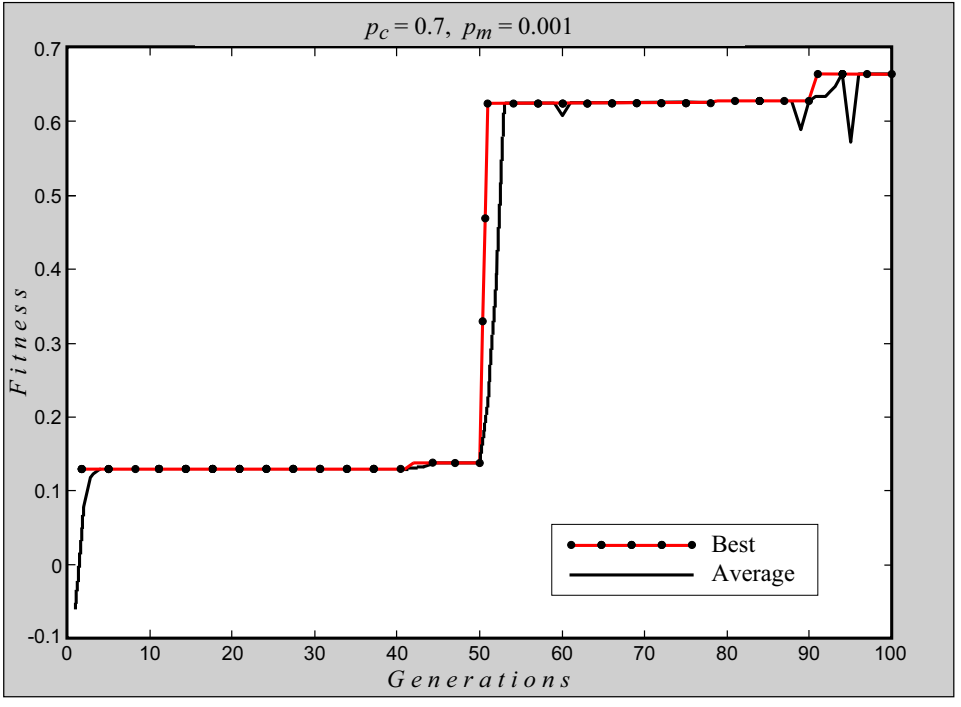
مکان کروموزوم‌ها بر روی رویه‌ی تابع «قله»: **ماکزیمم سراسری**



الگوریتم ژنتیک

مورد مطالعاتی: یافتن مقدار ماکزیمم یک تابع دومتغیری (۹ از ۱۱)

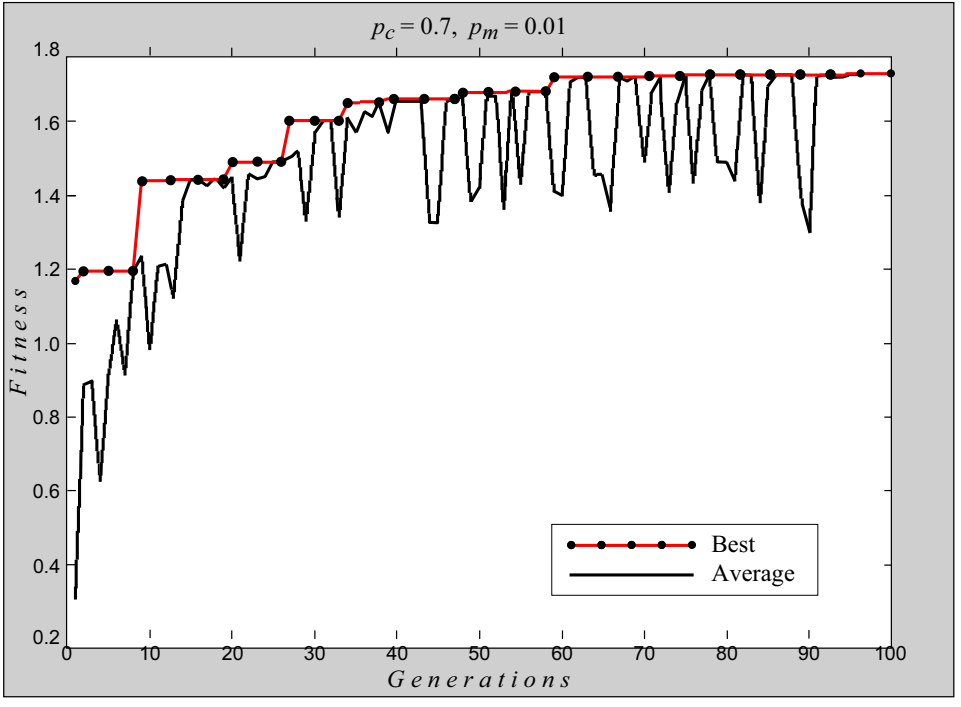
نمودارهای کارآیی برای ۱۰۰ نسل از ۶ کروموزوم: **ماکزیمم محلی**



الگوریتم ژنتیک

مورد مطالعاتی: یافتن مقدار ماکزیمم یک تابع دومتغیری (۱۰ از ۱۱)

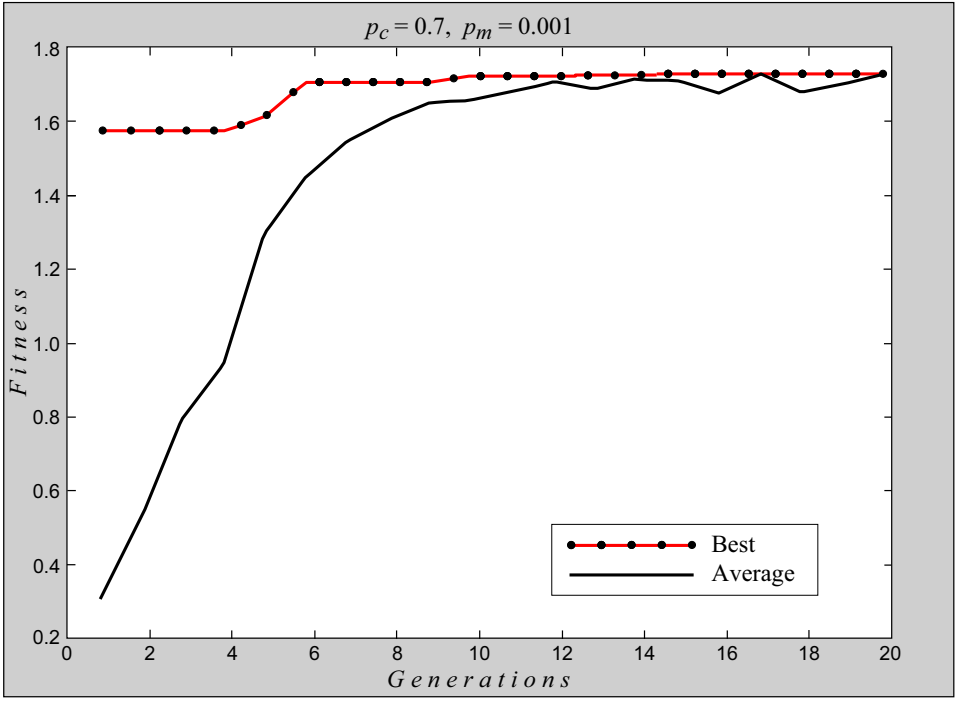
نمودارهای کارآیی برای ۱۰۰ نسل از ۶ کروموزوم: **ماکزیمم سراسری**



الگوریتم ژنتیک

مورد مطالعاتی: یافتن مقدار ماکزیمم یک تابع دومتغیری (۱۱ از ۱۱)

نمودارهای کارآیی برای ۲۰ نسل از ۶۰ کروموزوم



الگوریتم ژنتیک

مورد مطالعاتی: زمان‌بندی تعمیر (۱ از ۱۰)

CASE STUDY: MAINTENANCE SCHEDULING

مسائل زمان‌بندی تعمیر:
معمولاً با استفاده از ترکیبی از تکنیک‌های جستجو و هیوریستیک‌ها حل می‌شود.

- این مسائل پیچیده هستند و حل آنها دشوار است.
- این مسائل NP-complete هستند و نمی‌توانند با روش‌های جستجوی ترکیبیاتی حل شوند.
- زمان‌بندی حاوی رقابت برای منابع محدود است و با زیاد شدن قیدهای صوری بد پیچیده‌تر می‌شوند.

الگوریتم ژنتیک

مورد مطالعاتی: زمان‌بندی تعمیر (۲ از ۱۰)

CASE STUDY: MAINTENANCE SCHEDULING

زمان‌بندی ۷ واحد در ۴ بازه‌ی مساوی:

قیدهای مسئله:

- حداکثر بار مورد انتظار در طول چهار بازه، به‌ترتیب عبارت است از:

80, 90, 65, 70 MW

- تعمیر هر واحد در ابتدای یک بازه شروع می‌شود و در انتهای همان بازه یا بازه‌ی مجاور تمام می‌شود.
تعمیر نمی‌تواند لغو شود یا زودتر از زمان‌بندی انجام شده تمام شود.
- رزرو خالص سیستم قدرت باید در همه‌ی بازه‌ها بزرگ‌تر یا مساوی با صفر باشد.

ضابطه‌ی بهینه: ماکزیمم رزرو خالص در هر دوره‌ی تعمیر

الگوریتم ژنتیک

مورد مطالعاتی: زمان‌بندی تعمیر (۳ از ۱۰)

CASE STUDY: MAINTENANCE SCHEDULING

داده‌های واحد و نیازمندی‌های تعمیر:

Unit number	*Unit capacity, MW*	*Number of intervals required for unit maintenance*
1	20	2
2	15	2
3	35	1
4	40	1
5	15	1
6	15	1
7	10	1

الگوریتم ژنتیک

مورد مطالعاتی: زمان‌بندی تعمیر (۴ از ۱۰)

CASE STUDY: MAINTENANCE SCHEDULING

حوض ژن‌های هر واحد:

Unit 1:	1 1 0 0	0 1 1 0	0 0 1 1	
Unit 2:	1 1 0 0	0 1 1 0	0 0 1 1	
Unit 3:	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Unit 4:	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Unit 5:	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Unit 6:	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Unit 7:	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1

کروموزوم برای مسئله‌ی زمان‌بندی

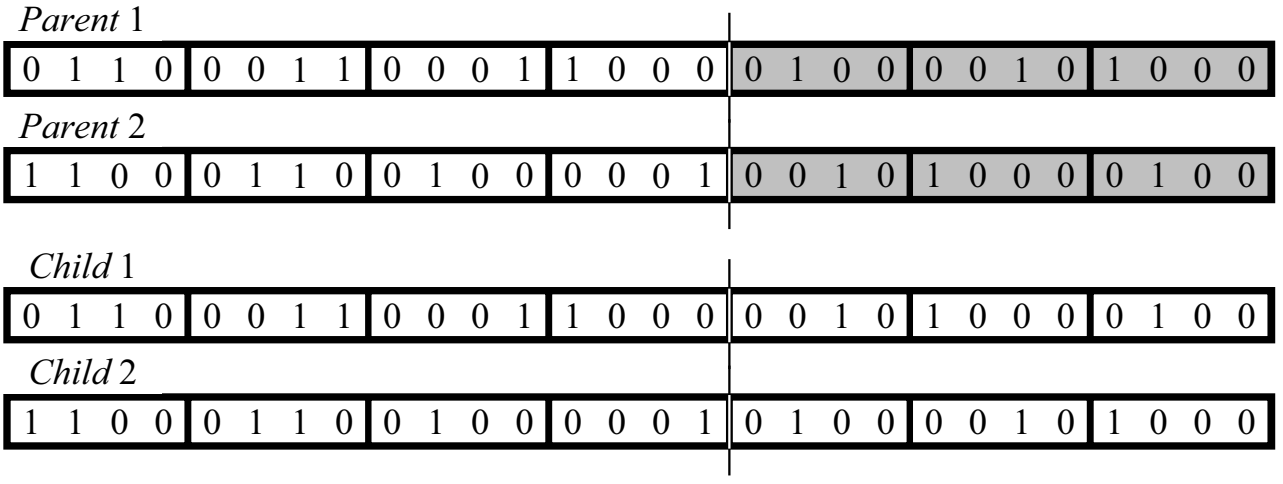
Unit 1	Unit 2	Unit 3	Unit 4	Unit 5	Unit 6	Unit 7
0 1 1 0	0 0 1 1	0 0 0 1	1 0 0 0	0 1 0 0	0 0 1 0	1 0 0 0

الگوریتم ژنتیک

مورد مطالعاتی: زمان‌بندی تعمیر (۵ از ۱۰)

CASE STUDY: MAINTENANCE SCHEDULING

عملگر تقاطع:

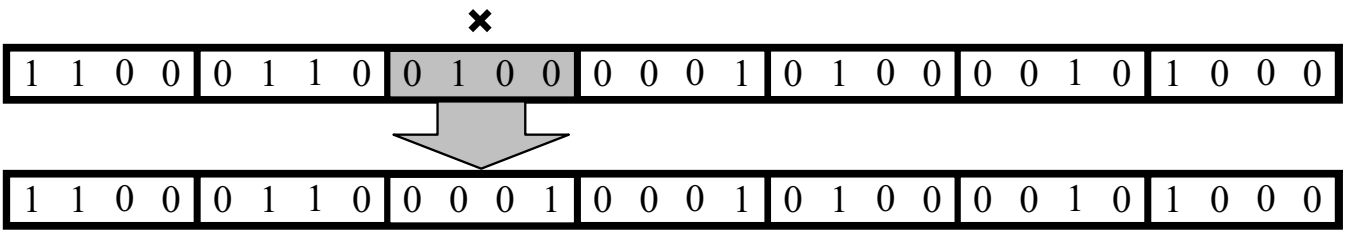


الگوریتم ژنتیک

مورد مطالعاتی: زمان‌بندی تعمیر (۶ از ۱۰)

CASE STUDY: MAINTENANCE SCHEDULING

عملگر جهش:

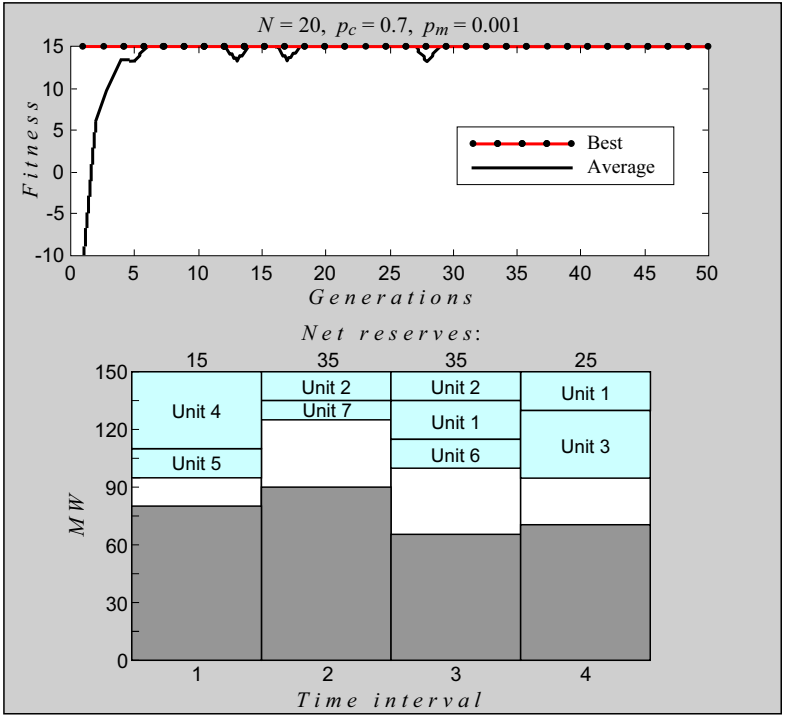


الگوریتم ژنتیک

مورد مطالعاتی: زمان‌بندی تعمیر (۷ از ۱۰)

CASE STUDY: MAINTENANCE SCHEDULING

نمودارهای کارآیی و بهترین زمان‌بندی تعمیر
ایجاد شده در یک جمعیت از ۲۰ کروموزوم (۵۰ نسل)

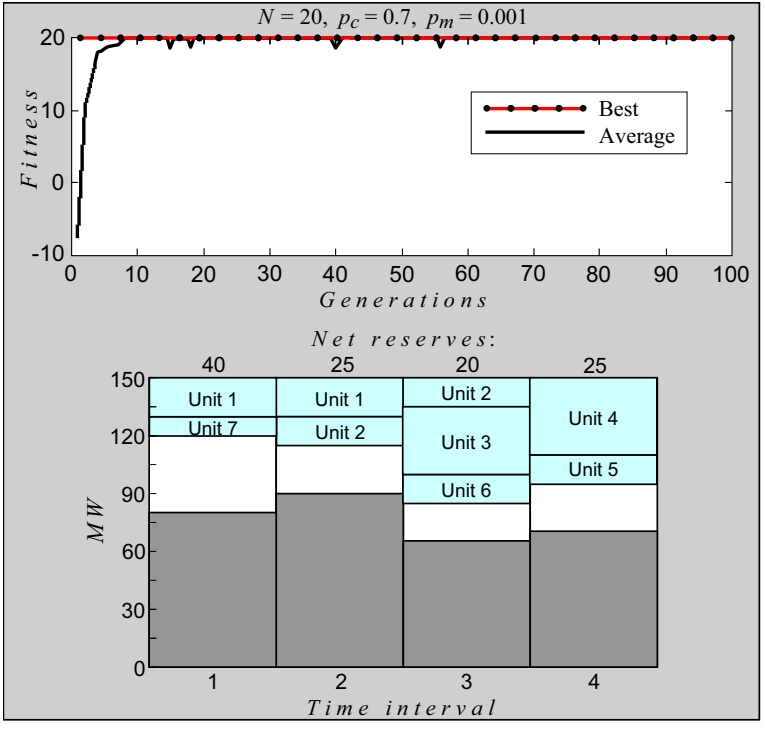


الگوریتم ژنتیک

مورد مطالعاتی: زمان‌بندی تعمیر (۸ از ۱۰)

CASE STUDY: MAINTENANCE SCHEDULING

نمودارهای کارآیی و بهترین زمان‌بندی تعمیر
ایجاد شده در یک جمعیت از ۲۰ کروموزوم (۱۰۰ نسل)

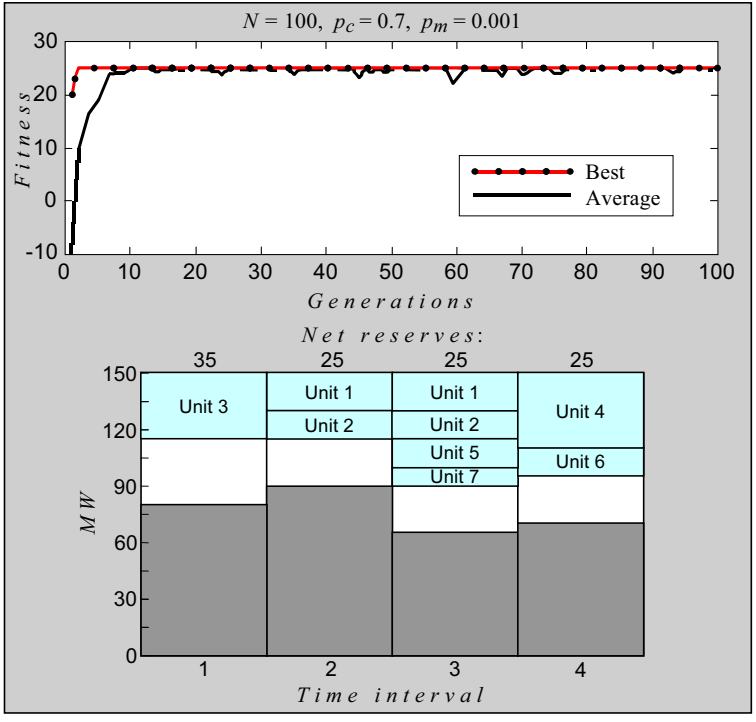


الگوریتم ژنتیک

مورد مطالعاتی: زمان‌بندی تعمیر (۹ از ۱۰)

CASE STUDY: MAINTENANCE SCHEDULING

نمودارهای کارآیی و بهترین زمان‌بندی تعمیر
ایجاد شده در یک جمعیت از ۱۰۰ کروموزوم (نرخ جهش 0.001)

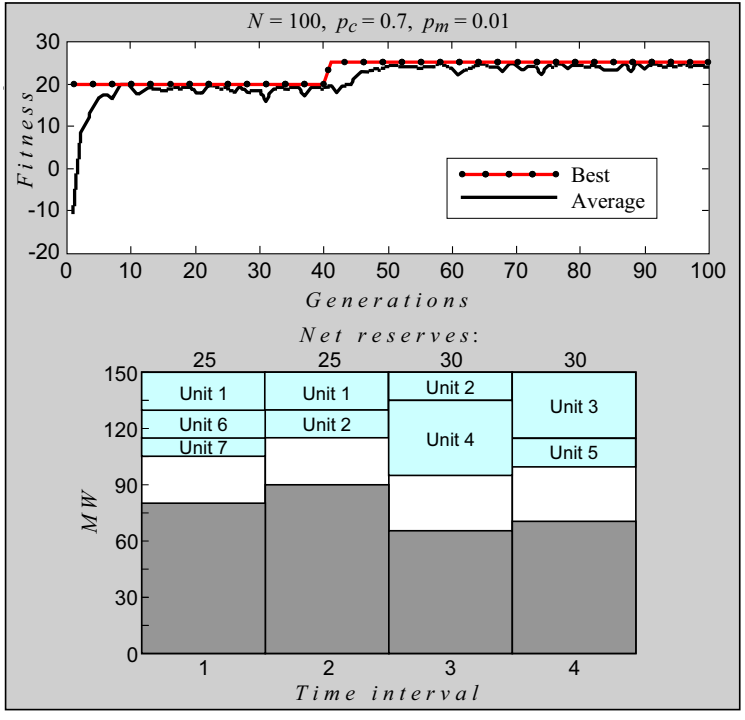


الگوریتم ژنتیک

مورد مطالعاتی: زمان‌بندی تعمیر (۱۰ از ۱۰)

CASE STUDY: MAINTENANCE SCHEDULING

نمودارهای کارآیی و بهترین زمان‌بندی تعمیر
ایجاد شده در یک جمعیت از ۱۰۰ کروموزوم (نرخ جهش 0.01)



هوش مصنوعی

هوش تطوری

۳

استراتژی‌های تطور

استراتژی‌های تطور

EVOLUTION STRATEGIES

استراتژی‌های تطور

رویکرد دیگری برای شبیه‌سازی انتخاب طبیعی

Rechenberg and Schwefel, 1960s, University of Berlin

هدف: حل مسائل بهینه‌سازی تکنیکی

برخلاف الگوریتم ژنتیک، استراتژی‌های تطور فقط از عملگر جهش استفاده می‌کنند.

استراتژی‌های تطور

فرم‌ها

EVOLUTION STRATEGIES

$\left(\mu \,/\, \rho, \lambda\right)$-ES

از μ والد λ فرزند تولید می‌شود.
در مرحله‌ی تولید فرزند از هر ρ والد، یک فرزند تولید می‌شود.

$\left(\mu + \lambda\right)$-ES

از مجموع جمعیت والدها و فرزندان $(\mu + \lambda)$،
تعداد μ مورد با توجه به یک استراتژی انتخاب، گزینش می‌شوند.

$\left(\mu \,,\, \lambda\right)$-ES

از تعداد λ فرزند، تعداد μ مورد برتر گزینش و به نسل بعد می‌روند.

$$1 \leq \mu \leq \lambda$$

استراتژی‌های تطور

ساده‌ترین فرم

EVOLUTION STRATEGIES

$$\mu = 1, \quad \lambda = 1$$

یک والد یک فرزند در نسل تولید می‌کند: (با اعمال جهش با توزیع نرمال)

$(1+1)$-ES

از مجموع جمعیت والدها و فرزندان $(1+1)$،
تعداد 1 مورد با توجه به یک استراتژی انتخاب، گزینش می‌شوند.

استراتژی‌های تطور

گام‌های استراتژی تطور (1+1)

۱) * انتخاب اندازه‌ی جمعیت N (تعداد پارامترها) و تعیین بازه‌ی ممکن برای هر پارامتر
* تعریف یک انحراف استاندارد برای هر پارامتر (σ)
* تعریف تابعی که باید بهینه شود
$(x_{1\min}, x_{1\max}), (x_{2\min}, x_{2\max}), \ldots, (x_{N\min}, x_{N\max})$

۲) انتخاب تصادفی مقدار اولیه برای هر پارامتر از بازه‌ی ممکن آن
⇐ تشکیل اولیه‌ی پارامترهای والد
$x_1, x_2, \ldots, x_N$

۳) محاسبه‌ی تابع راه‌حل متناظر با پارامترهای والد
$X = f(x_1, x_2, \ldots, x_N)$

۴) ایجاد یک پارامتر جدید (فرزند)
$x_i' = x_i + a(0,\sigma) \quad i = 1,2,\ldots,N$
با اضافه کردن یک متغیر تصادفی a با توزیع نرمال با میانگین صفر و انحراف معیار مشخص σ
(جهش با توزیع نرمال برای انعکاس فرآیند طبیعی تطور: تغییرات کوچک بیشتر از تغییرات بزرگ رخ می‌دهند.)

۵) محاسبه‌ی تابع راه‌حل متناظر با پارامترهای فرزند
$X' = f(x_1', x_2', \ldots, x_N')$

۶) مقایسه‌ی راه‌حل فرزند با والد: اگر فرزند بهتر بود، با والد جایگزین می‌شود وگرنه والد را نگه می‌داریم.

۷) تکرار از (۴) تا «رسیدن به راه‌حل راضی‌کننده» یا «رسیدن به تعداد نسل از قبل مشخص شده»

استراتژی‌های تطور

ویژگی‌ها

EVOLUTION STRATEGIES

یک استراتژی تطور، طبیعت یک کروموزوم را منعکس می‌کند.

* یک ژن تنها ممکن است به‌طور همزمان بر خصوصیات متعددی از موجود زنده اثر بگذارد
* یک خصوصیت واحد از یک فرد ممکن است توسط تعاملات همزمان چندین ژن تعیین شود.
* انتخاب طبیعی، بر روی مجموعه‌ای از ژن‌ها عمل می‌کند، نه بر روی یک ژن واحد به‌تنهایی.

هوش مصنوعی

هوش تطوری

۴

برنامه‌نویسی ژنتیک

برنامه‌نویسی ژنتیک

GENETIC PROGRAMMING

برنامه‌نویسی ژنتیک

تطور برنامه‌های کامپیوتری از طریق روش‌های انتخاب طبیعی

John Koza, 1990s

یکی از مسائل محوری در علم کامپیوتر:
چگونه می‌توان کاری کرد که کامپیوترها مسائل را حل کنند
بدون اینکه صراحتاً برای انجام آن کار برنامه‌نویسی شوند؟
برنامه‌نویسی ژنتیک، پاسخی است به این مسئله است.

در واقع، برنامه‌نویسی ژنتیک، گسترشی از الگوریتم ژنتیک مرسوم است،
اما هدف آن، تنها تطور یک بازنمایی رشته–بیتی از یک مسئله نیست،
بلکه کد کامپیوتری است که مسئله را حل می‌کند.

برنامه‌نویسی ژنتیک

GENETIC PROGRAMMING

برنامه‌نویسی ژنتیک

فضای برنامه‌های کامپیوتری ممکن را جستجو می‌کند
برای یافتن یک برنامه که برای حل مسئله‌ی در دست بررسی تناسب بالایی داشته باشد.

هر برنامه‌ی کامپیوتری، دنباله‌ای از عملیات (تابع‌ها) است
که بر روی مقادیر (آرگومان‌ها) اعمال می‌شود،

اما زبان‌های برنامه‌نویسی مختلف ممکن است انواع مختلفی از دستورها و عملیات را داشته باشند
و دارای محدودیت‌های نحوی متفاوت باشند.

چون برنامه‌نویسی ژنتیک برنامه‌ها را با اعمال عملگرهای ژنتیکی دستکاری می‌کند،
یک زبان برنامه‌نویسی باید به برنامه‌ی کامپیوتری اجازه دهد تا
مانند داده‌ها دستکاری شود
و داده‌های ایجادشده‌ی جدید مانند یک برنامه اجرا شود.
به این دلیل، زبان **LISP** به عنوان **زبان اصلی برای برنامه‌نویسی ژنتیک** انتخاب می‌شود.

برنامه‌نویسی ژنتیک

ساختار لیسپ

LISP STRUCTURE

لیسپ، یک ساختار بسیار نماد–محور دارد:
ساختمان داده‌های پایه: اتم‌ها و لیست‌ها

اتم *Atom*	کوچک‌ترین عنصر تقسیم‌ناپذیر در نحو لیسپ: مثل: عدد *21*، نماد *X*، رشته‌ی *"This is a string"*
لیست *List*	شیئ متشکل از اتم‌ها و/یا دیگر لیست‌ها: در قالب: مجموعه‌ی مرتبی از عناصر درون یک جفت پرانتز

برنامه‌نویسی ژنتیک

ساختار لیسپ: مثال

LISP STRUCTURE

$$(-(*AB)C)$$

معادل با:

$$A * B - C$$

(یعنی اول لیست داخلی‌تر ارزیابی می‌شود و سپس لیست کلی ارزیابی می‌شود.)

برنامه‌نویسی ژنتیک

ساختار لیسپ: بازنمایی گرافیکی

GRAPHICAL REPRESENTATION OF LISP S-EXPRESSIONS

در لیسپ به اتم‌ها و لیست‌ها، **عبارت‌های نمادین** گفته می‌شود:

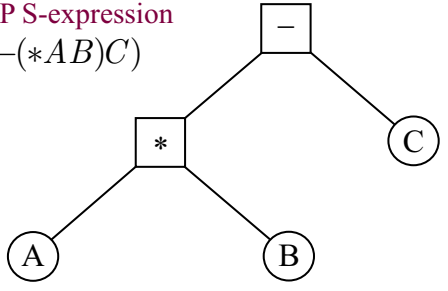
Symbolic Expressions – S-expression

در لیسپ همه‌ی داده‌ها و برنامه‌ها، **عبارت‌های نمادین** هستند.
این به لیسپ این قابلیت را می‌دهد که روی برنامه‌ها مشابه با داده‌ها کار کند:

برنامه‌های لیسپ می‌توانند خودشان را تغییر دهند،
یا حتی برنامه‌های لیسپ دیگر را بنویسند.

(این خاصیت قابل توجه لیسپ، آن را برای برنامه‌نویسی ژنتیک بسیار جذاب می‌کند.)

LISP S-expression
$(-(*AB)C)$

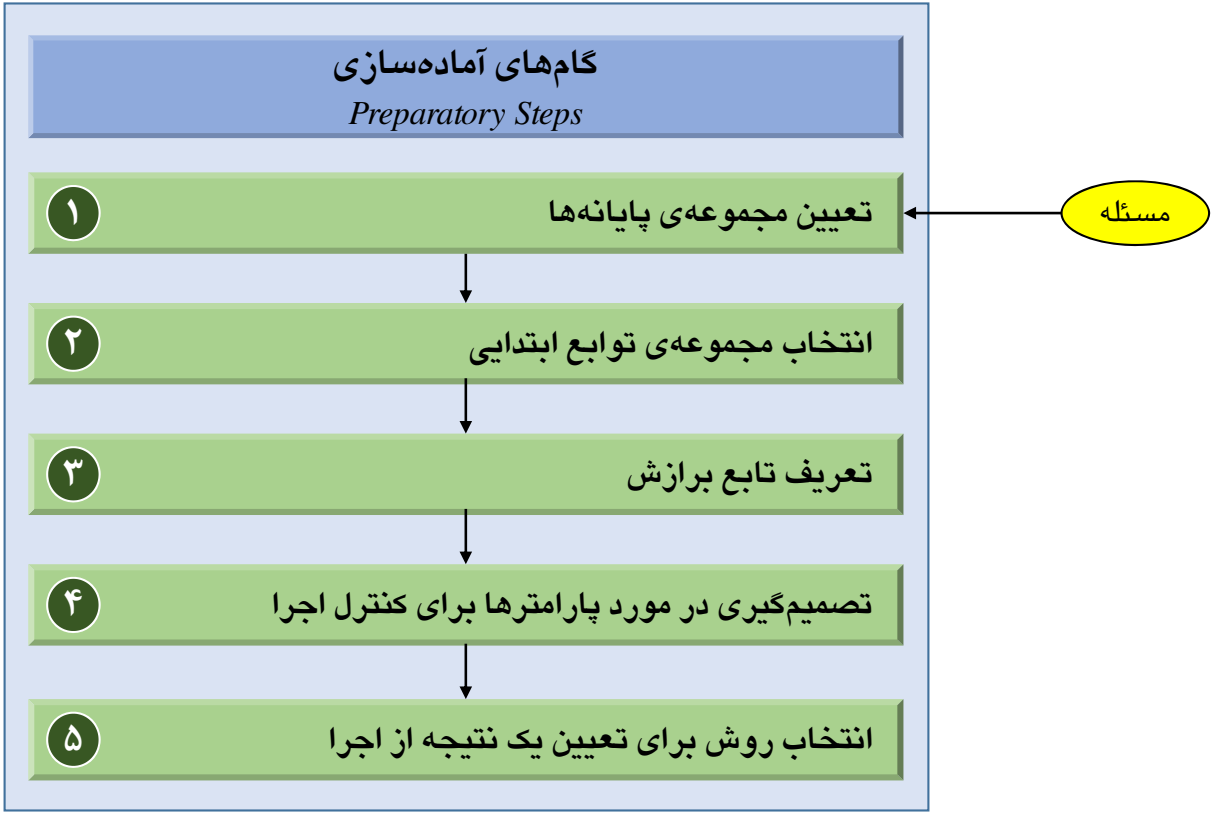


هر عبارت نمادین لیسپ را می‌توان
به‌صورت یک درخت ریشه‌دار مرتب برچسب‌دار نشان داد:

برنامه‌نویسی ژنتیک

اعمال برنامه‌نویسی ژنتیک بر روی یک مسئله

APPLYING GENETIC PROGRAMMING TO A PROBLEM



برنامه‌نویسی ژنتیک

مثال: برنامه‌ای برای محاسبه‌ی قاعده‌ی فیثاغورس (۱ از ۱۲)

هدف: استفاده از برنامه‌نویسی ژنتیک برای
کشف برنامه‌ای است که با قاعده‌ی فیثاغورث تطبیق دارد:

$$c = \sqrt{a^2 + b^2}$$

فرض می‌کنیم که فرمول را نمی‌دانیم ولی تعدادی نمونه‌ی عددی از مثلث‌های قائم‌الزاویه تصادفی داریم:

Side a	Side b	Hypotenuse c	Side a	Side b	Hypotenuse c
3	5	5.830952	12	10	15.620499
8	14	16.124515	21	6	21.840330
18	2	18.110770	7	4	8.062258
32	11	33.837849	16	24	28.844410
4	3	5.000000	2	9	9.219545

برای اندازه‌گیری کارآیی برنامه‌ی کامپیوتری کشف نشده،
از تعدادی مورد برازش (fitness cases) [نمونه‌های عددی] استفاده می‌کنیم.

برنامه‌نویسی ژنتیک

مثال: برنامه‌ای برای محاسبه‌ی قاعده‌ی فیثاغورس (۲ از ۱۲)

گام‌های آماده‌سازی
Preparatory Steps

۱ تعیین مجموعه‌ی پایانه‌ها

پایانه‌ها متناظر با **ورودی‌های** برنامه‌ی کامپیوتری است که باید کشف شود:

برنامه‌ی ما دو ورودی a و b می‌گیرد.

$$c = \sqrt{a^2 + b^2}$$

برنامه‌نویسی ژنتیک

مثال: برنامه‌ای برای محاسبه‌ی قاعده‌ی فیثاغورس (۳ از ۱۲)

گام‌های آماده‌سازی *Preparatory Steps*	انتخاب مجموعه‌ی توابع ابتدایی ۲

توابع می‌توانند با عملیات استاندارد ریاضی، عملیات استاندارد برنامه‌نویسی، توابع استاندارد ریاضی، توابع منطقی، یا توابع خاص دامنه بازنمایی شوند.

برنامه‌ی ما از چهار عمل حسابی اصلی و یک تابع ریاضی جذر استفاده می‌کند.

برنامه‌نویسی ژنتیک

مثال: برنامه‌ای برای محاسبه‌ی قاعده‌ی فیثاغورس (۴ از ۱۲)

گام‌های آماده‌سازی
Preparatory Steps

۳ تعریف تابع برازش

تابع برازش، میزان خوبی یک برنامه‌ی کامپیوتری که می‌تواند مسئله را حل کند، ارزیابی می‌کند.

برای مسئله‌ی ما، تابع برازش برنامه‌ی کامپیوتری می‌تواند با
خطای بین نتیجه‌ی واقعی تولید شده توسط برنامه و نتیجه‌ی صحیح داده‌شده توسط مورد برازش
اندازه‌گیری شود.
معمولاً این خطا بر روی فقط یک مورد برازش اندازه‌گیری نمی‌شود،
بلکه به‌جای آن مجموع خطاهای مطلق بر روی تعدادی مورد برازش محاسبه می‌شود.
هر چه این خطا به صفر نزدیک‌تر باشد، برنامه‌ی کامپیوتری بهتر است.

برنامه‌نویسی ژنتیک

مثال: برنامه‌ای برای محاسبه‌ی قاعده‌ی فیثاغورس (۵ از ۱۲)

گام‌های آماده‌سازی
Preparatory Steps

۴ تصمیم‌گیری در مورد پارامترها برای کنترل اجرا

برای کنترل یک اجرا، برنامه‌نویسی ژنتیک از همان پارامترهای اصلی
که در الگوریتم ژنتیک استفاده می‌شود، استفاده می‌کند.

شامل: اندازه‌ی جمعیت و تعداد نسل‌ها

برنامه‌نویسی ژنتیک

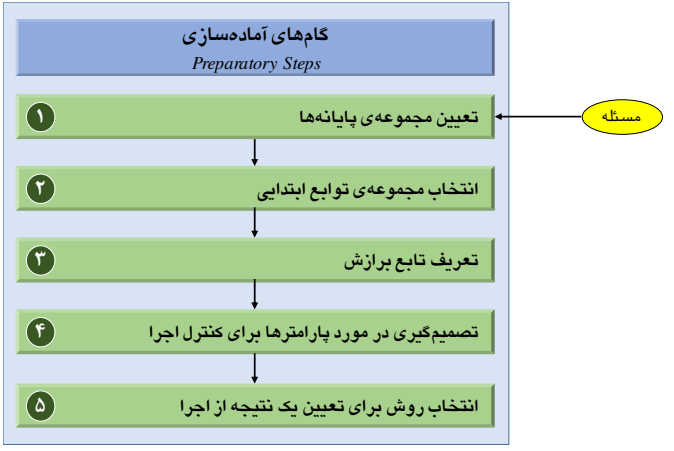
مثال: برنامه‌ای برای محاسبه‌ی قاعده‌ی فیثاغورس (۶ از ۱۲)

گام‌های آماده‌سازی *Preparatory Steps*	۵ انتخاب روش برای تعیین یک نتیجه از اجرا

معمولاً در برنامه‌نویسی ژنتیک، **بهترین برنامه‌ی تولید شده تا کنون** به عنوان نتیجه‌ی یک اجرا تعیین می‌شود.

برنامه‌نویسی ژنتیک

مثال: برنامه‌ای برای محاسبه‌ی قاعده‌ی فیثاغورس (۷ از ۱۲)



وقتی پنج گام آماده‌سازی کامل شد، اجرا می‌تواند انجام شود.

اجرای برنامه‌نویسی ژنتیک، با یک نسل تصادفی از یک جمعیت آغازین از برنامه‌های کامپیوتری شروع می‌شود.

هر برنامه از توابع چهار عمل اصلی حسابی و جذر و پایانه‌های a و b تشکیل می‌شود.

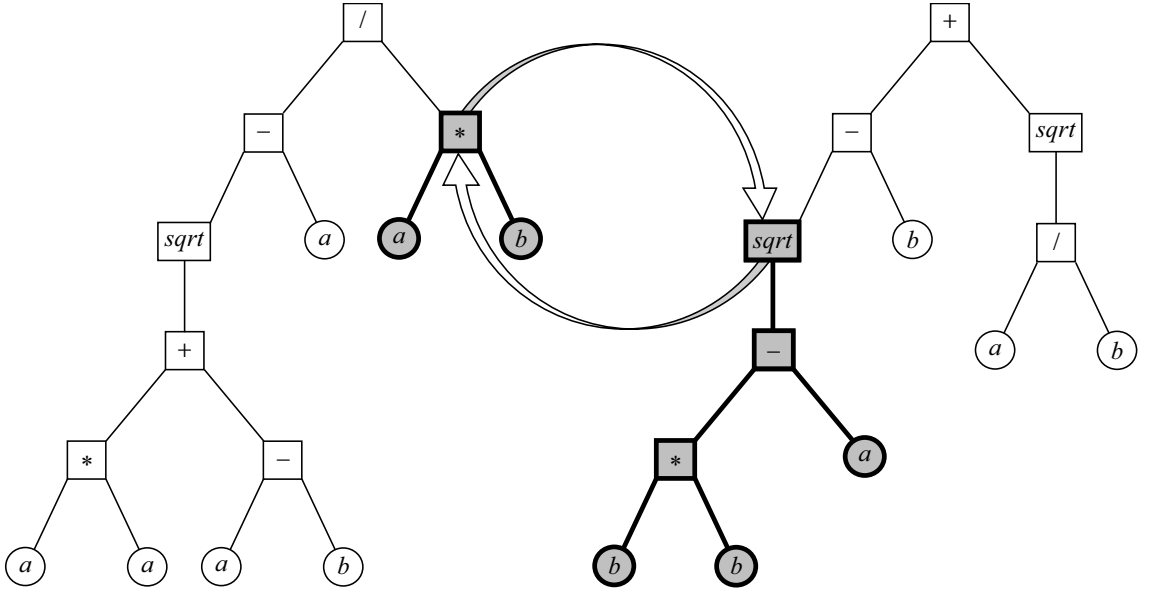
در جمعیت آغازین، معمولاً همه‌ی برنامه‌های کامپیوتری برازش ضعیفی دارند،
اما برخی افراد نسبت به دیگران بهتر هستند.
همان‌طور که کروموزوم برازنده‌تر شانس بیشتری برای تولید مثال دارد،
یک برنامه‌ی کامپیوتری برازنده‌تر شانس بیشتری برای بقا از طریق کپی کردن خود در نسل بعدی دارد.

برنامه‌نویسی ژنتیک

مثال: برنامه‌ای برای محاسبه‌ی قاعده‌ی فیثاغورس (۸ از ۱۲)

تقاطع در برنامه‌نویسی ژنتیک:

دو عبارت نمادین والد



(/ (− (*sqrt* (+ (* *a a*) (− *a b*))) *a*) (* *a b*))

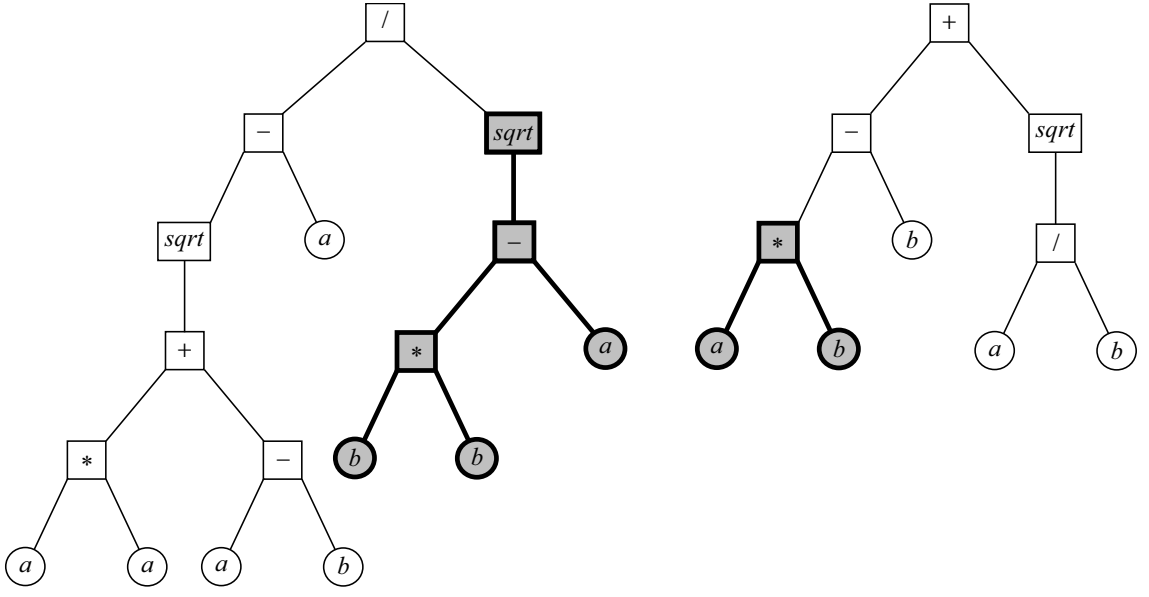
(+ (− (*sqrt* (− (* *b b*) *a*)) *b*) (*sqrt* (/ *a b*)))

برنامه‌نویسی ژنتیک

مثال: برنامه‌ای برای محاسبه‌ی قاعده‌ی فیثاغورس (۹ از ۱۲)

تقاطع در برنامه‌نویسی ژنتیک:

دو عبارت نمادین فرزند



(/ (− (*sqrt* (+ (* *a a*) (− *a b*))) *a*) (*sqrt* (− (* *b b*) *a*)))

(+ (−(* *a b*) *b*) (*sqrt* (/ *a b*)))

برنامه‌نویسی ژنتیک

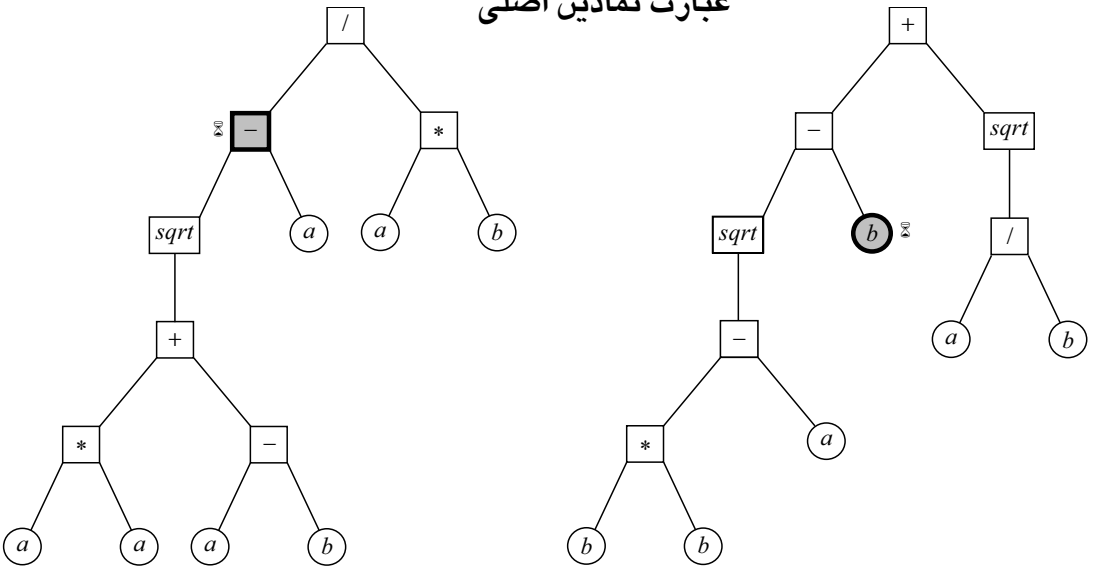
مثال: برنامه‌ای برای محاسبه‌ی قاعده‌ی فیثاغورس (۱۰ از ۱۲)

جهش در برنامه‌نویسی ژنتیک:

یک عملگر جهش می‌تواند به‌طور تصادفی هر تابع یا پایانه‌ای در عبارت نمادین لیسپ را تغییر دهد.

در جهش: یک تابع تنها می‌تواند با یک تابع و یک پایانه تنها می‌تواند با یک پایانه جایگزین شود.

عبارت نمادین اصلی



(/ (− (*sqrt* (+ (∗ *a a*) (− *a b*))) *a*) (∗ *a b*))

(+ (− (*sqrt* (− (∗ *b b*) *a*)) *b*) (*sqrt* (/ *a b*)))

برنامه‌نویسی ژنتیک

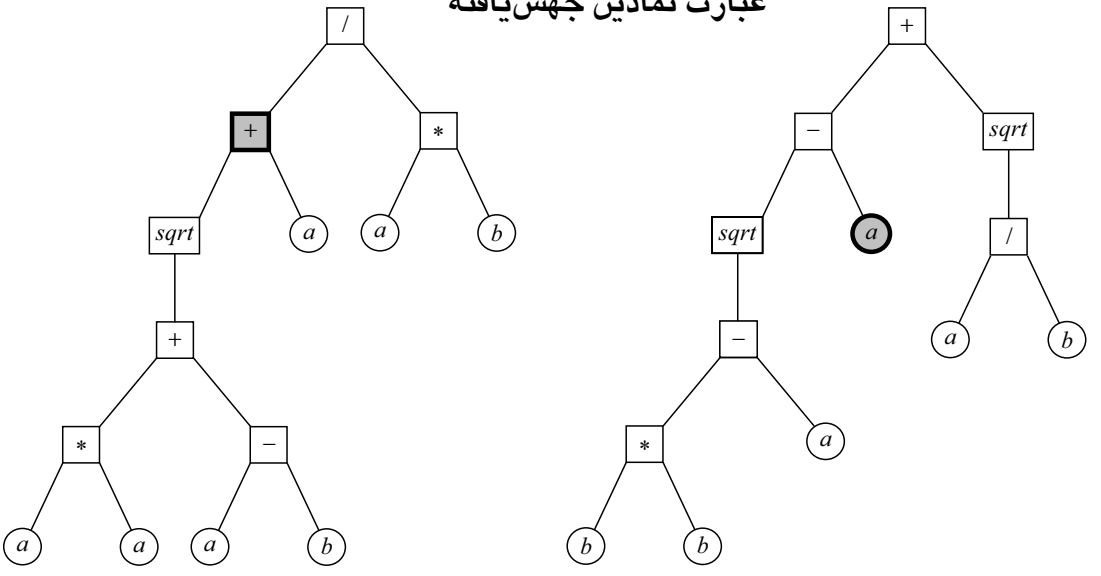
مثال: برنامه‌ای برای محاسبه‌ی قاعده‌ی فیثاغورس (۱۱ از ۱۲)

جهش در برنامه‌نویسی ژنتیک:

یک عملگر جهش می‌تواند به‌طور تصادفی هر تابع یا پایانه‌ای در عبارت نمادین لیسپ را تغییر دهد.

در جهش: یک تابع تنها می‌تواند با یک تابع و یک پایانه تنها می‌تواند با یک پایانه جایگزین شود.

عبارت نمادین جهش‌یافته



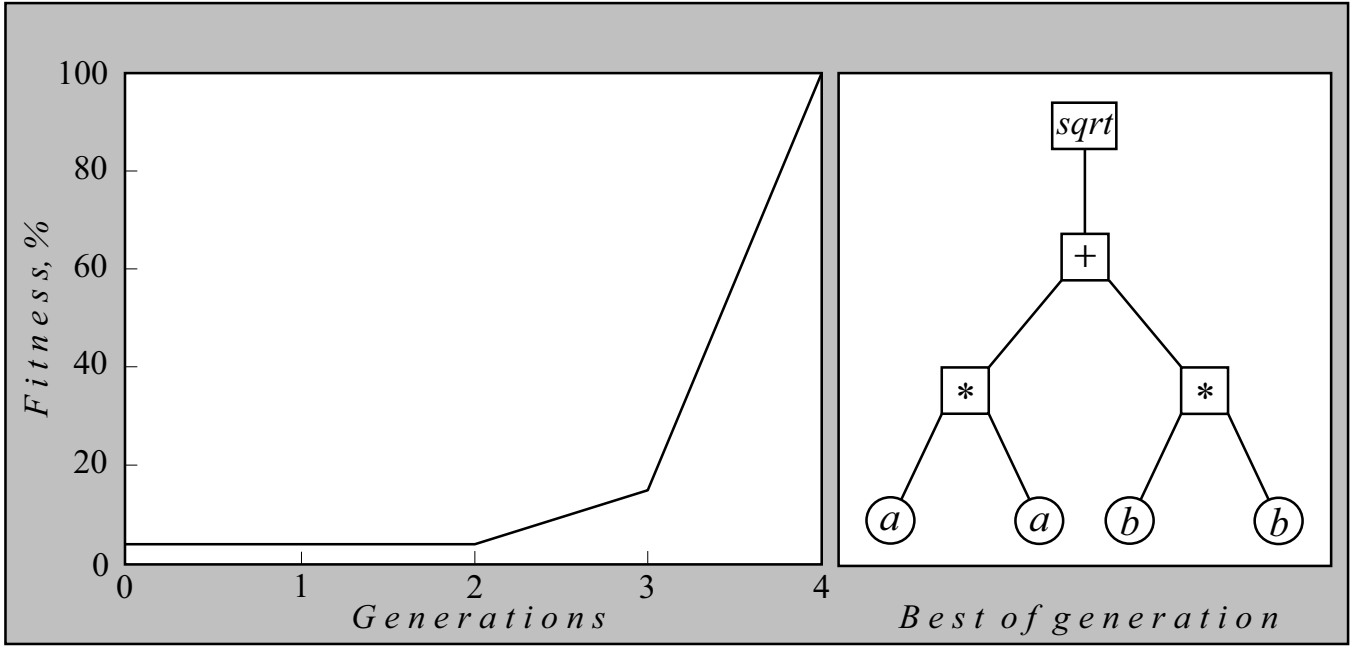
(/ (+ (*sqrt* (+ (* *a a*) (− *a b*))) *a*) (* *a b*))

(+ (− (*sqrt* (− (* *b b*) *a*)) *a*) (*sqrt* (/ *a b*)))

برنامه‌نویسی ژنتیک

مثال: برنامه‌ای برای محاسبه‌ی قاعده‌ی فیثاغورس (۱۲ از ۱۲)

تاریخچه‌ی برازش برای بهترین عبارت نمادین در هر نسل



برنامه‌نویسی ژنتیک

گام‌های برنامه‌نویسی ژنتیک

۱) * انتساب ماکزیمم تعداد نسل‌ها برای اجرا
* انتساب احتمالات برای کپی والد (cloning)، تقاطع، جهش (مجموع احتمالات باید مساوی یک شود.)

↓

۲) تولید یک جمعیت آغازین از برنامه‌های کامپیوتری به اندازه‌ی N
(با ترکیب توابع و پایانه‌های انتخاب شده به طور تصادفی)

↓

۳) اجرای هر برنامه در جمعیت، محاسبه‌ی برازش آنها، تعیین بهترین فرد تاکنون

↓

۴) انتخاب یک عملگر ژنتیکی بر اساس احتمالات نسبت‌دهی شده برای کپی والد، تقاطع، جهش

↓

۵) * اگر عملگر کپی والد انتخاب شد: انتخاب یک برنامه از جمعیت فعلی و کپی آن در جمعیت جدید
* اگر عملگر تقاطع انتخاب شد: انتخاب یک جفت برنامه، ایجاد یک جفت فرزند و قرار دادن در جمعیت جدید
* اگر عملگر جهش انتخاب شد: انتخاب یک برنامه، انجام جهش و قرار دادن در جمعیت جدید

↓

۶) تکرار (۴) تا رسیدن اندازه‌ی جمعیت جدید به اندازه‌ی جمعیت آغازین N

↓

۷) جایگزین کردن جمعیت جدید (والد) با جمعیت جدید (فرزند)

↓

۸) تکرار از (۳) تا «ارضا شدن شرط خاتمه»

برنامه‌نویسی ژنتیک

در مقایسه با الگوریتم ژنتیک

برنامه‌نویسی ژنتیک	الگوریتم ژنتیک
هر دو روی‌کرد تطوری را استفاده می‌کنند.	
تولید مثل برنامه‌های کامل برای حل یک مسئله‌ی خاص	تولید مثل رشته‌های بیتی کدگذاری راه‌حل
استفاده از بلوک‌های ساختمانی سطح بالا	مشکل بازنمایی مسئله در کد با طول ثابت
امکان بازنمایی متنوع از نظر طول و پیچیدگی	محدود شدن قدرت GA با بازنمایی ضعیف
عملکرد خوب کاربردها و موارد متعدد	مصنوعی بودن کدگذاری با طول ثابت

کاربردهای محاسبات تطوری

- تولید و بهینه‌سازی موسیقی
- هنر الگوریتمی
- تعیین موقعیت بهینه‌ی یک مرکز بحران
- طراحی بهینه‌ی آنتن آرایه‌ای
- حل مسائل بهینه‌سازی ترکیبیاتی (مثل TSP)
- رمزگشایی
- طرح‌ریزی مسیر حرکت ربات
- طراحی اجسام رادارگریز
- ساخت مدل‌های معکوس دینامیکی (خطی/غیرخطی)
- حل معادلات دیفرانسیل با مشتقات جزئی غیرخطی از مرتبه‌ی بالا
- ...

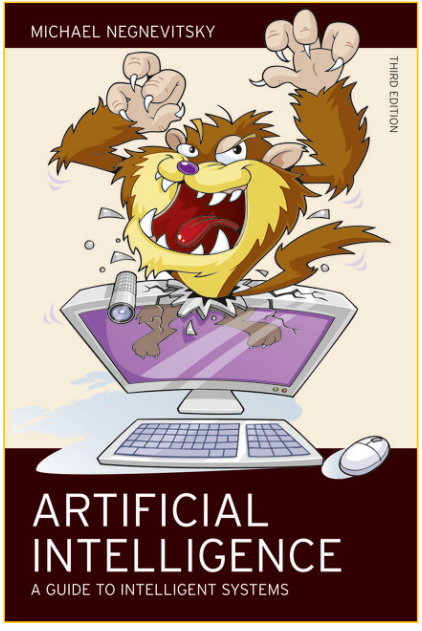
هوش مصنوعی

هوش تطوری

۵

منابع، مطالعه، تکلیف

منبع اصلی



Michael Negnevitsky,
Artificial Intelligence: A Guide to Intelligent Systems,
Pearson Education Canada, 2011.
Chapter 7

Evolutionary computation 7

In which we consider the field of evolutionary computation, including genetic algorithms, evolution strategies and genetic programming, and their applications to machine learning.

7.1 Introduction, or can evolution be intelligent?

Intelligence can be defined as the capability of a system to adapt its behaviour to an ever-changing environment. According to Alan Turing (Turing, 1950), the form or appearance of a system is irrelevant to its intelligence. However, from our everyday experience we know that evidences of intelligent behaviour are easily observed in humans. But we are products of evolution, and thus by modelling the process of evolution, we might expect to create intelligent behaviour. Evolutionary computation simulates evolution on a computer. The result of such a simulation is a series of optimisation algorithms, usually based on a simple set of rules. Optimisation iteratively improves the quality of solutions until an optimal, or at least feasible, solution is found.

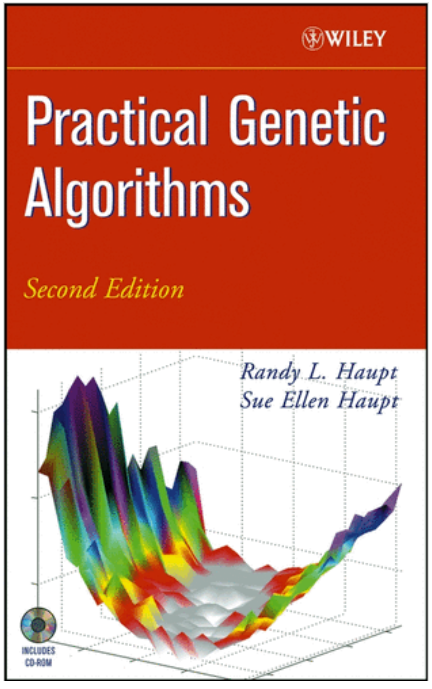
But is evolution really intelligent? We can consider the behaviour of an individual organism as an inductive inference about some yet unknown aspects of its environment (Fogel *et al.*, 1966). Then if, over successive generations, the organism survives, we can say that this organism is capable of learning to predict changes in its environment. Evolution is a tortuously slow process from the human perspective, but the simulation of evolution on a computer does not take billions of years!

The evolutionary approach to machine learning is based on computational models of natural selection and genetics. We call them **evolutionary computation**, an umbrella term that combines **genetic algorithms**, **evolution strategies** and **genetic programming**. All these techniques simulate evolution by using the processes of **selection**, **mutation** and **reproduction**.

7.2 Simulation of natural evolution

On 1 July 1858, Charles Darwin presented his theory of evolution before the Linnean Society of London. This day marks the beginning of a revolution in

منبع دوم



R.L. Haupt, S.E. Haupt.
Practical Genetic Algorithms,
Second Edition, John Wiley & Sons, 2004.

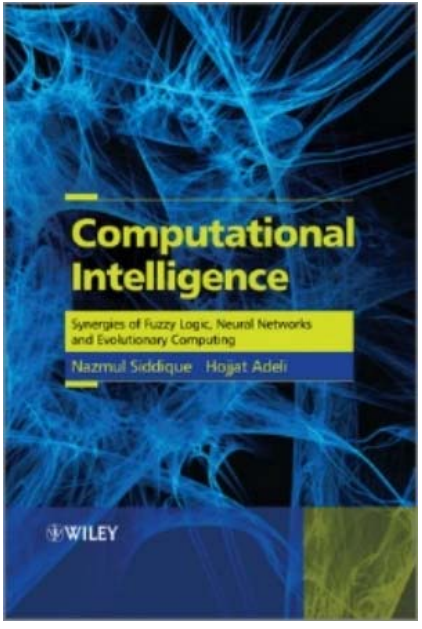
PRACTICAL GENETIC ALGORITHMS

SECOND EDITION

Randy L. Haupt
Sue Ellen Haupt

WILEY-INTERSCIENCE
A JOHN WILEY & SONS, INC., PUBLICATION

منبع سوم



Nazmul Siddique, Hojjat Adeli,
Computational Intelligence: Synergies of Fuzzy Logic, Neural Networks and Evolutionary Computing,
John Wiley & Sons, 2013.
Chapter 6

6

Evolutionary Computing

6.1 Introduction

We see a diversity of life on earth – millions of species each with its own unique behaviour patterns and characteristics or traits. All of these plants, animals, birds, fishes and other creatures have evolved, and continue evolving, over millions of years. They have adapted themselves to a constantly shifting and changing environment in order to survive. Those weaker and less fit members of species tend to die away, leaving the stronger and fitter to mate, create offspring and ensure the continuing survival of the species. Their lives are dictated by the laws of natural selection and Darwinian evolution – struggle for existence and survival of the fittest. Such an evolutionary process is shown in Figure 6.1.

And it is upon these ideas that evolutionary computing (EC) is based. Evolutionary computing is the emulation of the process of natural selection in a search procedure. In nature, organisms have certain characteristics that influence their ability to survive and reproduce. These characteristics are represented by encoding of information contained in the chromosomes of the organisms. New offspring chromosomes are created by means of mating and reproduction mechanisms. The end result will be offspring chromosomes that contain the best characteristics of each parent's chromosomes, which enable them to survive in an adverse environment. The process of natural selection ensures that more fit individuals have the opportunity to mate most of the time, leading to the expectation that the offspring will have similar or better fitness.

6.2 Evolutionary Computing

The population can be viewed simply as a collection of interacting creatures. As each generation of creatures comes and goes, the weaker ones tend to die away without producing children, while the stronger mate in the process of recombination to produce new and perhaps unique children with attributes from both parents to continue the evolutionary process. In nature, a diverse population within a species tends to allow the species to adapt to its environment

Computational Intelligence: Synergies of Fuzzy Logic, Neural Networks and Evolutionary Computing, First Edition.
Nazmul Siddique and Hojjat Adeli.