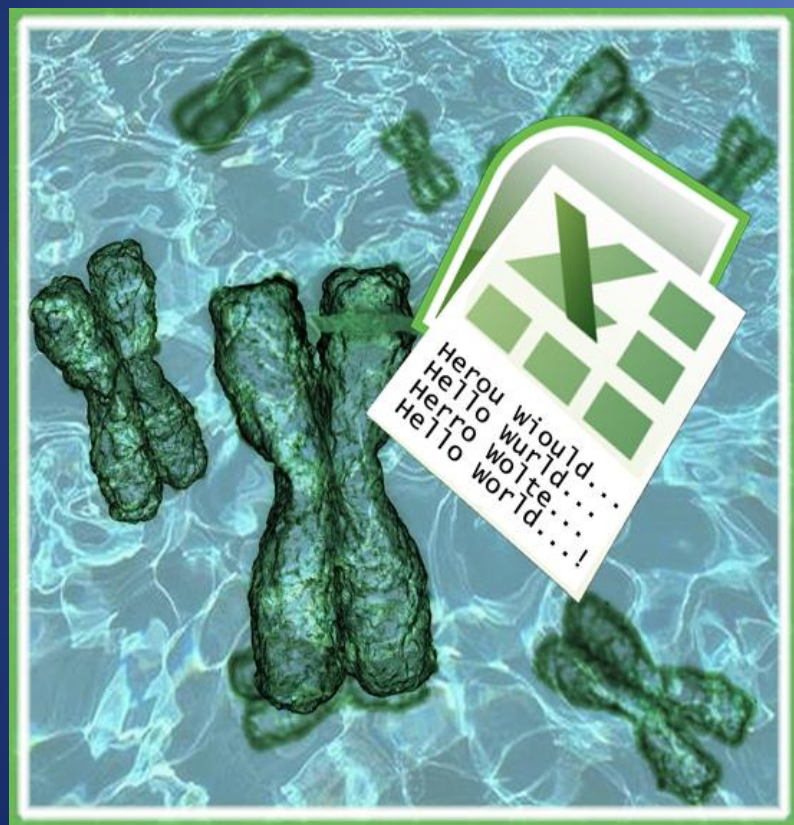


الگوریتم ژنتیک در حل مسائل بهینه‌سازی

(مقدمه‌ای بر الگوریتم ژنتیک چند هدفه)



مدرس:

دکتر سید امیر رضا ابطحی

عضو هیأت علمی و مدیر گروه علوم
تصمیم و مهندسی دانش

دانشگاه علوم اقتصادی

(پیاده‌سازی در محیط نرم‌افزار Excel)

اردیبهشت ۱۳۹۱

فهرست مطالب

1. بخش اول: الگوریتم ژنتیک تک هدفه

1. مقدمه
2. ساختار پاسخ
3. تابع برازندگی
4. عملگرهای الگوریتم ژنتیک
5. اعمال محدودیت‌های مسأله

2. بخش دوم: الگوریتم ژنتیک چند هدفه

1. مقدمه‌ای بر بهینه‌سازی چند هدفه
2. مرز پارتو در بهینه‌سازی چند هدفه
3. الگوریتم NSGA-II

بخش اول

مقدمه‌ای بر الگوریتم ژنتیک

شروع با یک رؤیا...



- فرض کنید با یک مسأله روبرو هستید
- نمی‌دانید چطور آنرا حل کنید
- چه کاری می‌توانید انجام دهید؟
- می‌توان به یک طریقی از کامپیوتر برای یافتن پاسخ مسأله استفاده کرد؟

یک راه حل اولیه

• الگوریتم « تولید و تست کورکورانه »:

Repeat

Generate a random possible solution

Test the solution and see how good it is

Until solution is good enough



می‌توانیم از این ایده خام استفاده کنیم؟



- برخی اوقات - بله:

- اگر تعداد پاسخ‌های ممکن کم باشند
- و شما زمان کافی داشته باشید
- چنین روشی می‌تواند استفاده شود

- برای اغلب مسائل - خیر:

- تعدد پاسخ‌های ممکن
- نداشتن زمان کافی برای تست همه آنها
- پس این روش نمی‌تواند استفاده شود

یک راه حل دیگر

Generate a *set* of random solutions

Repeat

- Test each solution in the set (rank them)

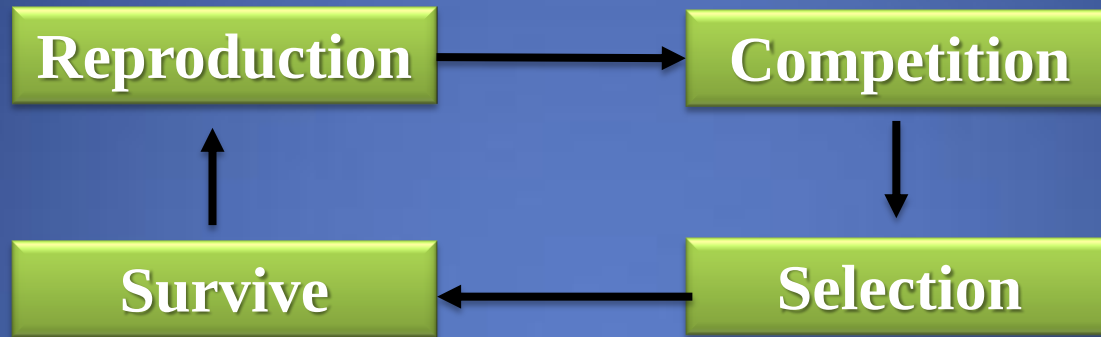
- Remove some bad solutions from set

- Duplicate some good solutions

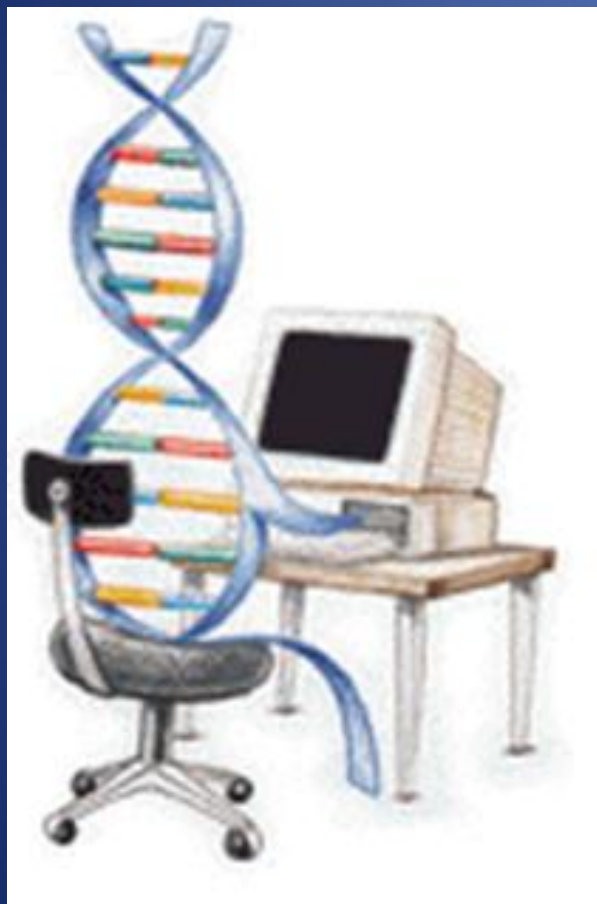
- make small changes to some of them

Until best solution is good enough

نظريه تكامل داروين



الگوریتم ژنتیک



- الگوریتم ژنتیک (GA) روشی است که از اصول تکامل و ژنتیک در جستجوی میان پاسخهای ممکن الهام گرفته است.

- ایده اصلی GA شبیه‌سازی عملگرهای فرایند تکامل در سیستم‌های طبیعی است همچون:

- وراثت (Inheritance)

- انتخاب (Selection)

- جهش (Mutation)

- تقاطع یا تولیدمثل (Crossover)

بقاء اصلح (Survival of Fittest)

- اصل اساسی تکامل که در GA استفاده شده است « اصل بقاء اصلح » است.
- پاسخ خوب نجات می‌یابد، در حالیکه پاسخ بد می‌میرد



بازگشت به الگوریتم

Generate a *set* of random solutions

Repeat

- Test each solution in the set (rank them)

- Remove some bad solutions from set

- Duplicate some good solutions

 - make small changes to some of them

Until best solution is good enough

تکمیل الگوریتم

• با وجود اینکه این الگوریتم ممکن است برای برخی از مسائل و فضاهاى جستجوی ساده کار کند، اما همچنان بسیار خام است:

- ساختار پاسخ‌های تولیدی چگونه است؟
- جستجو در چه فضایی باید انجام شود؟
- بهتر یا بدتر بودن پاسخ‌ها بر مبنای چه معیاری است؟
- منظور از انتخاب، تغییر و تقویت پاسخ‌های بهتر چیست؟
- چه تعداد پاسخ باید تست شوند؟

چگونه ساختار پاسخ را تعیین می کنید؟

- نحوه تعیین ساختار پاسخ (Coding) چیست؟

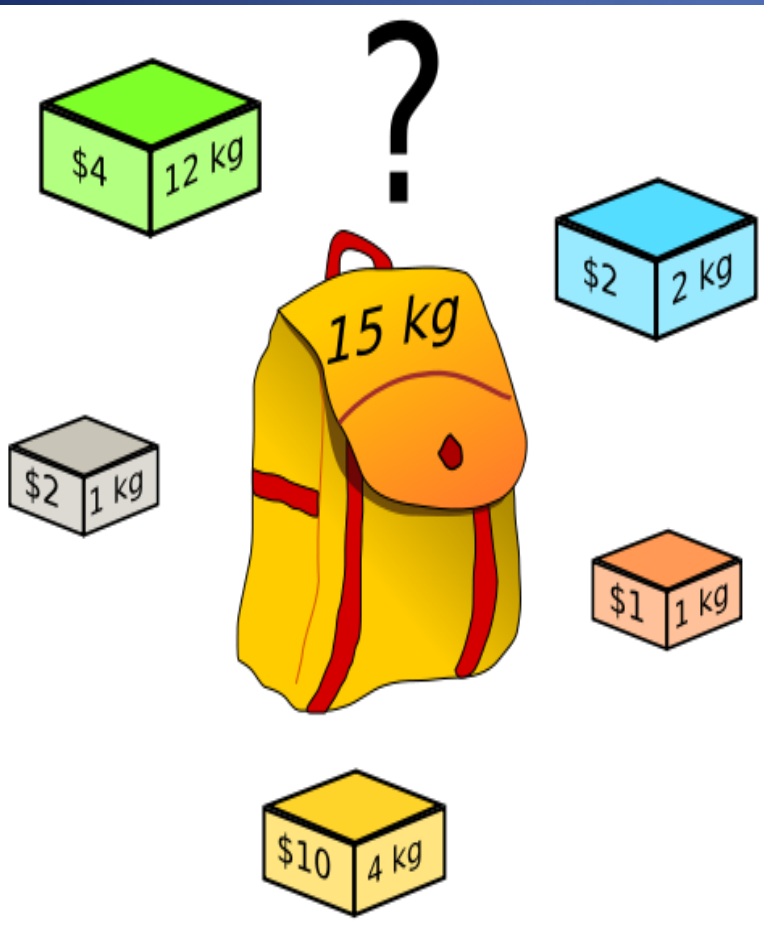
- واضح است که این موضوع بستگی به نوع مسأله دارد
- در GA معمولاً پاسخ به صورت آرایه های عددی با طول ثابت کد می شود

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

15	4	9	0	12	4	3	0
----	---	---	---	----	---	---	---

- هر بیت یکی از جنبه های پاسخ را نمایش می دهد
- در فرایند الگوریتم ژنتیک، باید بتوانیم هر رشته (String) را تست کنیم و به آن امتیازی بدهیم که نشان دهنده میزان خوب بودن پاسخ می باشد

مسأله کوله پشتی (بیان مسأله)



- یک کوله‌پشتی با وزن محدود و مشخص
- تعدادی از اقلام دارای وزن معین و سود مشخص
- اقلام بخش‌پذیر نیستند (یا انتخاب می‌شوند یا خیر)
- کدام اقلام را برای قرار دادن در کوله پشتی انتخاب کنیم تا سود بیشینه شود

مسأله کوله‌پشتی (یک مثال)

اقلام	۱	۲	۳	۴	۵	۶	۷
سود	۵	۸	۳	۲	۷	۹	۴
وزن	۷	۸	۴	۱۰	۴	۶	۴



- ظرفیت کوله‌پشتی برابر است با ۲۲ کیلوگرم
- کوله‌پشتی را چگونه پر کنیم که بیشترین سود حاصل شود!!

مدل ریاضی مسأله کوله پشتی

$$\text{Max} \sum_i b_i x_i$$

s.t.

$$\sum_i w_i x_i \leq W$$

$$x_i \in \{0,1\}$$

سود آیتم i ام	b_i
وزن آیتم i ام	w_i
وزن کل قابل حمل	W
متغیر باینری	x_i

مسأله کوله‌پشتی (ساختار پاسخ)

- یک آرایه باینری به طول هفت خانه

شماره ارقام	1	2	3	4	5	6	7
ساختار پاسخ	0	1	0	0	1	1	1
	1	0	0	0	1	0	1
	0	1	1	1	0	1	0
	1	1	1	1	0	0	0

- 1 نشاندهنده انتخاب قلم مربوطه و 0 نشاندهنده عدم انتخاب
- در اینجا نیز هر سطر یک کروموزوم (یک پاسخ کاندیدا) و مجموعه کروموزوم‌ها یک جمعیت از پاسخ‌ها هستند.



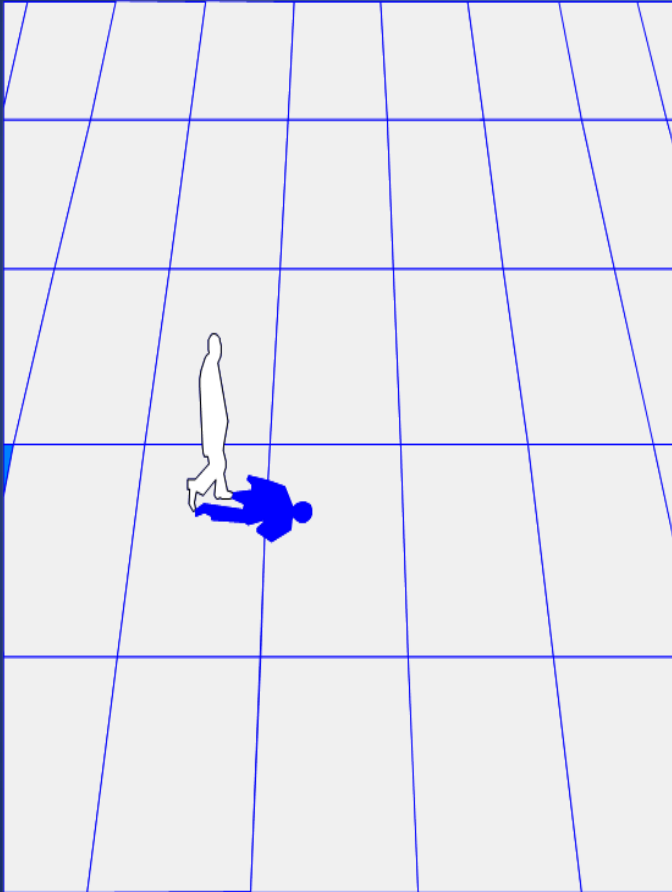
تابع برازندگی (Fitness Function)

- تابع برازندگی تابعی است که از طریق آن پاسخ‌های به دست آمده ارزیابی شده و مشخص می‌شود، کدام پاسخ از سایر پاسخ‌ها بهتر است



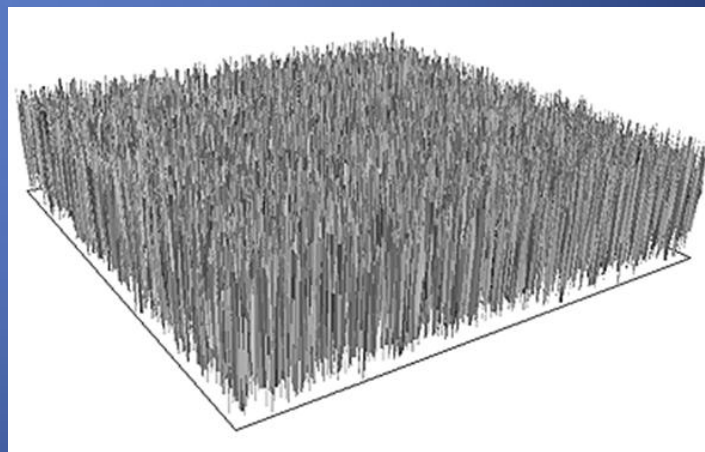
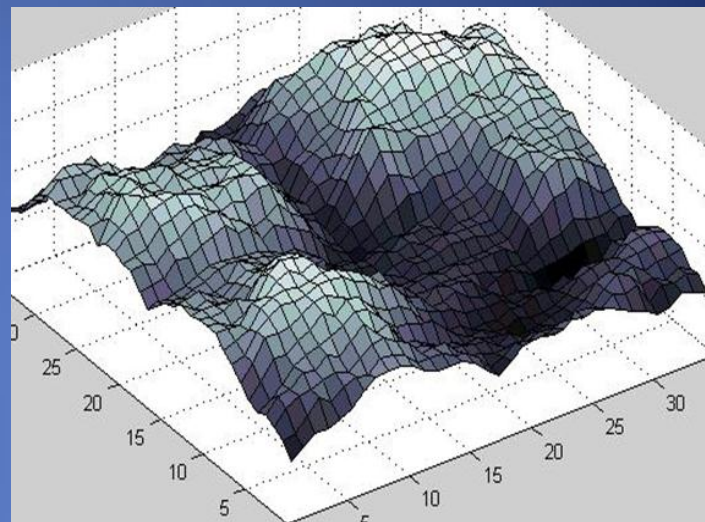
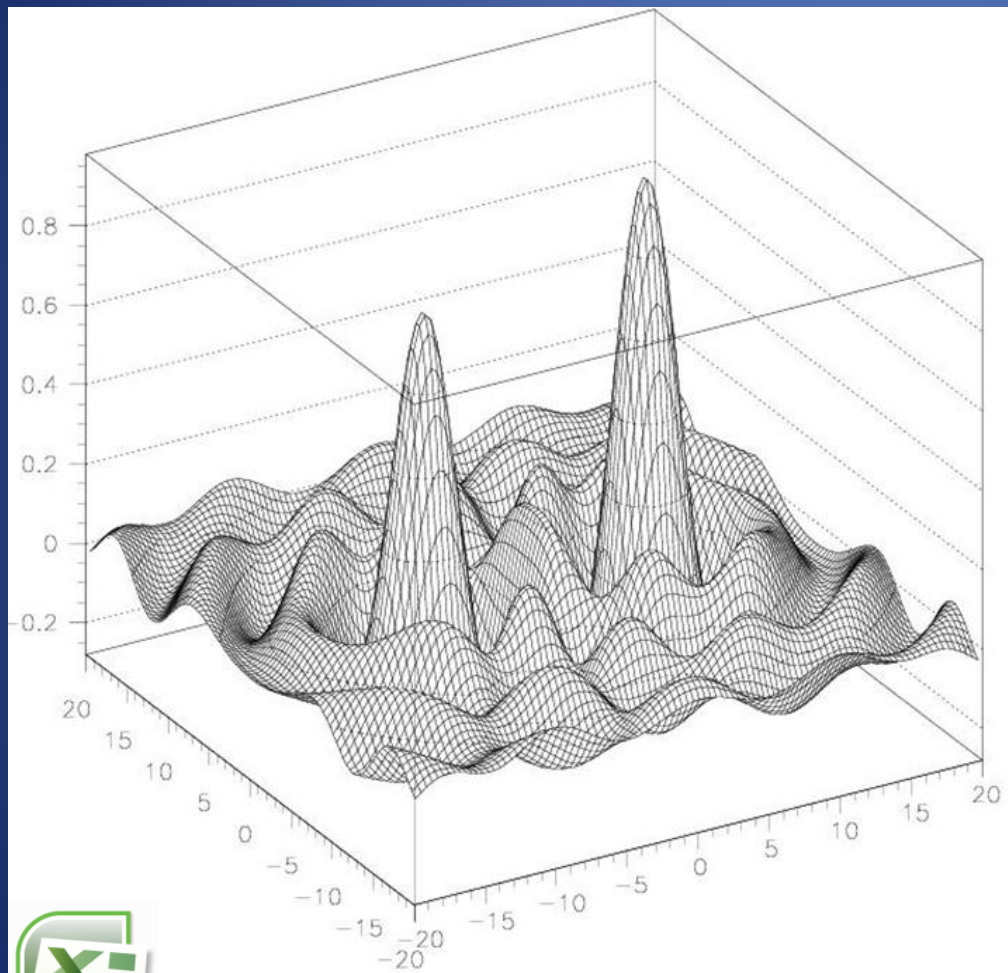
- میزان تابع برازندگی برای تمامی کروموزوم‌ها (پاسخ‌ها) محاسبه می‌شود
- تعیین تابع برازندگی دقیقاً بستگی به مسأله مورد نظر دارد

فضای جستجو



- برای تابع $f(x)$ فضای پاسخ یک بعدی است.
- اما برای برخی از مسائل فضای پاسخ باید چند بعدی باشد. مثلاً دو بعدی $f(x,y)$
- هر کروموزوم در حقیقت نقطه‌ای در فضای جستجو می‌باشد.
- GA به دنبال حرکت از یک نقطه به سمت مکان بهتری (برازندگی بیشتر) در فضای جستجو می‌باشد.

مقدار تابع برازندگی در فضای جستجو



تابع برازندگی در مسأله کوله‌پشتی

۷	۶	۵	۴	۳	۲	۱	اقلام
۴	۹	۷	۲	۳	۸	۵	سود
۴	۶	۴	۱۰	۴	۸	۷	وزن

$$Max \sum_i b_i x_i$$

شماره اقلام

1 2 3 4 5 6 7

ساختار پاسخ

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

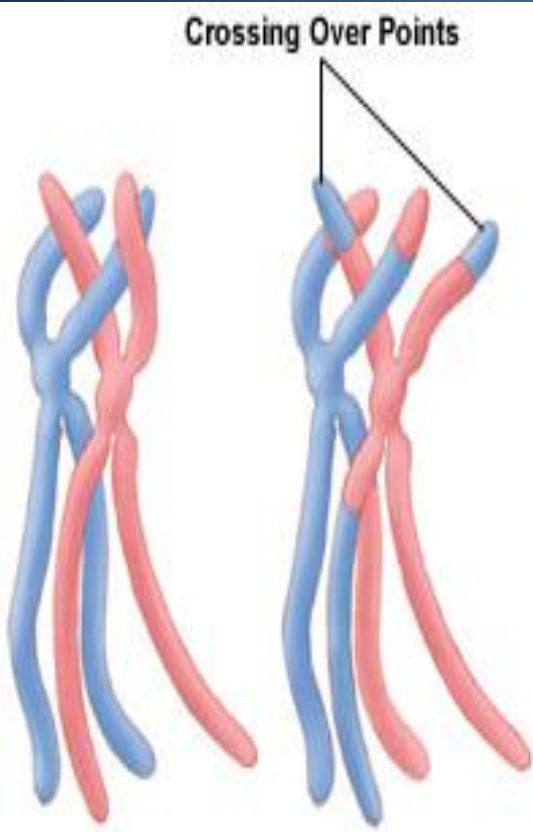


حرکت از یک نقطه به نقطه بهتر

- نقطه بعدی که در فضای پاسخ باید تست شود، کدام است؟
- GA پیشنهادات مشخصی در این زمینه دارد که از الگوی تکامل در سیستم‌های طبیعی تقلید شده است.
- عملگرهای الگوریتم ژنتیک (GA's Operators):
 - ترکیب دو والد (Crossover)
 - جهش در فرزند (Mutation)
 - انتخاب (Selection)

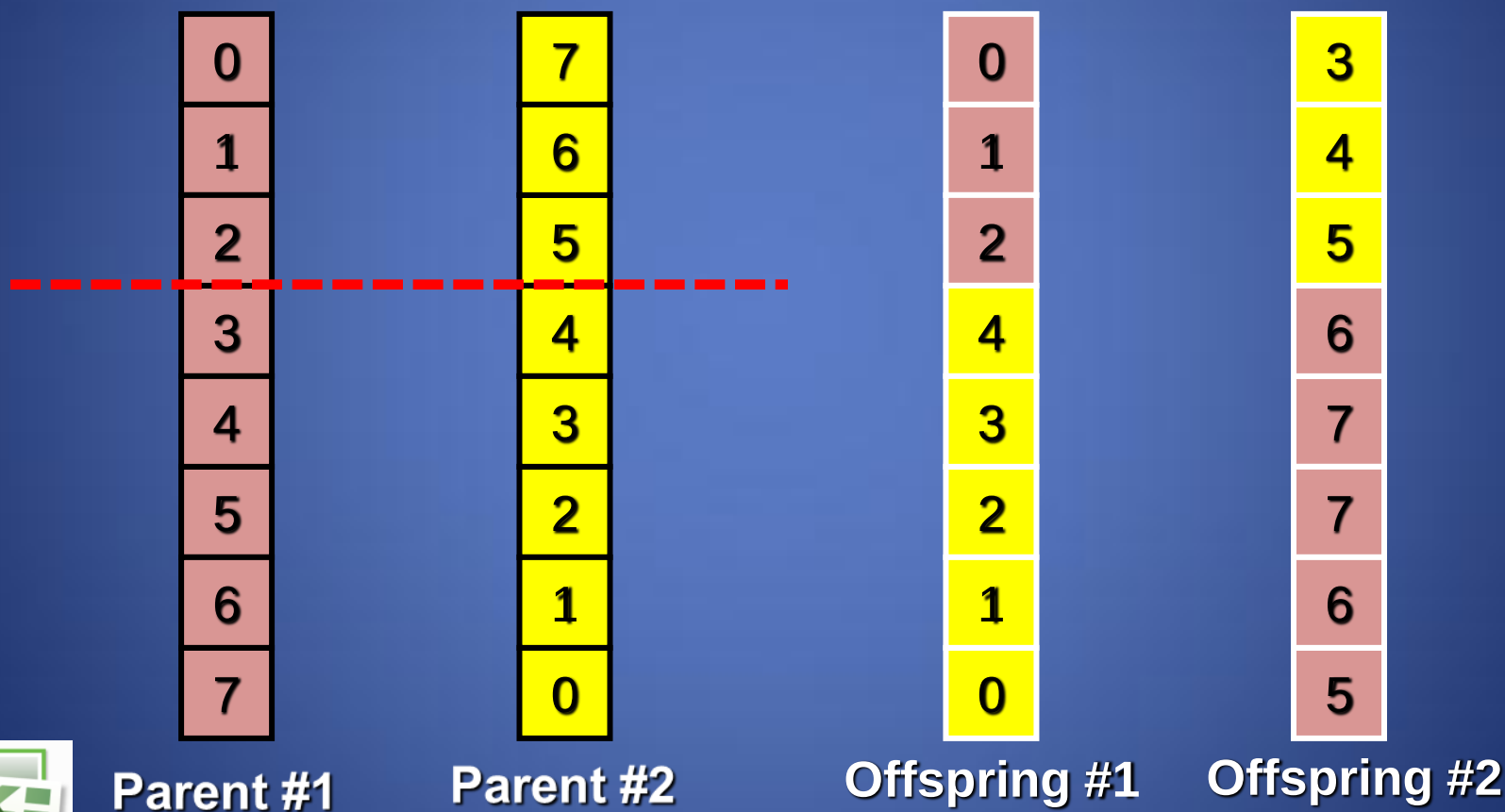
ترکیب (Crossover)

Crossing Over Points



- فرایند ترکیب یا تولید مثل (Reproduction)، ترکیب ژن‌های دو والد و تولید فرزند (Offspring) می‌باشد.
- در این فرایند کروموزوم‌های دو والد «خوب» انتخاب شده و با یک احتمال مشخصی (Crossover rate) با یکدیگر ترکیب می‌شوند.
- دو فرزند جدید در حقیقت دو نقطه جدید در فضای جستجو هستند که باید تست شوند.
- عملگر ترکیب باعث می‌شود تا پاسخ‌هایی از مناطق مختلف فضای جستجو مورد آزمون قرار بگیرند. وظیفه این عملگر ایجاد شرایطی برای پوشش هرچه بهتر فضای جستجو می‌باشد.

ترکیب تک نقطه‌ای (One-point Crossover)



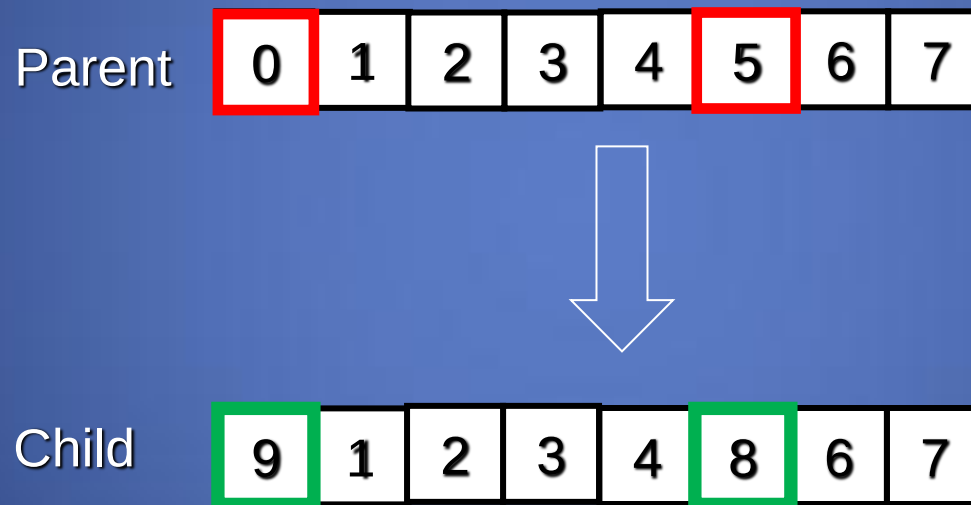
جهش (Mutation)



- جهش، تغییر یک یا چند ژن از کزوموزوم فرزند به صورت تصادفی است که با یک احتمال مشخصی (Mutation Rate) اعمال می‌شود.

- اعمال عملگر جهش در حقیقت، انجام جستجو در اطراف یک پاسخ جهت ایجاد بهبود در آن می‌باشد.

جهش (Mutation)



نرخ ترکیب و نرخ جهش

- برای تعیین نرخ و یا احتمال انجام پذیرفتن هر یک از عملگرها از اتفاقی که در سیستم‌های طبیعی می‌افتد الهام گرفته شده است

- احتمال متوسط برای اعمال عملگر ترکیب در اکثر موارد بالا (در حدود ۸۰ درصد) در نظر گرفته می‌شود.

- احتمال متوسط برای اعمال عملگر جهش نیز در اکثر موارد بسیار پایین (در حدود ۱-۲ درصد) در نظر گرفته می‌شود.



انتخاب (Selection)

- پس از تولید مثل و جهش، حال باید کروموزوم‌هایی برای انتقال به نسل بعدی انتخاب شوند.

- دو دیدگاه:

- فقط قوی‌ترین زنده می‌ماند

- کروموزومی انتخاب می‌شود که بالاترین مقدار برازندگی را داشته باشد

- امکانی برای زنده ماندن ضعیف‌ترها

- یک پاسخ ضعیف ممکن است مقدمه‌ای برای رسیدن به یک پاسخ قوی باشد

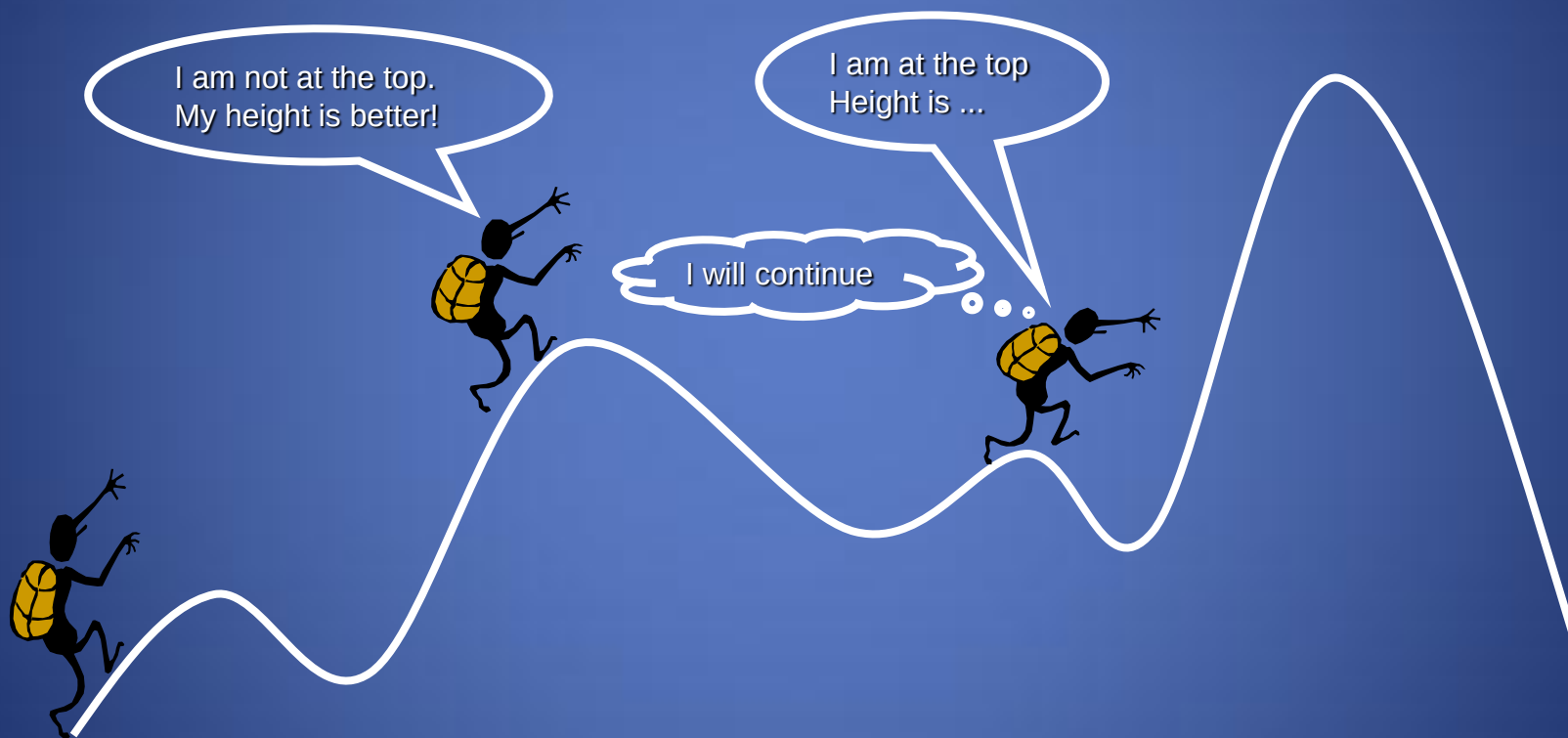
- پس احتمال انتخاب (هر چند اندک) برای پاسخ‌های ضعیف نیز در نظر بگیرید.



چرا ضعیف‌تر باید شانس انتخاب داشته باشد؟



چرا؟! زیرا...

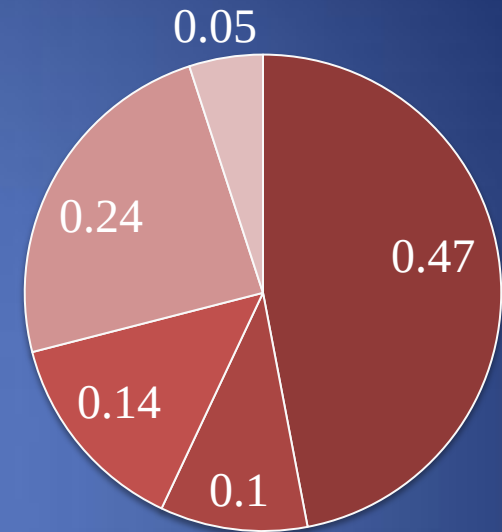


زیرا ...



انتخاب: چرخ رولت (Roulette Wheel)

Population	Fitness	Probability	C. P.
Chromosome #1	15	0.14	0.14
Chromosome #2	27	0.24	0.38
Chromosome #3	6	0.05	0.43
Chromosome #4	52	0.47	0.90
Chromosome #5	11	0.10	1



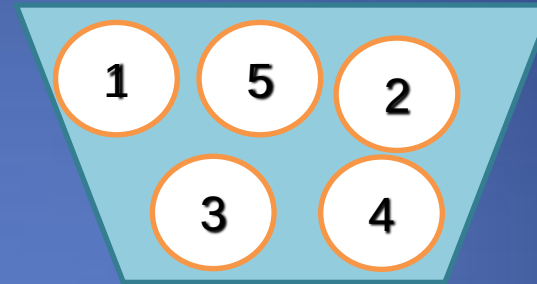
کروموزوم #۲ انتخاب می‌شود



یک عدد تصادفی بین $[0..1] = 0.21$

انتخاب: تورنومنت (tournament)

Population	Fitness
Chromosome #1	15
Chromosome #2	27
Chromosome #3	6
Chromosome #4	52
Chromosome #5	11



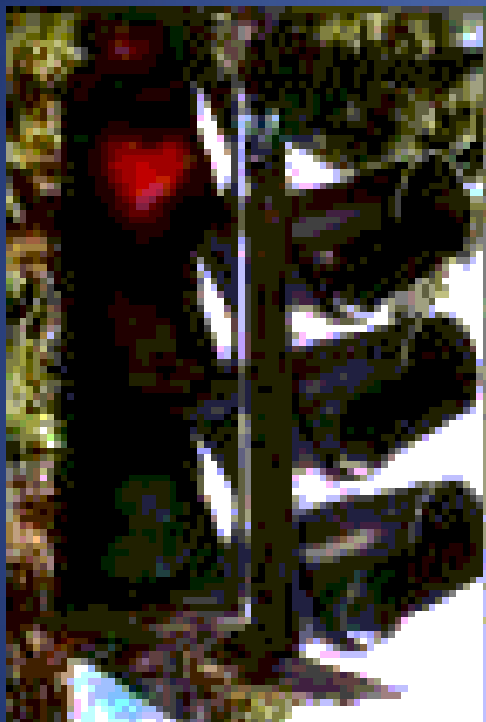
دو عدد تصادفی بین $[1..5]$ و 2 و 4



کروموزوم #4 انتخاب می شود



جستجو تا چه زمانی ادامه یابد؟ (Stopping Criteria)



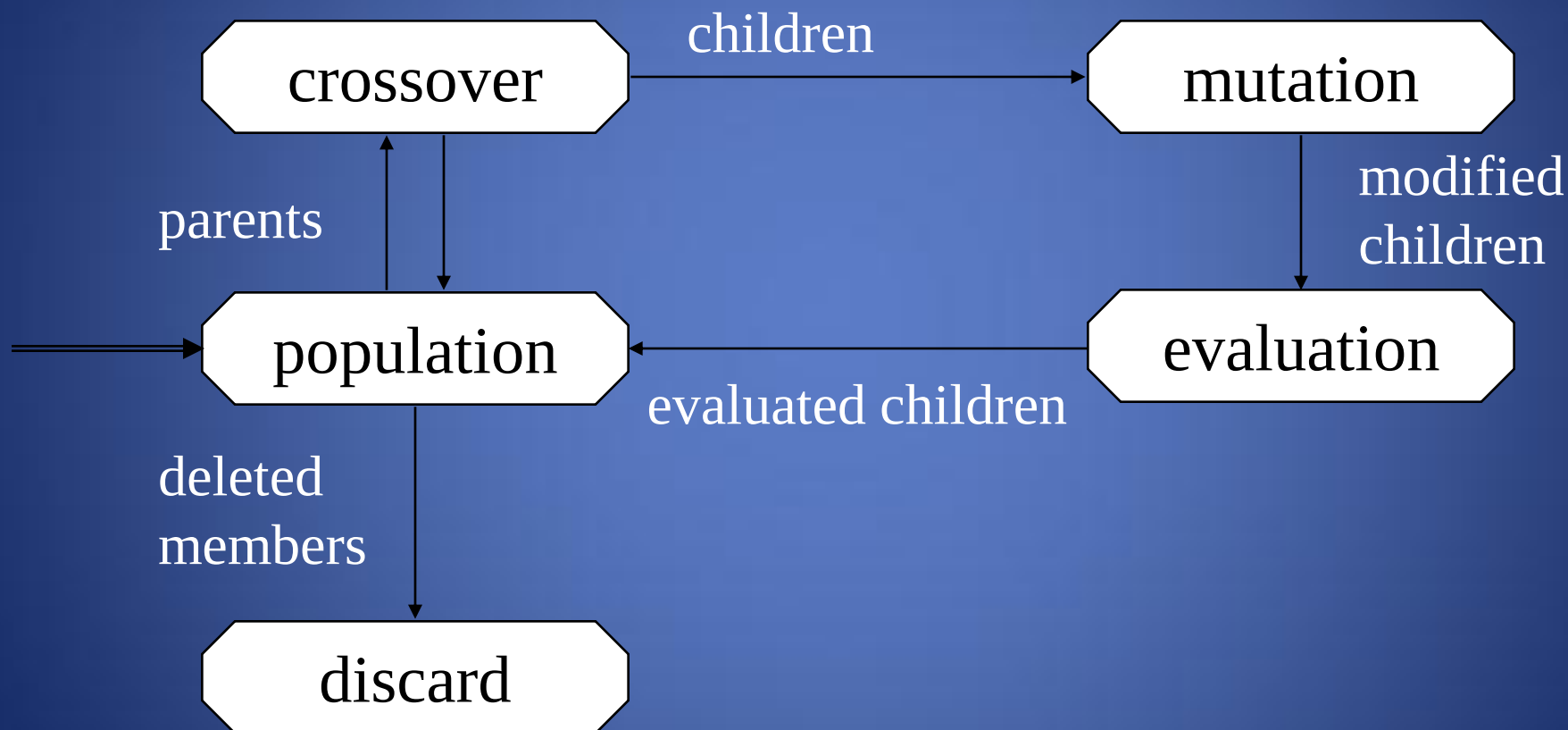
- چه زمانی باید به جستجو پایان داد؟

- دو دیدگاه:

- توقف پس از تولید تعداد مشخصی از نسل‌ها

- توقف در زمانی که با جستجوی بیشتر، دیگر بهبودی در مقدار تابع هدف حاصل نمی‌شود

چرخه الگوریتم ژنتیک



الگوریتم ژنتیک

Initialize population

Repeat

- Select parents

- Perform crossover

- Perform mutation

- Perform evaluation

- Perform selection

- Insert selected offspring into population

Until stopping criteria



اعمال محدودیت‌های مسأله

- محدودیت‌ها در واقع محدوده فضای موجه مسأله را مشخص می‌کنند.
- دو راه برای مواجهه با پاسخ‌هایی که از خارج فضای موجه هستند وجود دارد:
 - استراتژی حذف
 - استراتژی جریمه
- در استراتژی حذف، پاسخ‌های غیر موجه به سادگی حذف می‌شوند. با وجود سادگی در پیاده‌سازی اما این استراتژی چندان کارا نیست. (چرا؟!)
- در استراتژی جریمه در صورت خارج شدن الگوریتم از فضای موجه و تولید پاسخ‌های غیر موجه، الگوریتم به نحوی جریمه می‌شود.

استراتژی جریمه (Penalty Strategy)



- برای جلوگیری از تخطی (Violation) الگوریتم از فضای موجه باید پاسخ‌هایی (کروموزوم‌هایی) که غیر موجه هستند تضعیف شوند تا احتمال انتقال آن‌ها به نسل بعدی کمتر شود

- این تضعیف با دستکاری حساب‌شده در مقدار تابع برازش پاسخ‌های دارای تخطی (Violated Chromosomes) انجام می‌شود

- یک پاسخ را می‌توان به این صورت جریمه کرد:

- افزودن مقدار مشخصی به مقدار تابع برازش در مسائل مینیمم‌سازی
- کاستن مقدار مشخصی از مقدار تابع برازش در مسائل ماکزیمم‌سازی

- به این ترتیب با بدتر شدن مقدار برازش، شانس انتخاب پاسخ دارای تخطی و انتقال به نسل بعدی کاهش می‌یابد

محاسبه میزان تخطی

- فرض کنید محدودیتی به شکل زیر داشته باشیم:

$$y \leq y_0$$

- می‌خواهیم این محدودیت برقرار باشد. حال در صورت برقرار نبودن چه اتفاقی باید بیفتد.

- تخطی الگوریتم را می‌توان به صورت زیر تعریف کرد:

$$v(y \leq y_0) = \begin{cases} 0; & \text{if } y \leq y_0 \\ +; & \text{if } y > y_0 \end{cases}$$

- بنابراین زمانی که v مقدار مثبتی داشته باشد، به این معنا خواهد بود که پاسخ موجه نبوده، تخطی صورت گرفته و الگوریتم باید جریمه شود

محاسبه میزان تخطی

- در ساده‌ترین حالت میزان میزان مطلق تخطی را می‌توان به صورت زیر در نظر گرفته می‌شود

$$v = y - y_0$$

y	y_0	$y - y_0$	$\frac{y - y_0}{y_0}$
15	10	5	0.5
26	20	6	0.3
37	30	7	0.23
48	40	8	0.2

- بنابراین نرمال‌سازی باعث می‌شود بتوان تخطی را فارق از مقیاس تعیین کرد
- پس می‌توان مقدار تخطی را اینچنین تعیین کرد

$$v = \frac{y - y_0}{y_0}$$

اعمال محدودیت

- اگر یک مسأله بهینه‌سازی به صورت زیر داشته باشیم

$$\begin{array}{ll}\min & z \\ \text{s.t} & \\ & y \leq y_0\end{array}$$

- برای مسائل بهینه‌سازی دارای محدودیت، در کل، محدودیت‌ها حذف شده و به جای آن از تابع جریمه $P(v)$ استفاده می‌شود.

$$\min \hat{z} = z + P(v)$$

تعیین مقدار جریمه

- حال بر مبنای قانون تناسب مجازات و جرم، مقدار جریمه باید تابعی از میزان تخطی (v) باشد: $P(v)$

- مقدار تابع جریمه می‌تواند به سه شکل تعیین و اعمال شود:

- تابع جریمه جمع‌شونده

$$\hat{z} = z + \alpha v$$

- تابع جریمه ضرب‌شونده

$$\hat{z} = z(1 + \beta v)$$

- تابع جریمه ترکیبی

$$\hat{z} = z(1 + \beta v) + \alpha v$$

- α و β دو مقدار همیشه مثبت هستند که برای عملکرد صحیح الگوریتم باید تنظیم شوند





پایان بخش اول

بخش دوم

الگوریتم ژنتیک در بهینه‌سازی
چند هدفه

مقدمه‌ای بر بهینه‌سازی چند هدفه

- در مسائل بهینه‌سازی تک‌هدفه، ما به دنبال یافتن مقدار بهینه یک تابع هدف در فضای موجه بودیم.

- اما در بسیاری از مسائل دنیای واقعی ما با مسائلی روبرو هستیم که در آن‌ها به دنبال بهینه‌سازی چندین تابع هدف که بعضاً با یکدیگر در تضاد هستند می‌باشیم:

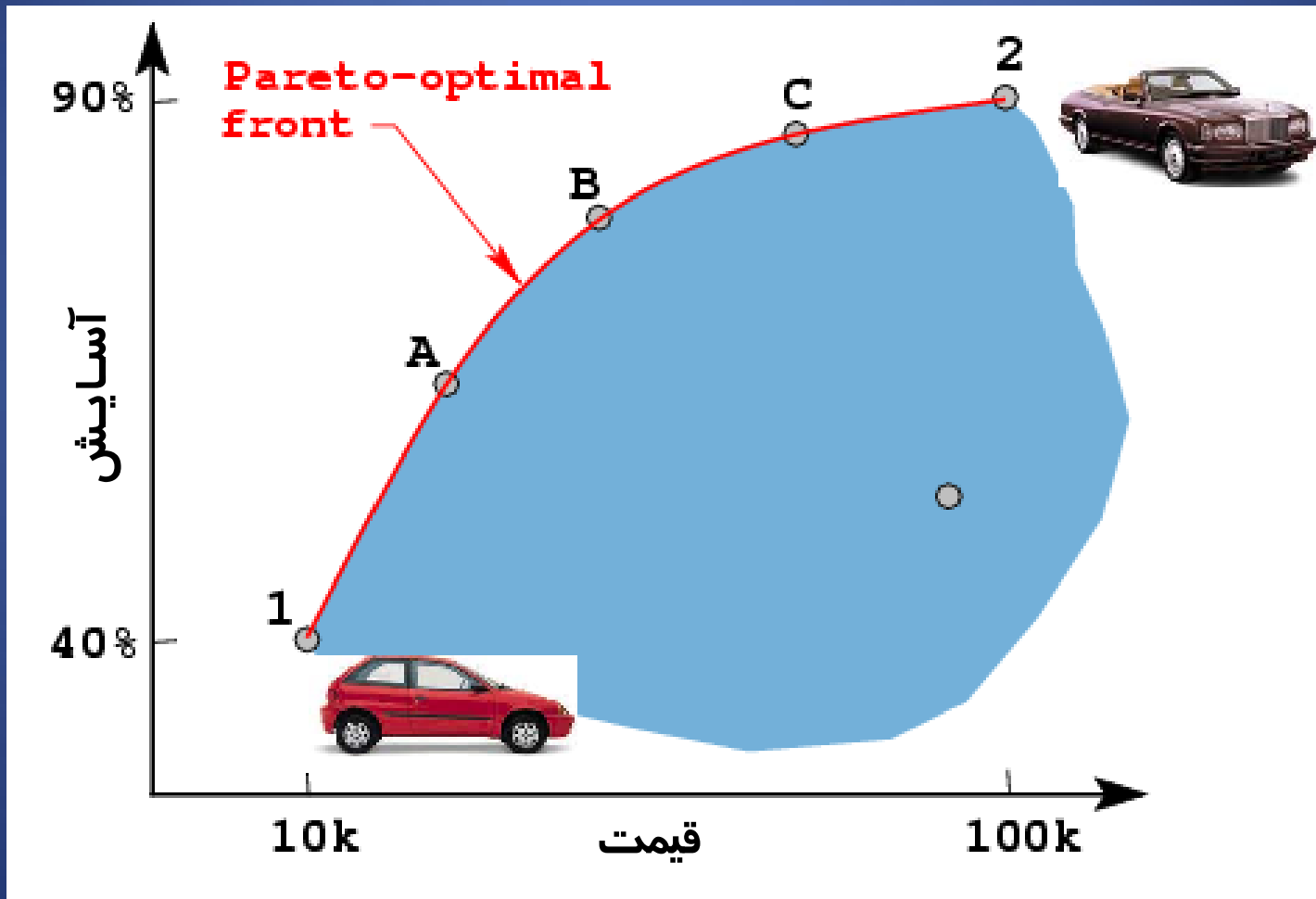
- مسائل زمان‌بندی پروژه (کمینه‌سازی زمان پروژه، کمینه‌سازی هزینه پروژه، بیشینه‌سازی کیفیت پروژه و ...)

- مسائل قابلیت اطمینان (بیشینه‌سازی قابلیت اطمینان سیستم، کمینه‌سازی وزن سیستم، کمینه‌سازی هزینه سیستم، ...)

- مسائل تعیین سبد پروژه‌ها (بیشینه‌سازی سود، کمینه‌سازی بودجه مورد نیاز، ...)



در زندگی روزانه نیز معمولاً با این نوع مسائل مواجه هستیم



مدل بهینه‌سازی چند هدفه

- یک مسأله بهینه‌سازی چند هدفه را می‌توان به صورت زیر فرموله کرد

$$\text{Min } f(\vec{x}) = \{f_1(\vec{x}), f_2(\vec{x}), \dots, f_3(\vec{x})\}$$

s.t.

$$g(\vec{x}) \leq b$$

- $\vec{x} = (x_1, x_2, \dots, x_n)$ بردار متغیرهای تصمیم در فضای پاسخ n بعدی است.

روش‌های مواجهه با مسائل چند هدفه

- از منظر درگیر کردن تصمیم‌گیر (Decision Maker) در فرایند حل، روش‌ها را می‌توان به این سه دسته تقسیم‌بندی نمود:

– پیش از فرایند جستجو:

- دریافت ترجیحات DM در مورد اهمیت اهداف پیش از فرایند جستجو (Defining single objective)

– حین فرایند جستجو:

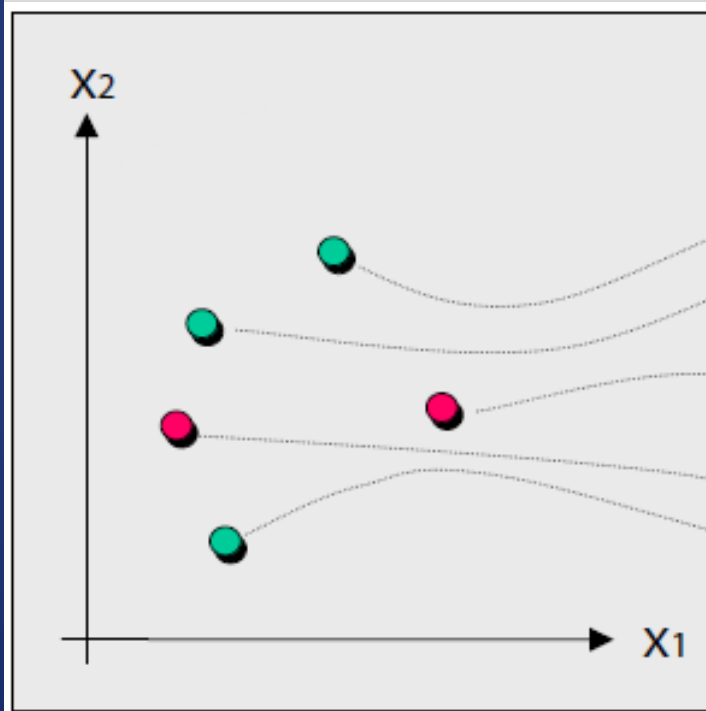
- دریافت ترجیحات DM در حین فرایند جستجو (Guiding search interactively)

– پس از فرایند جستجو:

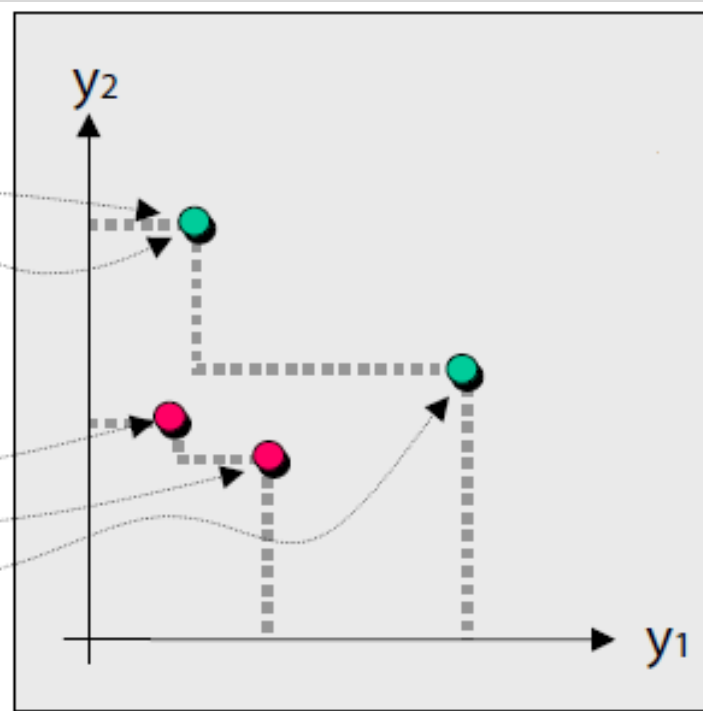
- دادن امکان انتخاب به DM با ارائه پاسخ‌هایی که از لحاظ فنی بر یکدیگر ترجیحی ندارند (Finding/Approximating Pareto Set)

کدام پاسخ بهینه است؟

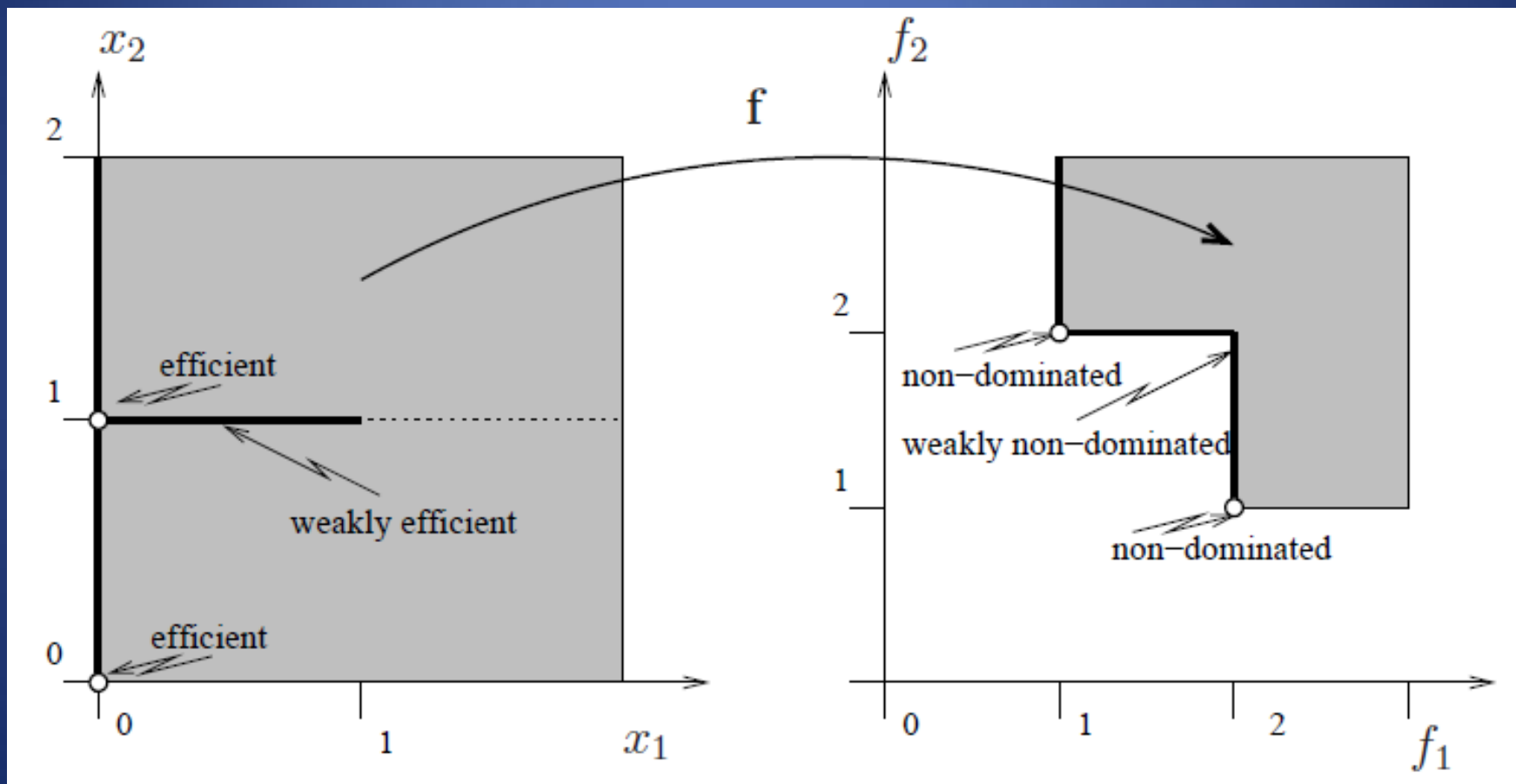
فضای متغیرهای تصمیم



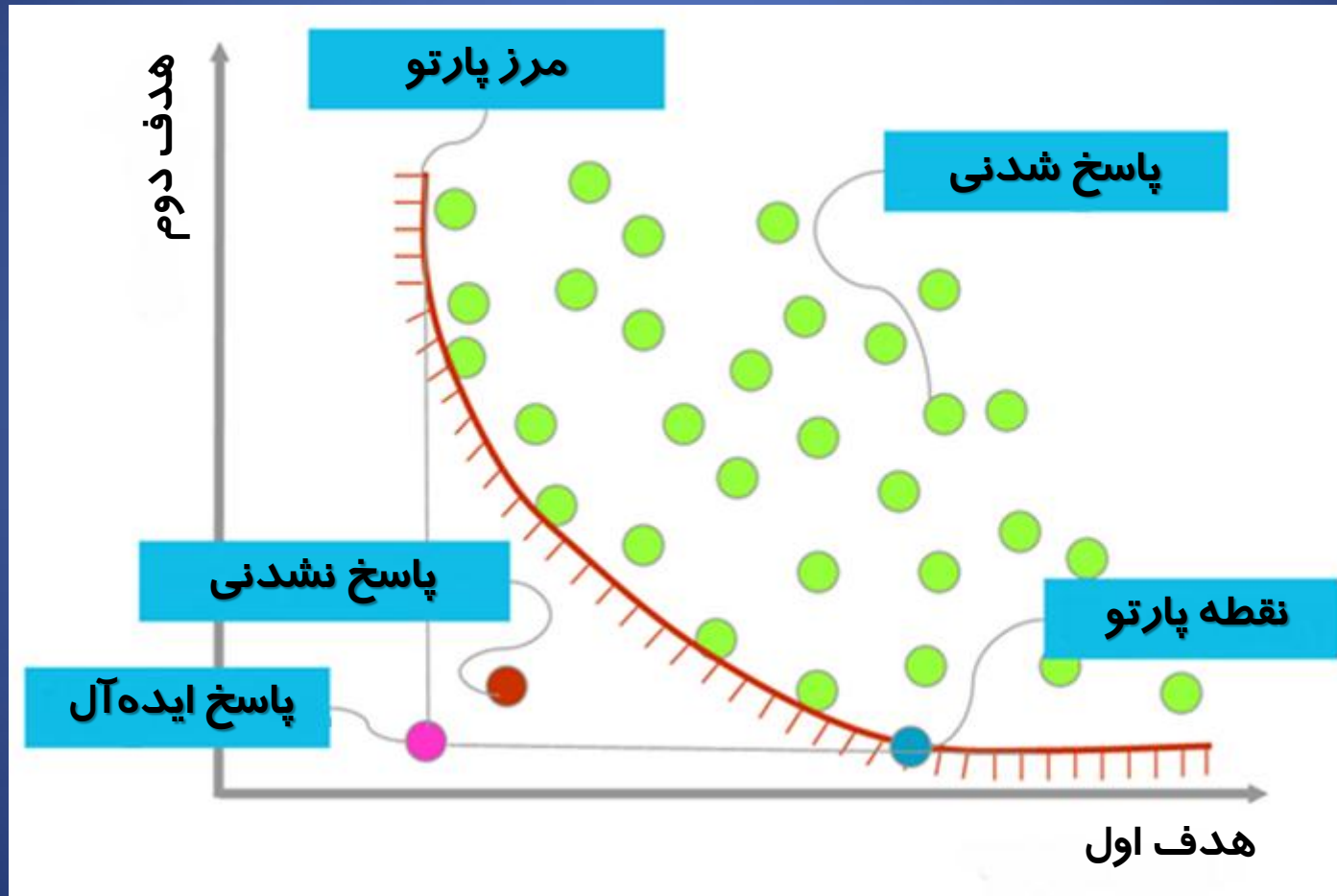
فضای اهداف



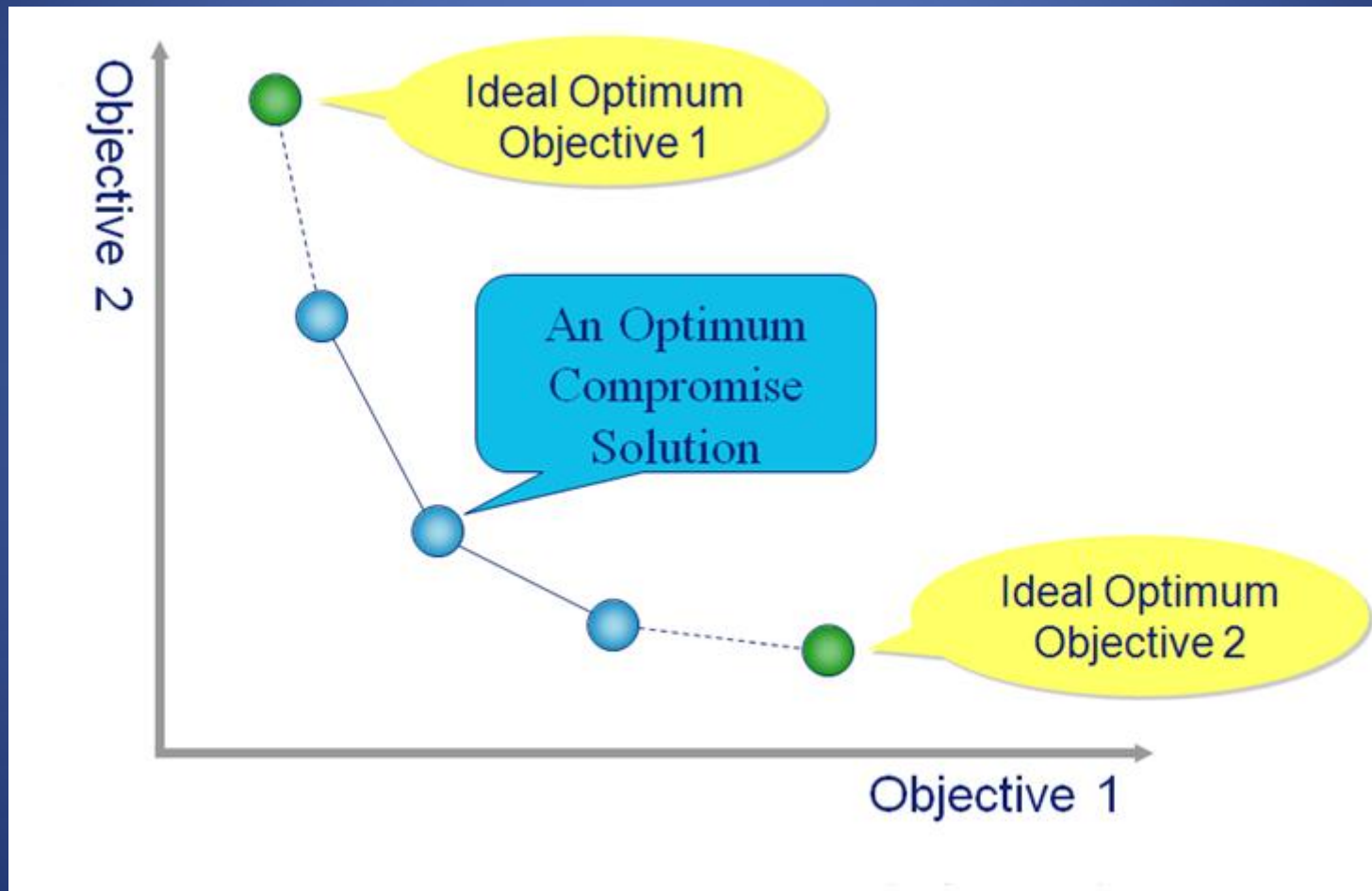
پاسخ‌های غیر مسلط و غیر مسلط ضعیف (Non-dominated solutions)



مرز پارتو (Pareto Front)



آیا پاسخ ایده آل معنا دارد؟



چند تعریف ریاضی مهم

تعریف ۱:

– x^* را پاسخ بهینه کامل گویند اگر و تنها اگر نقطه‌ای مانند $x^* \in X$ وجود داشته باشد بطوریکه $f_i(x^*) \geq f_i(x)$ برای تمامی i ها و $x \in X$ صادق باشد.

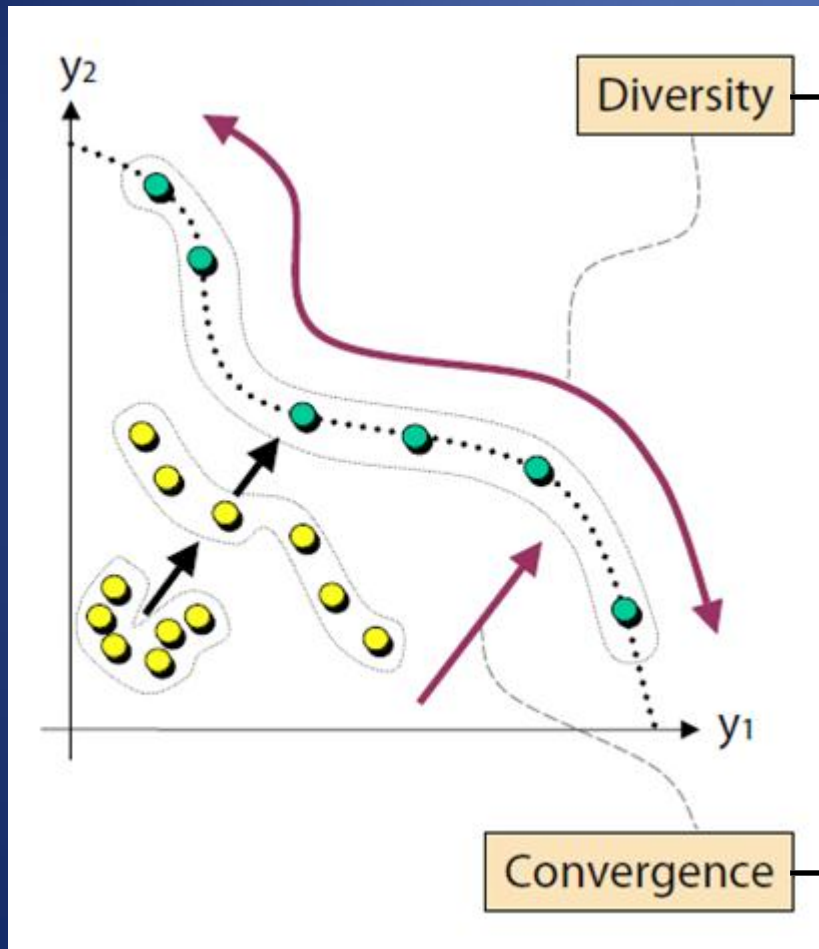
تعریف ۲:

– x^* را پاسخ بهینه پارتو گویند اگر و تنها اگر هیچ نقطه دیگری همچون $x \in X$ وجود نداشته باشد بطوریکه رابطه $f_i(x) \geq f_i(x^*)$ برای تمامی i ها صادق بوده و حداقل برای یک j ، رابطه $f_j(x) \neq f_j(x^*)$ برقرار باشد.

تعریف ۳:

– x^* را پاسخ بهینه ضعیف گویند اگر و تنها اگر هیچ نقطه دیگری همچون $x \in X$ وجود نداشته باشد بطوریکه رابطه $f_i(x) > f_i(x^*)$ برای تمامی i ها برقرار باشد.

نکات مورد توجه



چگونه باید از مرز پارتو یک تخمین دارای تنوع کافی ایجاد کنیم؟

Diversity Estimation

چگونه از گم شدن پاسخ‌های غیر مسلط در طول فرایند الگوریتم جلوگیری کنیم؟

Environmental Selection

چگونه الگوریتم را به سمت تولید مرز پارتو هدایت کنیم؟

Mating Selection

الگوریتم ژنتیک چند هدفه

بنابراین، در طراحی الگوریتم ژنتیک چند هدفه باید به چند موضوع مهم توجه شود:

- Fitness Assignment
- Diversity Estimation
- Environmental Selection
- Mating Selection

بر مبنای روش‌های مختلف پیاده‌سازی این سه، انواع متفاوتی از الگوریتم ژنتیک چند هدفه ابداع شده‌اند:

- MOGA
- SPEA
- SPEA2
- NSGA
- NSGA-II

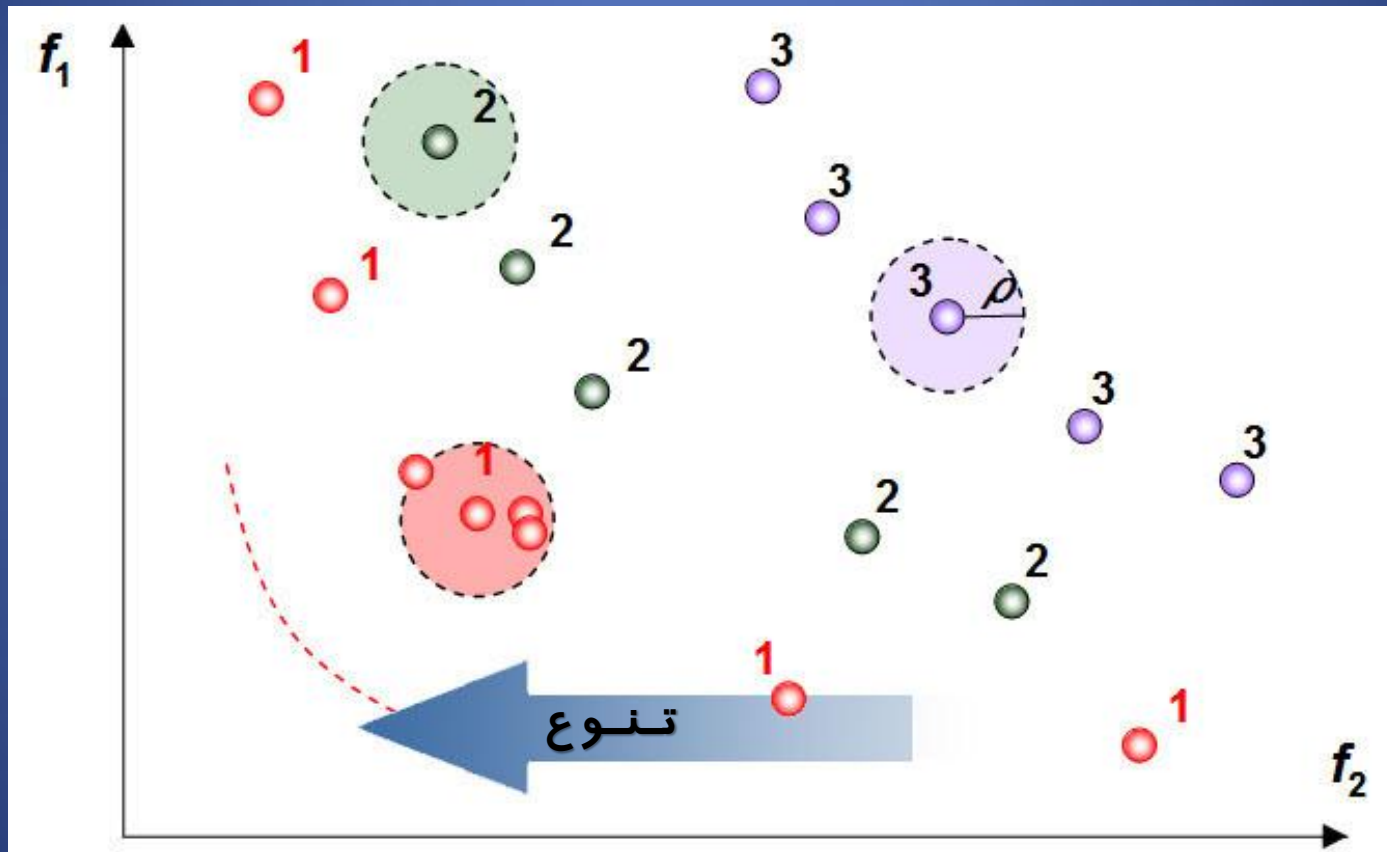
Non-dominated Sorting Genetic Algorithm (NSGA-II)



- این الگوریتم توسط دب در سال ۲۰۰۱ ارائه شده است

- پراوازه و پرکاربردترین الگوریتم ژنتیک چند هدفه در پژوهش‌های صورت گرفته در دنیا (۶۴۸۱ بار استناد بر اساس آمار Google Scholar)

اساس NSGA-II



اجزاء NSGA-II

- NSGA-II را می‌توان به این سه بخش تقسیم کرد:

- روشی سریع برای مرتب‌سازی پاسخ‌های غیر مسلط (تشکیل مرزهای پارتو)

- روشی برای توزیع یکنواخت پاسخ‌ها در مرزهای پارتو

- حلقه اصلی الگوریتم

Fast Non-dominated Sorting Approach

fast-non-dominated-sort(P)

for each $p \in P$

$S_p = \emptyset$

$n_p = 0$

for each $q \in P$

if $(p \prec q)$ then

$S_p = S_p \cup \{q\}$

else if $(q \prec p)$ then

$n_p = n_p + 1$

if $n_p = 0$ then

$p_{\text{rank}} = 1$

$\mathcal{F}_1 = \mathcal{F}_1 \cup \{p\}$

$i = 1$

while $\mathcal{F}_i \neq \emptyset$

$Q = \emptyset$

for each $p \in \mathcal{F}_i$

for each $q \in S_p$

$n_q = n_q - 1$

if $n_q = 0$ then

$q_{\text{rank}} = i + 1$

$Q = Q \cup \{q\}$

$i = i + 1$

$\mathcal{F}_i = Q$

If p dominates q

Add q to the set of solutions dominated by p

Increment the domination counter of p

p belongs to the first front

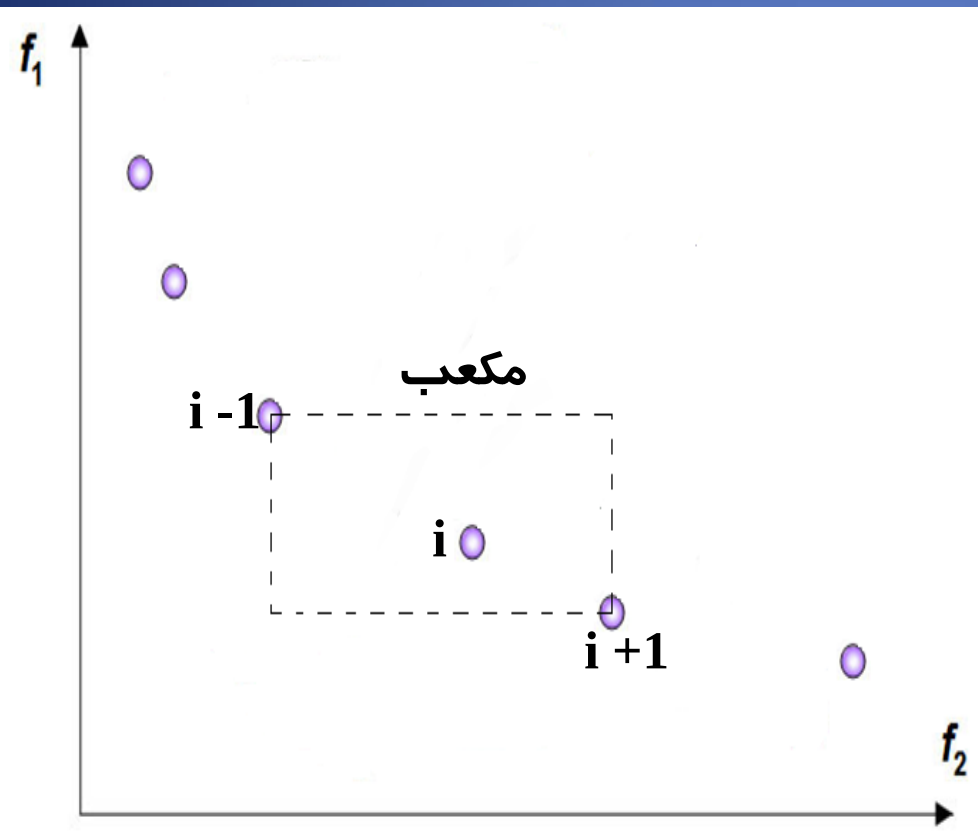
Initialize the front counter

Used to store the members of the next front

q belongs to the next front

Diversity Preservation Approach

- منظور از تنوع در حقیقت ساخت مرز پارتو به صورتی است که پاسخ‌ها به صورت همگن در طول مرز توزیع شده باشند.



- NSGA-II از شاخصی به نام فاصله ازدحام (Crowding Distance) استفاده می‌شود

Crowding Distance Assignment Algorithm

crowding-distance-assignment($\mathcal{I}$)

$l = \mathcal{I} $	number of solutions in $\mathcal{I}$
for each i , set $\mathcal{I}[i]_{\text{distance}} = 0$	initialize distance
for each objective m	
$\mathcal{I} = \text{sort}(\mathcal{I}, m)$	sort using each objective value
$\mathcal{I}[1]_{\text{distance}} = \mathcal{I}[l]_{\text{distance}} = \infty$	so that boundary points are always selected
for $i = 2$ to $(l - 1)$	for all other points
$\mathcal{I}[i]_{\text{distance}} = \mathcal{I}[i]_{\text{distance}} + (\mathcal{I}[i + 1].m - \mathcal{I}[i - 1].m) / (f_m^{\max} - f_m^{\min})$	

نحوه انتخاب

- تا کنون دو اتفاق افتاده است:

1. پاسخ‌ها در قالب چند مرز پارتو تقسیم‌بندی شده‌اند.
2. به هر یک از پاسخ‌ها عددی به نام شاخص تراکم نسبت داده شده است.

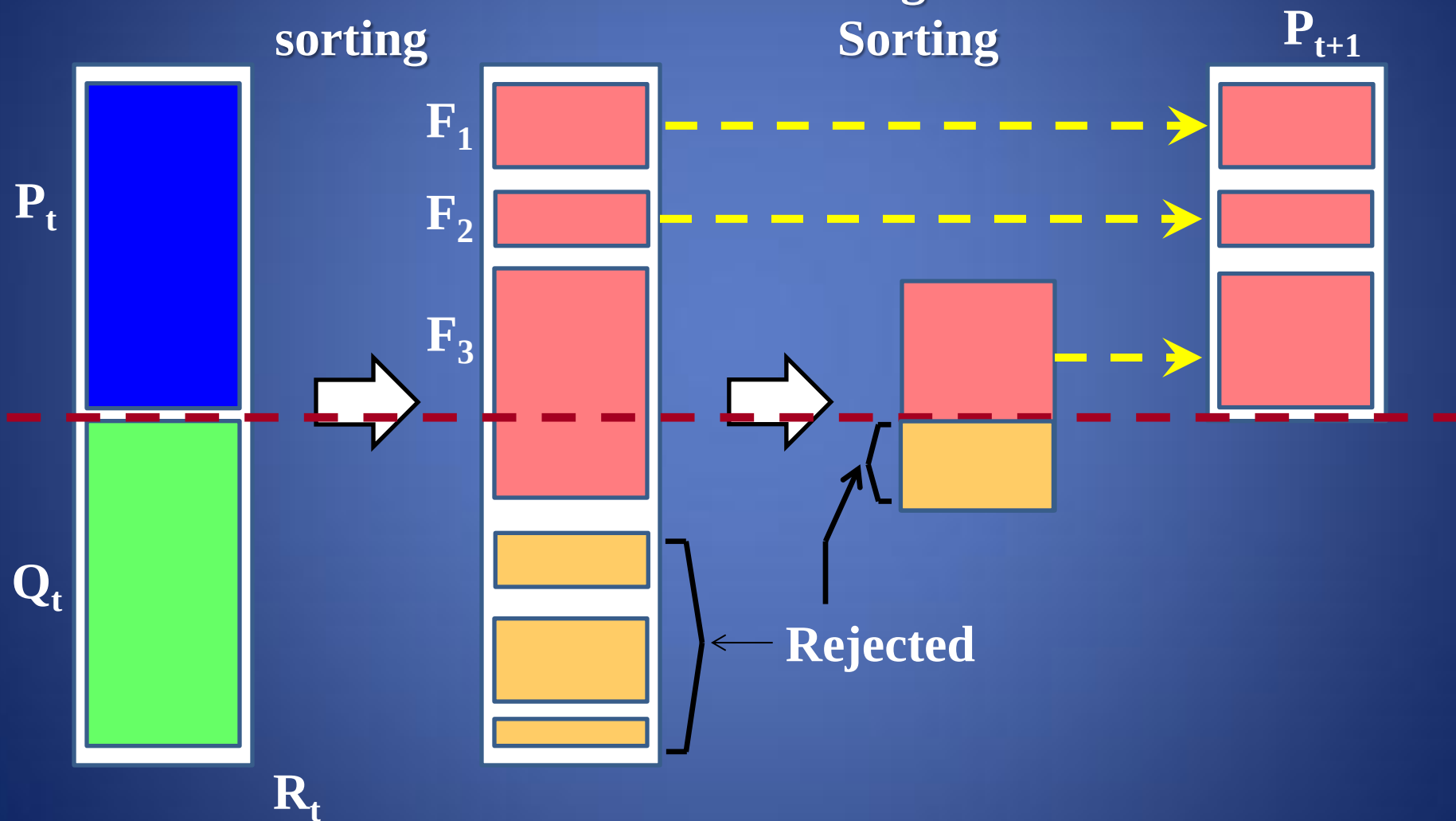
- حال برای انتخاب پاسخ‌ها به این صورت عمل می‌شود:

- میان دو پاسخ که روی دو مرز متفاوت هستند، پاسخی که بر روی مرز با رتبه پایین‌تر (بهتر) هست انتخاب می‌شود.
- میان دو پاسخ بر روی یک مرز، پاسخی که دارای شاخص تراکم کمتری است، انتخاب می‌شود. (چرا؟!)

NSGA-II Procedure

Non-dominated
sorting

Crowding Distance
Sorting



Main Loop

$R_t = P_t \cup Q_t$

$\mathcal{F} = \text{fast-non-dominated-sort}(R_t)$

$P_{t+1} = \emptyset$ and $i = 1$

until $|P_{t+1}| + |\mathcal{F}_i| \leq N$

$\text{crowding-distance-assignment}(\mathcal{F}_i)$

$P_{t+1} = P_{t+1} \cup \mathcal{F}_i$

$i = i + 1$

$\text{Sort}(\mathcal{F}_i, \prec_n)$

$P_{t+1} = P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|)]$

$Q_{t+1} = \text{make-new-pop}(P_{t+1})$

$t = t + 1$

combine parent and offspring population

$\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \dots)$, all nondominated fronts of R_t

until the parent population is filled

calculate crowding-distance in $\mathcal{F}_i$

include i th nondominated front in the parent pop

check the next front for inclusion

sort in descending order using $\prec_n$

choose the first $(N - |P_{t+1}|)$ elements of $\mathcal{F}_i$

use selection, crossover and mutation to create

a new population Q_{t+1}

increment the generation counter

مسأله کوله پشتی چند هدفه

اقلام	۱	۲	۳	۴	۵	۶	۷
سود	۵	۸	۳	۲	۷	۹	۴
وزن	۷	۸	۴	۱۰	۴	۶	۴
قیمت	۱۰	۲۰	۱۸	۲۵	۱۴	۲۲	۲۰

- ظرفیت کوله‌پشتی برابر است با ۲۲ کیلوگرم
- حداکثر بودجه برای خرید ۸۰ واحد
- کوله‌پشتی را چگونه پر کنیم که بیشترین سود و کمترین وزن به دست آید؟

پایان کارگاه اول