**بهینه سازی**

به دنبال پیدا کردن متغیرهای مستقل مطلوب خودمون هستیم.

مسئله[[1]](#footnote-1) : نگاشت کردن متغیرهای مستقل بر متغیرهای وابسته در غالب یک الگوریتم.

* **مثال:**

پیدا کردن ابعاد بهینه پنجره یک اتاق

سوال ۱ : ابعاد بهینه رو بر اساس چه چیزی میخوای؟ تابع هدف!

جواب ۱ : بر اساس نور بهینه

سوال ۲ :‌نور بهینه یعنی چی ؟

جواب ۲ :‌مثلا در یک اقلیم کویری لازمه که دریافت تابش / نور محدود بشه و از طرف دیگه باید فضامون بر اساس فلان استاندارد حداقل میزان نور رو دریافت کنه.

در هر مسئله بهینه سازی متغیرهای هدف و مستقل میتونن متفاوت باشن.

متغیر هدف ( وابسته) [[2]](#footnote-2): پس مقدار نوری که وارد فضا میشه. (‌در بهینه سازی یک هدفه ، یک متغییره)

متغیر مستقل [[3]](#footnote-3): با فرض ثابت ماندن مساحت پنجره، مثلا ارتفاع و عرض پنجره، یا عمق سایه بان یا ... . باید محدوده تغییرات رو برای یافتن جواب بهینه مشخص کنیم. (‌میتونه بیش از یک متغیر باشه)

عملیات نگاشت[[4]](#footnote-4) : فرایند اتصال بین متغیر های مستقل و وابسته

Independent variables

x1, x2 ,…, xn

Dependent variables

y

mapping

مرور : حداقل مقدار تابع رو پیدا کنید.

با دیدن نمودار متوجه میشیم که در زمانی که x = 0 باشه،‌ مقدار f(x) حداقل خواهد بود. به دلیل وجود رابطه ریاضی روشن و مشخص در این مسئله، یافتن جواب کار راحتی خواهد بود.

روش حل ریاضی‌:

در نقطه x = 0 ، شیب خط مماس برابر با صفره. از اینجا میتونیم نقاط اکسترمم رو پیدا کنیم. پس برای پیدا کردن اکسترمم های یک معادله، لازمه که مشتقش رو برابر صفر قرار بدیم تا اکسترمم هاش بدست بیاد.

بهینه سازی در مباحث هوش مصنوعی gradient descent می نامند.یعنی گرادیان کاهشی. با حرکت از هر نقطه ای به اطراف بر اساس شیب خط میتونیم متوجه بشیم که داریم به سمت مینیموم حرکت میکنیم یا ماکسیمم.

در موضوعات معماری، تبدیل هر مسئله به یک رابطه روشن ریاضی کار دشواریه.

مثال : میزان دید از یک پنجره به بیرون رو بر اساس یک معادله ریاضی بنویسید.

X1  = عرض پنجره

X2  = طول پنجره

اگر من بتونم رابطه ریاضی بین متغیرهای مستقل و وابسته رو در قالب یک فرمول ریاضی پیدا کنم، میتونم ازش مشتق بگیرم و به بر اساس گرادیان و شیب خط به سمت اکسترمم ها حرکت کنم و جواب مطلوبمو پیدا کنم.

اما الزاما همیشه نمیشه این رابطه رو پیدا کرد. پس ما تلاش میکنیم از این فرایند یک الگوریتم بسازیم.

توی گرس هاپر ما یک سری اسلایدر داریم که میتونه نقش xها رو برای ما پیدا کنه.

Slider 1

Slider 2

Slider 3

algorithm

y

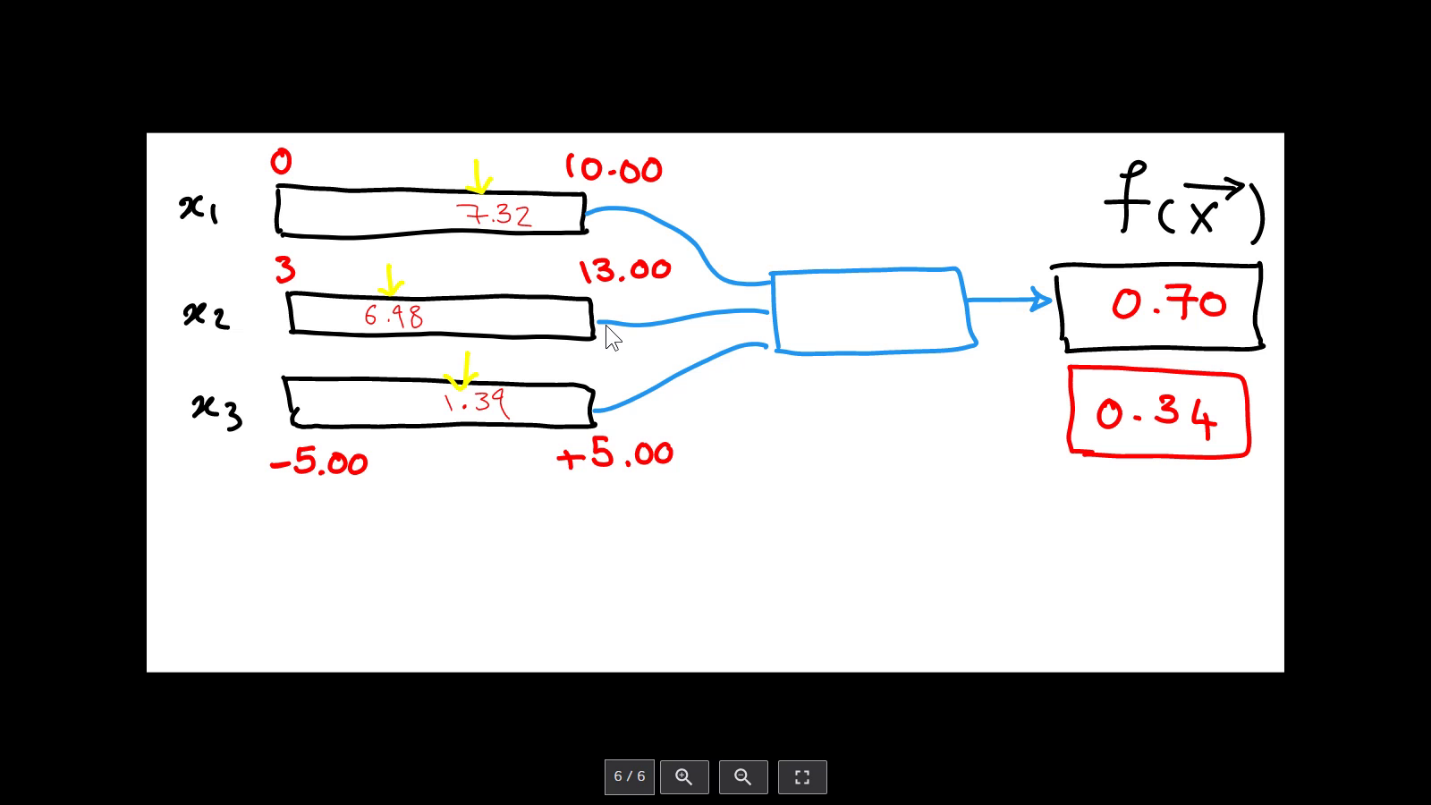
در موضوعات این چنینی به جای حل مسئله به روش ریاضی، از روش های فراابتکاری ، مسئله رو حل میکنیم.

بهینه سازی :‌به دنبال مینیمایز/ماکسیمایز کردن یک تابع هستیم.

تابع نام های دیگری مانند تابع هدف[[5]](#footnote-5) یا تابع هزینه[[6]](#footnote-6) هم دارد. نام های دیگری نیز دارد.[[7]](#footnote-7)[[8]](#footnote-8)

تعیین دامنه متغیرهای مستقل:

برای اسلایدرها باید بازه تعریف کنیم

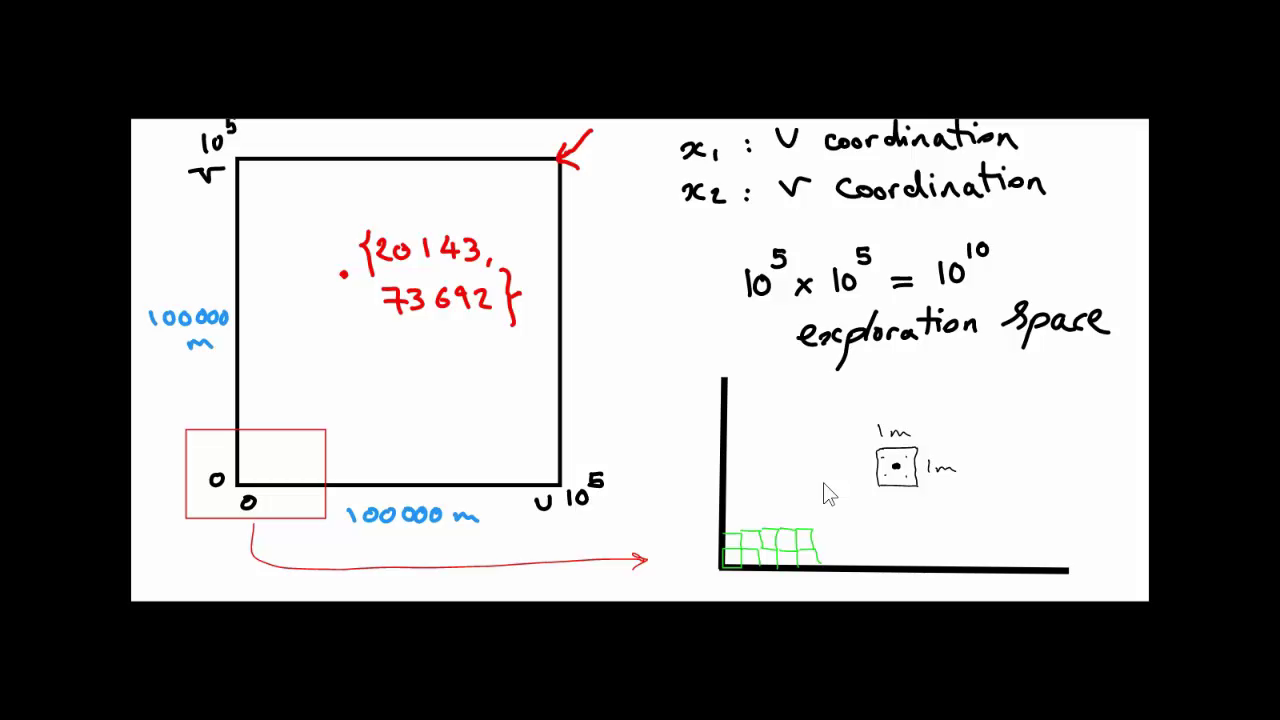


اول باید ببینیم که هر اسلایدری چندتا حالت رو ممکنه به خودش بگیره. برای فهمیدنش لازمه که حداقل، حداکثر و گام تغییرات رو متوجه باشیم.

**الگوریتم ژنتیک**

فرض کنین، میخواهیم نقطه ای که کمترین ارتفاع رو از سطح دریا داره توی یک کویر پیدا کنیم. این کار رو با برداشت انسانی انجام میدیم.

کویر همون فضای جستجو[[9]](#footnote-9)ی ماست.



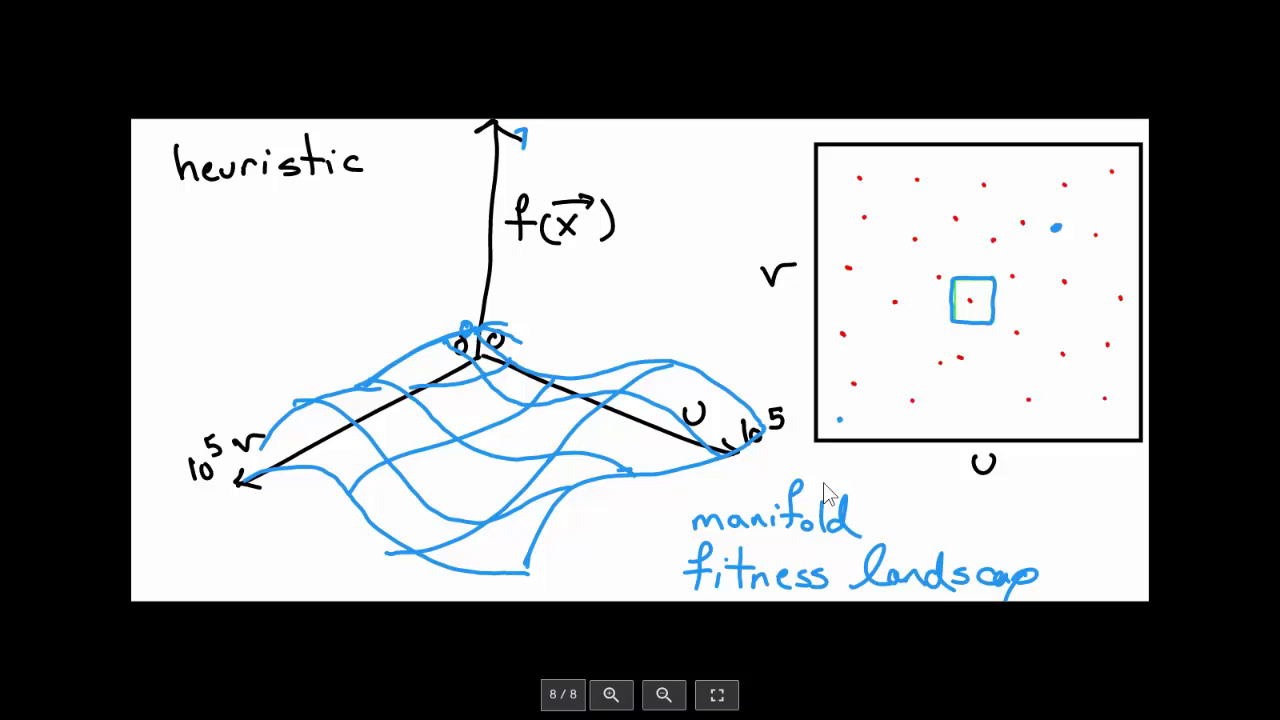
در این مسئله ، وابسته به دقت گام های هر متغییر جواب ها تصاعدی زیاد میشه و ما نمیتونیم همه جواب ها رو محاسبه کنیم و اندازه گیری کنیم. در این موارد، از الگوریتم های هیوریستیک استفاده می‌کنیم. الگوریتم هایی که با اونها به دنبال کشف فضا هستیم.

مثلا ممکنه ما ۵۰ نفر رو ببریم کویر و بگیم رندوم پخش بشن و ارتفاع نقطه ای که توش هستن رو گزارش کنن. ایجنتی که بهترین جواب رو داشت، بقیه یک گام بهش نزدیک شن و باز گزارش کنن و باز... .

سوال اینه که آیا جوابی که بهش میرسیم مینیموم گلوبال هست؟ نه الزاما! ممکنه کلا جواب جای دیگه ای باشه.

این مثال، مفهوم کشف کردنه فضاست. یه تعدادی ایجنت داریم، یه تعدادی هم لوپ داریم. که باهاش یه جوابی رو گزارش میکنیم با تعداد خیلی کمتر که یک منطقی داره ولی الزاما ایده ال نیست.

اگه این مسئله رو ببریم در فضای کارتزینی، خواهیم داشت :



توی مثال کویر، با بردن اطلاعات به فضای کارتزینی، شاید شکل گیری یک fitness landscape یا manifold هستیم. از اونجایی که ما فرم این manifold رو نمیدونیم، نمیشه ترسیمش کرد. اگه میدونستیم که دیگه نیازی به بهینه سازی نبود. به راحتی نقطه مینیموم رو پیدا میکردیم.

اگه توی مسئله ای بیش از دو متغیر مستقل داشتیم، فضای fitness landscape رو نمی‌تونیم نمایش بدیم، اما از نظر ریاضی کاملا وجود داره.

از اونجایی که ما به دنبال کشف فضای fitness هستیم، مسئلمون مربوط به کشف فضاست و در دسته بندی متاهیوریستیک قرار میگیره.

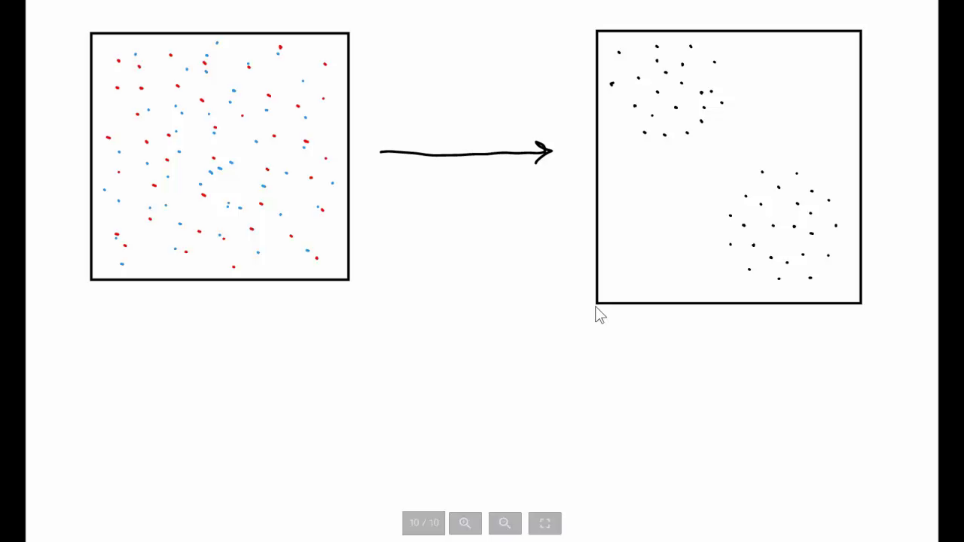
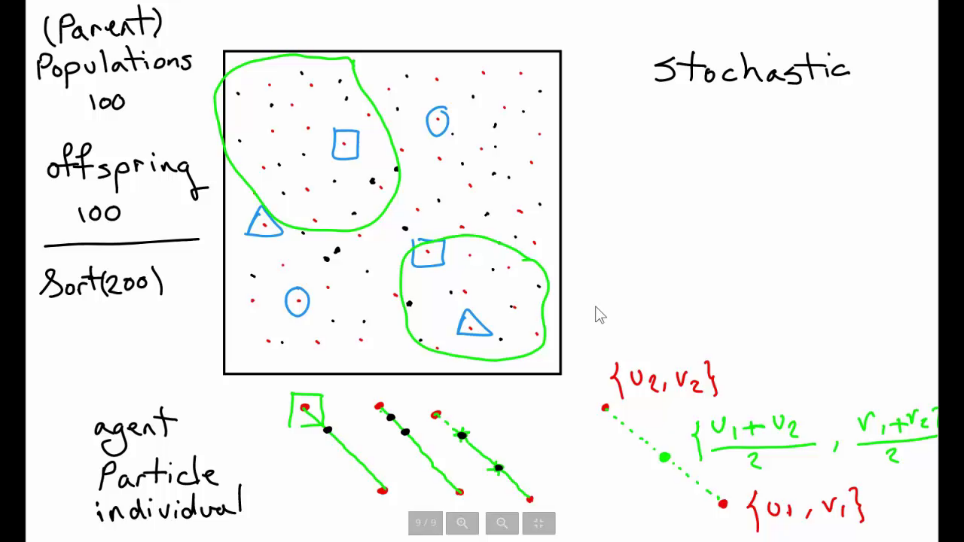
**روش کار ژنتیک**

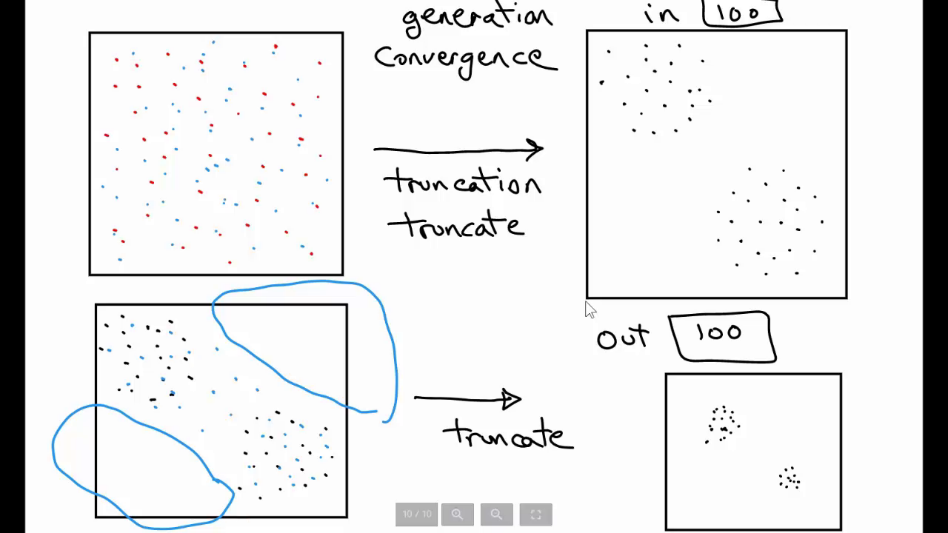
یه فضایی داریم به اسم فضای جستجو

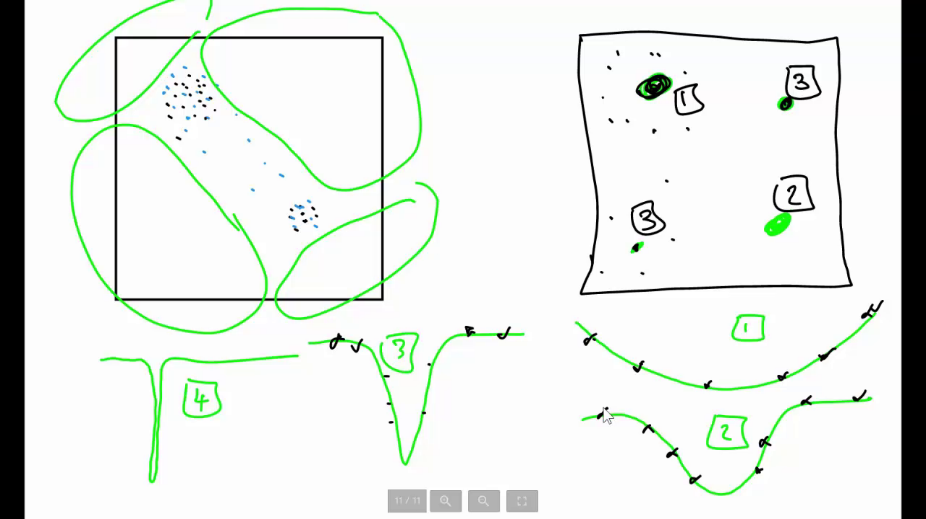
یه تعدادی ایجنت[[10]](#footnote-10) داریم . مثلا صد تا داریم که این ها رو بصورت random uniform پخش میکنیم.

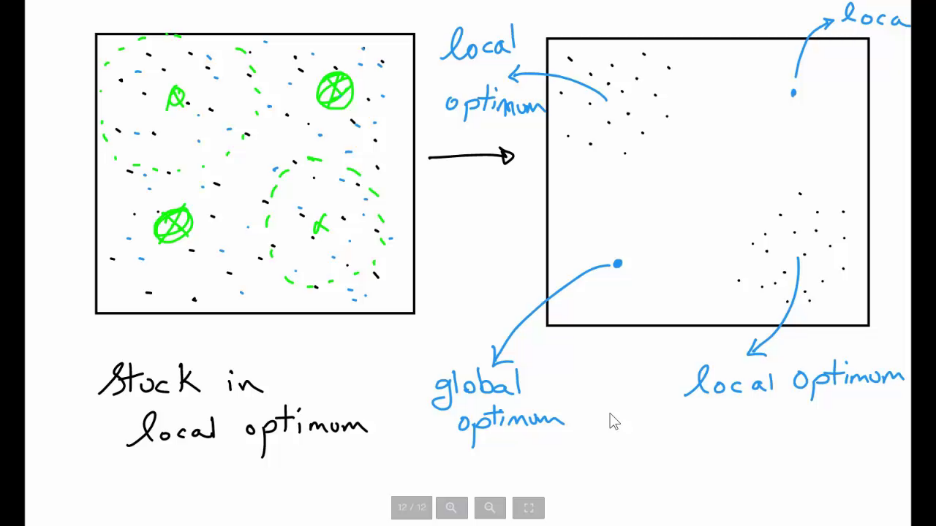
شروع الگوریتم ژنتیک رندوم هست به همین دلیل بهشون میگن Stochastic .یعنی شروع از رندوم ولی یونیفرم.

بعد از اینکه رندوم شروع شد، از هر دو ایجنت، یک کاپل میسازیم که در اینصورت ۵۰ تا کاپل داریم. هدف اینه که فرزند تولید کنن. یعنی نقطه جدید سمپلینگ درست کنن که این نقطه یه جایی بین دوتا والد هست. برای ایجاد فرزندها از روش ها و حالت های مختلفی استفاده میشه کرد. به جمعیت والدها[[11]](#footnote-11) ، فرزندها [[12]](#footnote-12) هم اضافه میشن. این جامعه رو سورت میکنیم و یه تعدادی رو از بهترین ها انتخاب میکنیم و باز فرایند رو انجام رو ادامه میدیم. به این فرایند[[13]](#footnote-13) (حذف)‌گفته میشه.









برای اینکه از local optimum ها فرار کنیم، یا احتمال پیدا کردن global optimumها بیشتر بشه، باید از جهش استفاده کنیم.

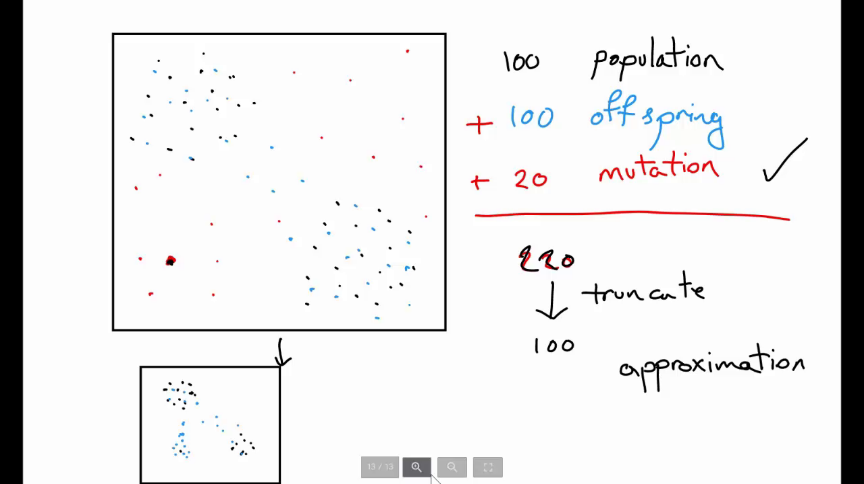
**جهش[[14]](#footnote-14) :**

اتفاقی خارج از قاعده تکرار شونده موجود

جهش میتونه خودش قاعده ای مختلفی داشته باشه.

1. یکی از بُعدهای یکی از والدها رو مثلا عوض کنیم
2. کلا رندوم باشه

با هدف رسیدن به global optimum . با همه این توضیحات، همچنان امکان نرسیدن به global optimum وجود دارد. به همین دلیل الگوریتم ژنتیک، یک الگوریتم approximate است.



ما نیاز به شرط خروج[[15]](#footnote-15) داریم :

1. شرط تعداد نسل :‌به تعداد مشخصی نسل پیش بره و بعدش تموم شه.
2. شرط عدم بهبود پاسخ :‌یه وقت فلان نسل گذشت و جواب بهتر نشد تموم شه.

**Exploration VS exploitation**

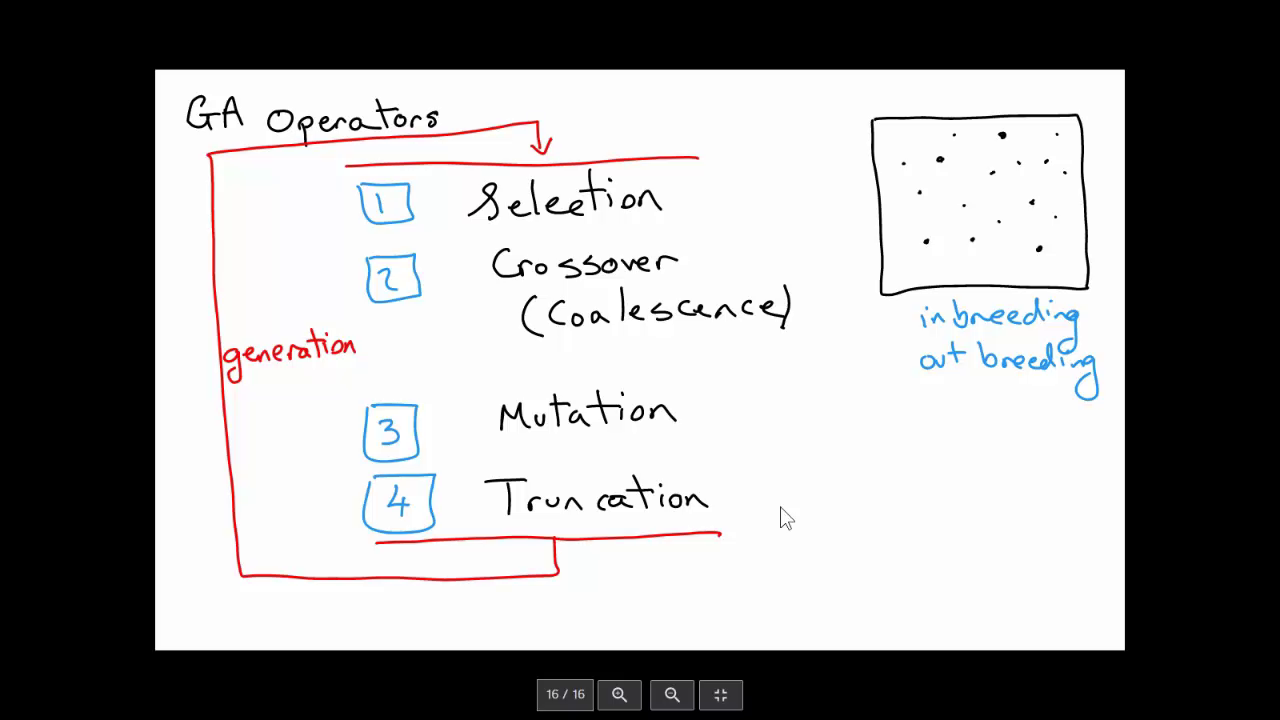
Exploration:‌دوست داریم دنبال جواب های بیشتری باشیم که نکنه چیزی رو از دست داده باشیم.

Exploitation:‌دوست داریم تمرکزمون رو بذاریم روی یک ناحیه و با دقت اونجا رو بگردیم.

در نسل های اول بیشتر explore کن و در نسل های پایانی بیشتر exploit کن.

**جمع بندی فرایند الگوریتم ژنتیک**

1. الگوریتم ژنتیک مستقل از خود مسئله کار میکنه و به همین دلیل بهش میگیم فرا ابتکاری [[16]](#footnote-16)
2. الگوریتم یک فرایند تکرار شونده رو طی می کنه پس یک الگوریتم تکاملی[[17]](#footnote-17) هست
3. الگوریتم شروع رندوم داره[[18]](#footnote-18)
4. برای موضوعات پیوسته بهتره.
5. اوپریتورها:
   1. انتخاب[[19]](#footnote-19) : کاپل ساختن ها با چه قاعده ای انجام بشه. مثلا بهترین جواب ها با هم و بدترین جواب ها با هم یا برعکس[[20]](#footnote-20)
   2. ترکیب[[21]](#footnote-21) :
   3. جهش :
   4. ترانکیت[[22]](#footnote-22) :



1. problem [↑](#footnote-ref-1)
2. Dependent variables [↑](#footnote-ref-2)
3. Independent variables / features / attributes / dimensions [↑](#footnote-ref-3)
4. mapping [↑](#footnote-ref-4)
5. Objective function میتونه مینیمایز باشه یا ماکزیمایز [↑](#footnote-ref-5)
6. Cost function همیشه دنبال کاهش هستیم [↑](#footnote-ref-6)
7. Fit function [↑](#footnote-ref-7)
8. Loss function [↑](#footnote-ref-8)
9. Exploration / search space [↑](#footnote-ref-9)
10. Agent / particle / individual / population [↑](#footnote-ref-10)
11. parents [↑](#footnote-ref-11)
12. offspring [↑](#footnote-ref-12)
13. Truancation [↑](#footnote-ref-13)
14. mutation [↑](#footnote-ref-14)
15. Termination condition [↑](#footnote-ref-15)
16. Meta-heuristic [↑](#footnote-ref-16)
17. Evolutionary [↑](#footnote-ref-17)
18. Stochastic [↑](#footnote-ref-18)
19. selection [↑](#footnote-ref-19)
20. Inbreeding / outbreeding [↑](#footnote-ref-20)
21. Crossover / coalescence [↑](#footnote-ref-21)
22. Truncate [↑](#footnote-ref-22)