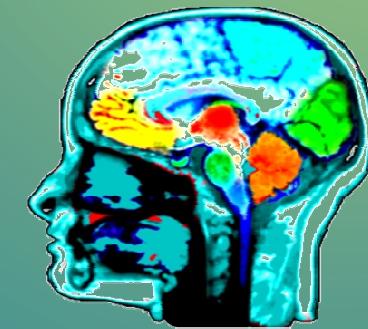




[These slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley.]

# Introduction To Artificial Intelligence

Isfahan University of Technology (IUT)  
1402



Local Search

Dr. Hamidreza Hakim  
[hakim@iut.ac.ir](mailto:hakim@iut.ac.ir)

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ

# Generated and Test

---

- **Algorithm**

1. Generate a (potential goal) state:
  - Particular point in the problem space, or
  - A path from a start state
2. Test if it is a goal state
  - Stop if positive
  - go to step 1 otherwise

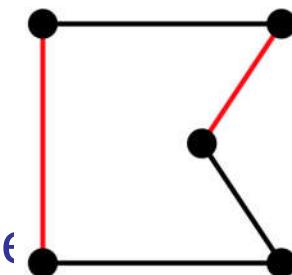
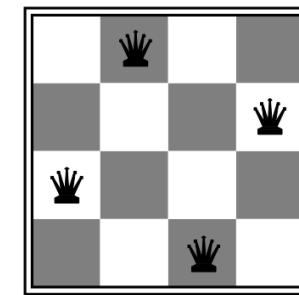
- **Systematic or Heuristic?**

- It depends on “Generate”

اتفاقی که تا قبل از این جلسه می‌افتد: ما می‌اوهدیم استیت رو می‌ساختیم و هر گام یک استیتی برآمون ایجاد می‌کرد و بعد بر یک اساسی یکدوم از اون استیت‌ها رو انتخاب می‌کردیم و به اون استیت جدید که می‌رسیدیم می‌دیدیم اون استیت هدف است یا نه

# Local search algorithms

- In many optimization problems,  
**path** is irrelevant;  
the goal state **is** the **solution**
- Then state space = set of “complete” configurations;  
find **configuration satisfying constraints**,
- e.g., n-queens problem; or,
- find **optimal configuration**, e.g., travelling salesperson problem



اما

توی لوکال سرچ:  
چندین یال مطرح شده

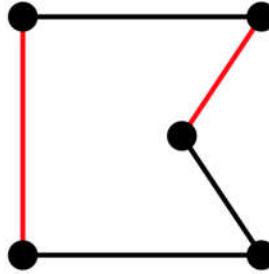
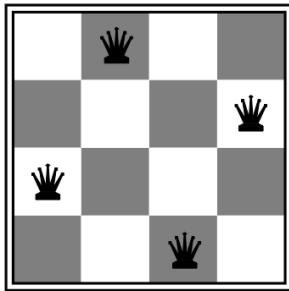
یکسری مسائلی وجود داره که میخوایم به اون استیت هدف برسیم ولی مسیر رسیدن به اون هدف  
برامون مهم نیست ینی ما دیگه نمی خوایم مسیریابی بکنیم  
پس لوکال سرچ بر این مبنای استند که مسیر رو ما دیگه نمیخوایم نگهش داریم و فقط دنبال اون حالت  
نهایی هستیم

اون حالت جواب یا حالت نهایی یک شرایطی داره  
مثال:

مسئله  $n$  وزیر یا مسئله فروشنده دوره گرد

پس ما میخوایم به اون حالت نهایی برسیم با یکسری تکنیک هایی

# Local search algorithms



- In such cases, can use ***iterative improvement*** algorithms:  
keep a single “current” state,  
try to improve it
- Constant space, suitable for online as well as offline search

رویه به چه صورت است:

ما سعی میکنیم توی استیت ساختن اینجا هوشمندی به خرج بدیم ما توی یک حالتی هستیم و میخوایم حالت بعدی رو جوری انتخاب بکنیم که که حالت بعدی بیوقته همون حالت هدف پس از یک حالتی بریم به یک حالت دیگه ای که یا حالت بعدی حالت نهایی باشه یا نزدیک به حالت نهایی باشه ینی ما یک **current** استیت داریم و میخوایم هی اینو بهتر بکنیم

# Why Local Search?

---

- Low Memory
  - Find acceptable solution (not perfect).
- 
- Optimization Problem (objective function)

روش لوکال سرچ سرعتشون خیلی بیشتر است و از لحاظ مموری هم خیلی بهترن

یک نکته دیگه ای که راجع به لوکال سرچ وجود داره اینه که ما دنبال جواب ایده ال نیستیم و پیدا کردن جواب ایده ال جلوتر می بینیم که خیلی سخت است

یک اصطلاحی داشتیم تحت عنوان **Heuristic** فانکشن ما به هر استیتی یک **Heuristic** فانکشن نسبت می دادیم که بهمون می گفت فاصله این حالت تا استیت هدف چقدر است معادل این توى مسائل بهینه سازی بهش میگن **objective** فانکشن

# Hill Climbing

---

- Simple, general idea:
  - Start wherever
  - Repeat: move to the best neighboring state
  - If no neighbors better than current, quit



ما میخوایم این 8 تا وزیر رو بچینیم و یک وضعیت الان داریم و میخوایم یک وضعیت بهتری پیدا بکنیم

Hill Climbing یا تپه نوردی:

یک حالتی داریم که این حالت یک ارزشی دارد و ما میخوایم از این حالت بریم به یک حالت دیگه --> از یه جایی شروع می کنیم و می ریم به استیت های بعدی و فقط استیت های رو انتخاب میکنیم که نسبت به استیت های قبلی بهترن --> این رویه رو که ادامه بدیم یه جایی دیگه هیچ گزینه ای نداریم ینی دیگه بهتر از این نمیشه و اینجا دیگه متوقف می شیم

# Hill Climbing

---

- Simple Hill Climbing
  - expand the current node
  - evaluate its children one by one (using the heuristic evaluation function)
  - choose the FIRST node with a better value
  
- Steepest Ascend Hill Climbing
  - expand the current node
  - Evaluate all its children (by the heuristic evaluation function)
  - choose the BEST node with the best value(greedy algorithms)

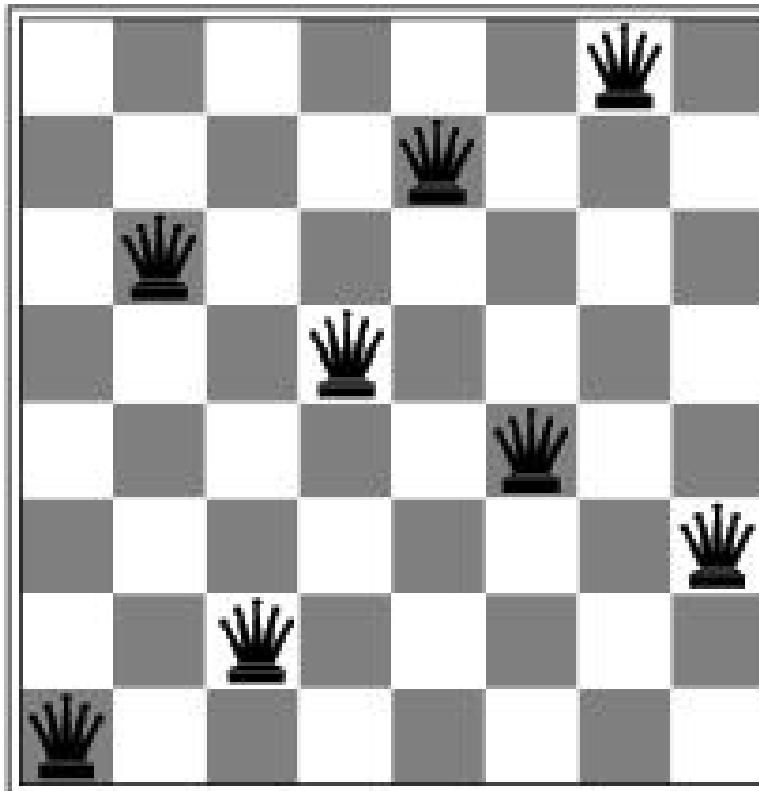
## :Simple Hill Climbing

اول کار یکیشو بردار و به اون حالت برو و بیا این **Heuristic** فانکشنش رو حساب بکن توى اون فضای حالت و همین رویه رو برو جلو و شرط توقف اینجا اینه که به استیت هدف برسیم

## :Steepest Ascend Hill Climbing

یک روش بهتر از قبلی وجود داره به نام **Steepest Ascend Hill Climbing** همینجوری که داریم نگاه می کنیم همه این حالت هایی که داره رو سورت بکنیم و گردیدی انتخاب بکنیم و شرط توقف اینجا اینه که به یک وضعیتی برسیم که دیگه نتونیم حالتی تولید بکنیم

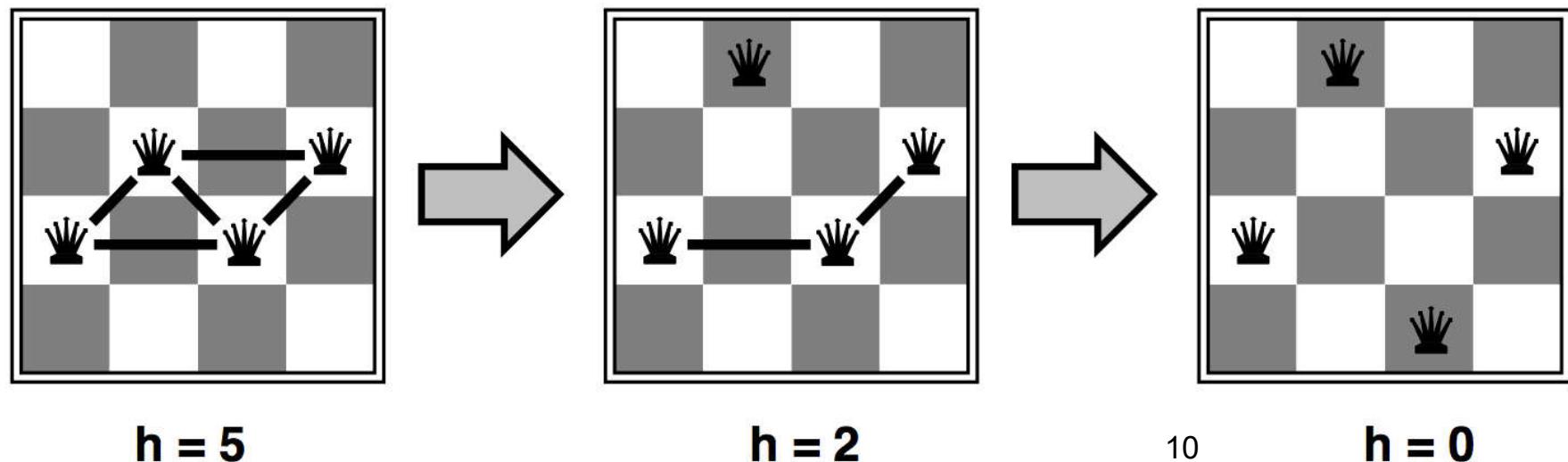
# $n$ -queens problem



---

# Heuristic for $n$ -queens problem

- Goal:  $n$  queens on board with no **conflicts**, i.e., no queen attacking another
- States:  $n$  queens on board, one per column
- Actions: move a queen in its column
- Heuristic value function: number of conflicts



10

---

# Hill-climbing algorithm

---

```
function HILL-CLIMBING(problem) returns a state
    current ← make-node(problem.initial-state)
    loop do
        neighbor ← a highest-valued successor of current
        if neighbor.value ≤ current.value then
            return current.state
        current ← neighbor
```

*“Like climbing Everest in thick fog with amnesia”*

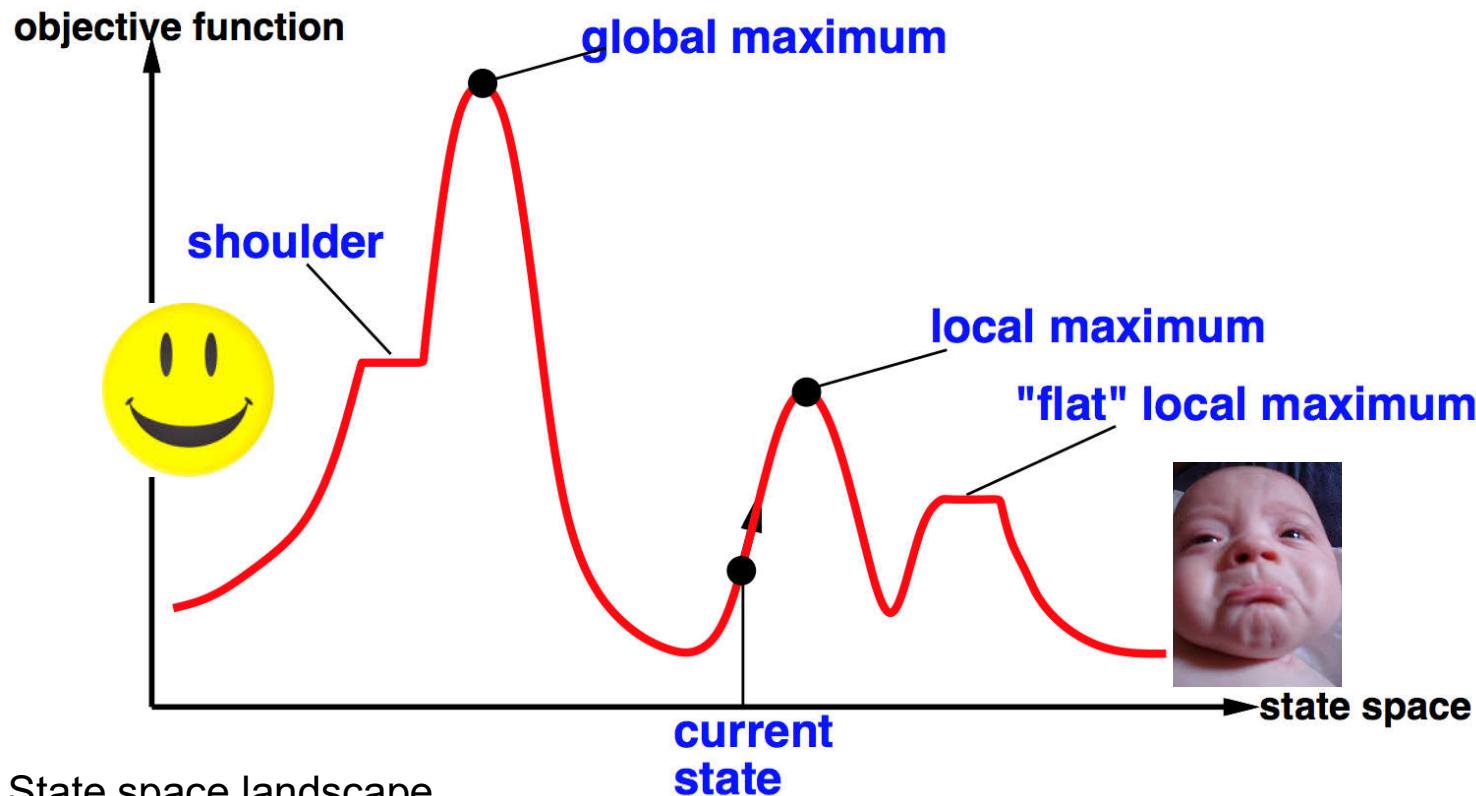
Hill-climbing روی مسئله  $n$  وزیر:

هر کدام از این وزیر ها اگر بخوان حرکت بکن چه بلایی سر Heuristic فانکشن هاشون میاد و اونی که از همه کمتر است رو انتخاب بکنیم

رویه که Hill-climbing داره اینه که:

ما هی نود می سازیم و به هر نودی که رسیدیم یک چرخه ای رو داریم که بیا از این نود یک همسایه ای رو پیدا بکن و به یک حالتی برسیم و همسایه رو گذاشته اون کسی که بهترین همسایه است اونو بذار مبنا و دوباره اون همسایه رو بذار current و اینقدر این رویه رو انجام بده تا دیگه هیچ همسایه ای نداشته باشیم که نسبت به خودت بهتر باشه و این مسیر رو ما داریم طی میکنیم

# Global and local maxima



State space landscape

Objective function  $\sim$ =heuristic function

Optimal solution vs locally optimal solution

Local search may not guarantee the absolute optimal solution

## Random restarts

- find global optimum
- duh

## Random sideways moves

- Escape from shoulders
- Loop forever on flat local maxima

توی این مسئله اگر ما بیایم هر حالتی رو با یک مقداری مپ بکنیم ینی همه حالت های مسئله رو بذارم روی محور افقی و میزان جذابیت هر حالت رو بذاریم روی محور عمودی هر حالتی برآم یک موقعیتی توی این فضا پیدا میکنه که به این میگن Steepest Ascend که همه حالت ها رو و میزان محدودیتشون رو مشخص میکنه

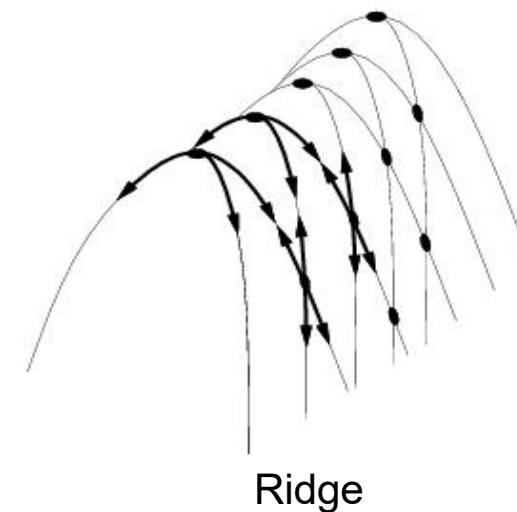
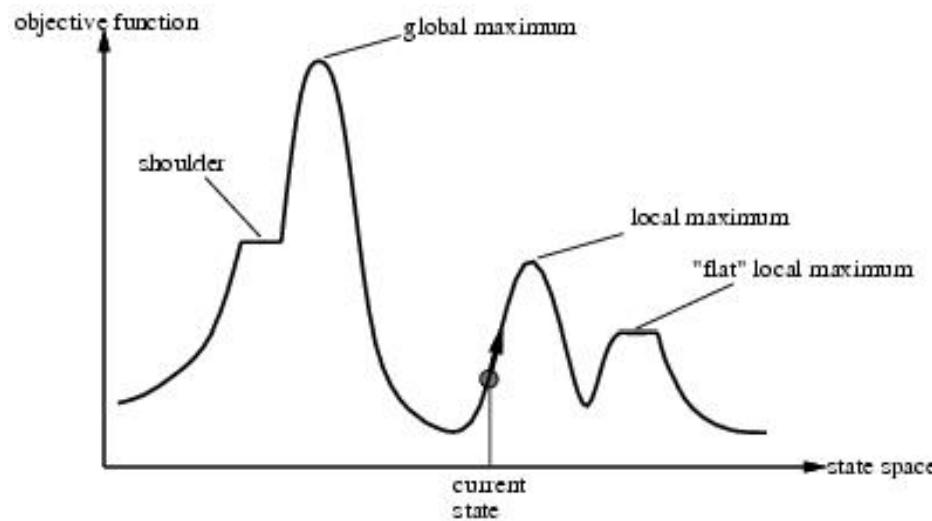
چندتا پیشنهاد داریم:

1- دوباره از اول شروع کنیم ینی وقتی به یک ماکزیمم رسانیدیم دوباره از اول شروع کنیم و دوباره حرکت بکنیم به این امید اینکه در تلاش جدید می تونیم به یک بهینه بهتری بررسیم

2- اگر ما موقعیتمون رو توی این فضا تصادفی برداریم این بهتر میشه ینی بحث رندوم بودن رو اعمال بکنیم ینی فقط دنبال این نباشیم که heuristic بهتر رو انتخاب بکنیم یکم به رندوم هم توجه بکنیم امیدوارم باشیم که اگر توی این تپه گیر کردیم برایم به تپه اصلی بررسیم

یک پیشنهاد دیگه هم وجود داره: وقتی به یک حالت ماکزیمم رسانیدیم از همون جا دوباره بپریم یه جای دیگه ینی حالتمن رو تصادفی عوض بکنیم

# Drawbacks



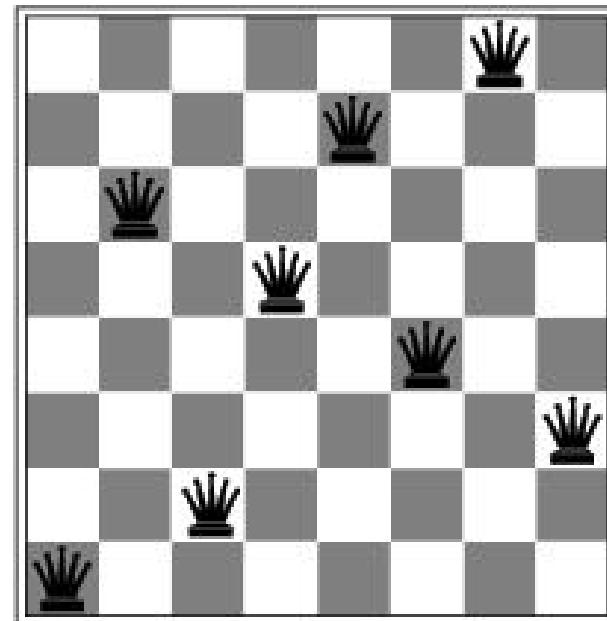
- Ridge = **sequence of local maxima** difficult for greedy algorithms to navigate
- Plateau = an area of the state space where the evaluation function is **flat**.

یک مسئله ای دیگه که وجود داره و کار رو برآمون سخت میکنه و بعضی از مسائل رو واقعا حل نشدنی می کنه اینه که تعداد local maxima هاش خیلی زیاد باشه

اگر ما مسئله ای داشته باشیم که مرتب هر حالتیش رو دنبال میکنیم با یک ذره تغییر دادن به local گیر میکنه این مسائل رو به این راحتی نمیشه حلش کرد

# Hill-climbing example

18	12	14	13	13	12	14	14
14	16	13	15	12	14	12	16
14	12	18	13	15	12	14	14
15	14	14	14	14	13	16	13
14	14	17	15	15	14	16	16
17	15	16	18	15	15	15	15
18	14	15	15	15	14	15	16
14	14	13	17	12	14	12	18



a) Shows a state of  $h=17$  and the  $h$ -value for each possible successor.

Best  $h$  is 12

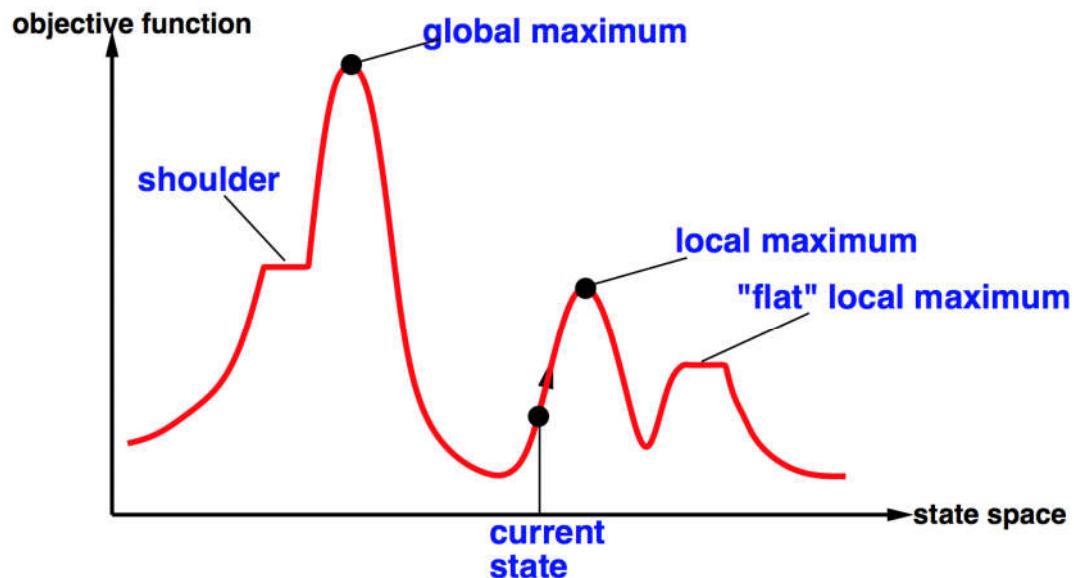
b) A local minimum in the 8-queens state space ( $h=1$ ).

---

# Hill-climbing variations

## ■ Stochastic hill-climbing

- Random selection among the uphill moves.
- The selection probability can vary with the steepness of the uphill move.
- Low rate converge



چندتا ایده که الگوریتم Hill-climbing رو بهتر میکنه:

### -1 Stochastic hill-climbing ینی تپه نورده تصادفی:

این الگوریتم میگه تصادفی انتخاب بکن یکی از اینارو تصادفی انتخاب بکن و در جهت اون حرکت بکن --> این تصادفی که ما داره اعمال میکنیم ینی گریدی دیگه عمل نمیکنیم باعث میشه که ما از لحاظ احتمالاتی به جواب بهینه برسیم ولی خیلی دیر کار همگرا میشه توی این روش یکم اون لوكال ماکزیمم ها رو مدیریت میکنه

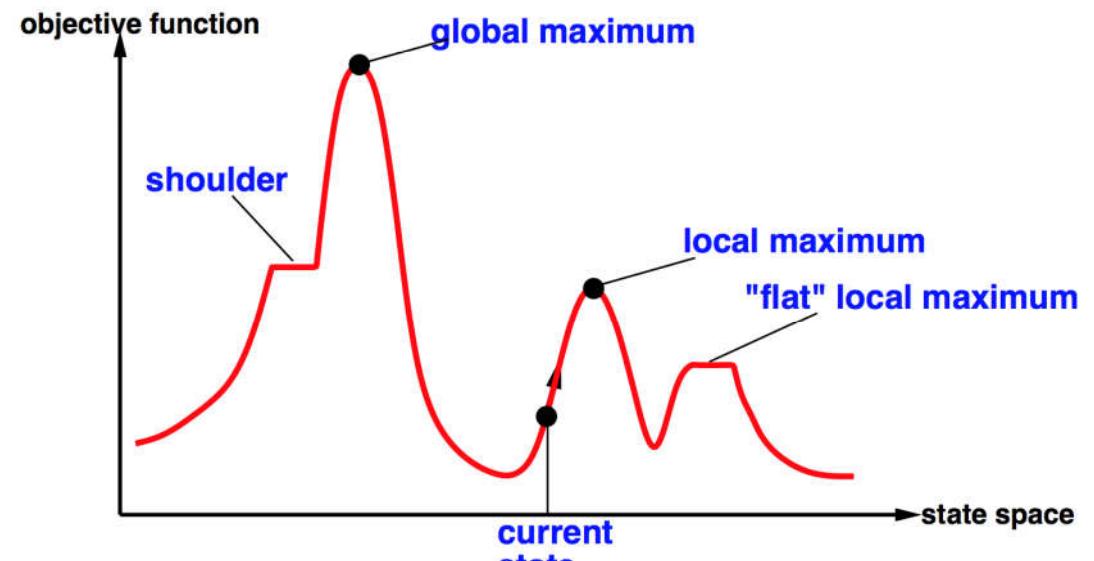
# Hill-climbing variations

- First-choice hill-climbing

- Stochastic hill climbing by generating successors randomly until a better one is found.

(First generate then check)

High speed in 10000 children state



## :First-choice hill-climbing

توی روشن قبلی ما می اومدیم تصادفی انتخاب میکردیم بین اون هایی که خوب هستن و اگر اون حالت، حالت خوبی بود همون رو می رفتیم ولی توی اینجا اول یک حالتی رو بین اون حالت هایی که وجود دارن که هر کدامشون به نوعی اشاره می کنن که استیت بهتری هستن اونی رو انتخاب می کنیم که ماکزیم تر است

توی اونی رو انتخاب میکنیم که از همه بهتر است  
توی Stochastic یکسری فرزند برآش می سازیم و هیوریستیک همشون رو حساب میکنیم و یکیش رو به تصادف انتخاب میکنیم و در جهت اون حرکت میکنیم  
توی First-choice ما دیگه نماییم همه بچه ها رو حساب بکنیم بلکه یک حالت تصادفی تولید میکنیم و بعد هیوریستیکش رو حساب میکنیم اگر بهتر بود که میگیم بله و بعد همون رو ادامه می دیم و اگر بهتر نبود از همون حالت پدر باید دوباره یک حالتی تولید میکنیم و این روش وقتی خوبه که ما بچه های زیادی داریم توی حالت ها First-choice

# Hill-climbing variations

## ■ Random-restart hill-climbing

- A series of Hill Climbing searches from randomly generated initial states
- Try try try!(probability Complete)

$p$ : probability of success in each hill-climbing search

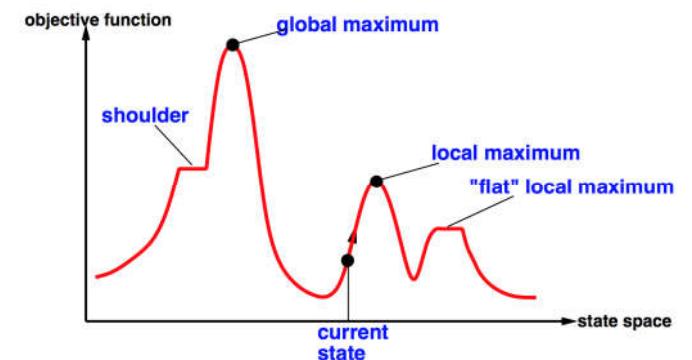
- Expected no of restarts =  $1/p$  (Geometric Dist)
- Expected no of steps =  $(1/p - 1) \times nf + ns$

$nf$ : average number of steps in a failure

$ns$ : average number of steps for the success

Result on 8-queens:

- $p \approx 0.14 \Rightarrow 1/p \approx 7$  iterations
- $(7 - 1) \times 3 + 4 \approx 22$  steps
- For  $3 \times 10^6$  queens needs less than 1 minute
- Using also sideways moves:  $p \approx 0.94 \Rightarrow 1/p \approx 1.06$  iterations



## :Random-restart hill-climbing

ما تپه نورده رو انجام بدیم و به هر جوابی که رسیدیم دوباره بباییم تکرارش بکنیم

چند بار باید این تکرار رو انجام بدیم؟ اگر احتمال رسیدن به جواب  $p$  باشه چند بار باید تکرار بکنیم که به جواب بررسیم؟ ما یک ازماишی رو داریم انجام میدیم و احتمال اینکه موفق بشه  $p$  و احتمال اینکه موفق نشه یک منهای  $p$  و خود این ازمايشه از یک متغیر تصادفی پیروی میکنه که یک تابع توزیع خواهد داشت که طبق این تابع توزیع وقتی که میخوایم بگیم این ازمايش رو چند بار تکرار بکنیم که به شرایط موقیتش بررسیم همین اتفاق می افته یک میانگینی داره که میانگینش مشخصه که وقتی که خواستیم میانگین این رو ببینیم یک فرمولی داره  $p/1$  بار این رو باید تکرار بکنیم که به طور متوسط به اون موقیت بررسیم

توی مسئله 8 وزیر اومدن این رو حساب کردن که وقتی که با روش Random-restart hill-climbing داریم انجام میدیم احتمال اینکه توی یکبار دفعه اول به طور متوسط 0.14 حالت موقیت داریم که به اون حالت ایده ال بررسیم پس طبق این توضیحات ما باید 7 بار این ازمايش رو تکرار بکنیم که 6 بار به شکست بخوریم و بار 7 ام موفق بشیم با 7 بار تکرار ما میتونیم به جواب بهینه بررسیم

$nf$  : یک تعداد استپ رو طی کردیم

$ns$  : خود اون حالتی هم که موفق شده اینه

مثالا برای مسئله 8 وزیر دیدن که توی 22 تا گام داره حل میشه و به جواب می رسه  
نکته: اگر ما بباییم به این الگوریتم Random-restart hill-climbing اجازه حرکت های جانبی هم بدم یعنی اگر توی یک وضعیت گیر کرد همه بچه هاش مقدار های مشابه دارند یعنی اجازه بدم حرکت های اینطوری هم داشته باشه و دوباره نگیم از اول شروع بکن --> حتی مقدار این احتماله بهتر میشه و iterations توسعه پیدا میکنه

# Effect of land-scape shape on hill climbing

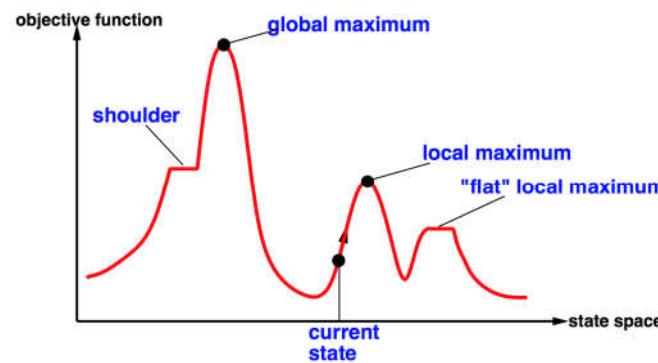
- Shape of state-space land-scape is important:

Few local max and platea: random-restart is quick

Real problems land-scape is usually **unknown** a priori

NP-Hard problems typically have an **exponential number of local maxima**

Reasonable solution can be obtained after a small no of restarts



NP-Hard

اگر توی اون الگوریتم ما تعداد لوکال ماکزیمم هاش کم باشه و تعداد این فلات هاش محدود باشه این الگوریتم، الگوریتم خوبی است ولی توی مسائل واقعی ما نمی دونیم چندتا قله وجود داره در این حالت میگن مسئله جوچه تیغی اتفاق افتاده یعنی توی هر تپه ای که می ریم کنار هم یک تپه ای وجود داره یا توی همون تپه هم تعدادی تپه وجود داره با این حال برای پیدا کردن جواب نسبتا خوب الگوریتم های خوبی هستن

# Hill-climbing variations

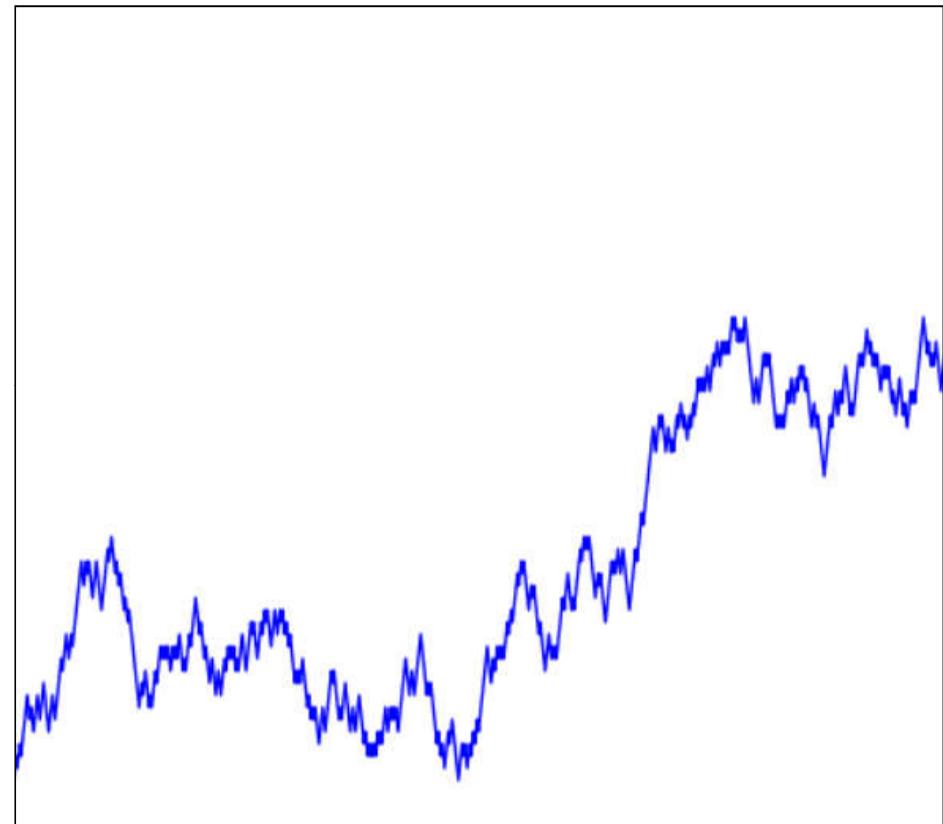
---

- Stochastic hill-climbing
  - Random selection among the uphill moves.
  - The selection probability can vary with the steepness of the uphill move.
- First-choice hill-climbing
  - Stochastic hill climbing by generating successors randomly until a better one is found.
- Random-restart hill-climbing
  - A series of Hill Climbing searches from randomly generated initial states
- Simulated Annealing
  - Escape local maxima by allowing some "bad" moves (not uphill move)  
but gradually decrease their frequency

???

# Random Walk

- Find the global maximum
- But extremely inefficient



---

# Simulated annealing

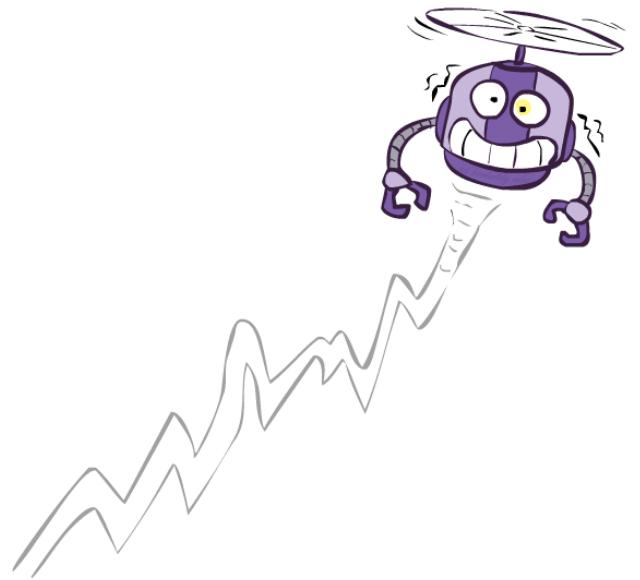
---

- Resembles the annealing process used to cool metals slowly to reach an ordered (low-energy) state
- Basic idea:
  - Allow “bad” moves occasionally, depending on “temperature”
  - High temperature => more bad moves allowed, shake the system out of its local minimum
  - Gradually reduce temperature according to some schedule



# Simulated annealing algorithm

```
function SIMULATED-ANNEALING(problem,schedule) returns a state
    current ← problem.initial-state
    for t = 1 to ∞ do
        T ← schedule(t)
        if T = 0 then return current
        next ← a randomly selected successor of current
        ΔE ← next.value – current.value
        if ΔE > 0 then current ← next
        else current ← next only with probability  $e^{\Delta E/T}$ 
```



الگوریتم :Simulated annealing

از یک حالتی شروع می کنیم و یک temperature داریم  
توفی زمان 0 که هیچی

از یک حالتی شروع می کنیم و همسایه های این حالت رو پیدا می کنیم و بعد می خوایم بین این همسایه ها یکی رو انتخاب بکنیم و هر همسایه ای یک هیوریستیکی داره نسبت به هیوریستیک ما که این بهمون یک دلتا  $E$  میده ینی اگر ما اون رو انتخاب کردیم هیوریستیک فانکشن رو اینقدر کم کردی یا زیاد کردی --> حالا این دلتا  $E$  یک اختلاف است و ما می خوایم بریم به سمت ماکزیمم ینی ما اگر این همسایه رو انتخاب کنیم اینقدر این هیوریستیک بهتر شده و اینقدر به سمت قله حرکت کردیم پس میزان جذابیت اون رو داره این دلتا  $E$  نشونمون میده

اگر یک همسایه ای مقدارش نسبت به الانمون برتر باشه این دلتا  $E$  یک مقدار کوچیکتری پیدا می کنه اگر اون همسایه ما همسایه بهتری بود ینی همیشه داره به ما پیشنهاد میده برو به همسایه بهتری اونو انتخاب می کنیم ولی اگه اون همسایه، همسایه بدتری بود اونو نمیداریم کنار بلکه اونو با یک احتمالی انتخاب می کنیم که این احتمال  $e^{-\frac{E}{T}}$  به توان منفی دلتا  $E$  به روی  $T$  است --> پس وقتایی که دلتا  $E$  بزرگه ما با یک احتمال خیلی کمتری سر و کار داریم ولی این احتمال صفر نیست پس حالت های بعد احتمال اینکه انتخاب بشن کمتره ولی صفر نیست

این  $T$  گذر زمان رو نشون میده

این  $T$  وقتی که خیلی کوچیک میشه مقدار کسر بزرگ میشه  
ینی هم با گذر زمان احتمال حالت های بد رو داریم کم می کنیم و هم به اندازه خود بد بودنشون داریم  
جریمنشون می کنیم

# The effect of varying $\Delta$ for fixed T=10

---

$\Delta$	$e^{\Delta/10}$
-43	0.01
-13	0.27
0	1.00

**A negative value for  $\Delta$  implies a downhill step. The lower the value of  $\Delta$ , the bigger the step would be downhill, and the lower the probability of taking this downhill move.**

---

# The effect of varying T for fixed $\Delta = -13$

---

T	$e^{-13/T}$
1	0.000002
50	0.56
$10^{10}$	0.9999...

*The greater the value of T, the smaller the relative importance of  $\Delta$  and the higher the probability of choosing a downhill move.*

---

# Local beam search

---

- Keep track of  $k$  “current” states instead of one
  - Initially:  $k$  random initial states
  - Next: determine all successors of the  $k$  current states
  - If any of successors is goal → finished
  - Else select  $k$  best from the successors and repeat.
- Major difference with  $k$  random-restart search
  - Information is shared among  $k$  search threads.
- Can suffer from lack of diversity.
  - **Stochastic** variant: choose  $k$  successors at proportionally to the state success.

الگوریتم بیم سرج:

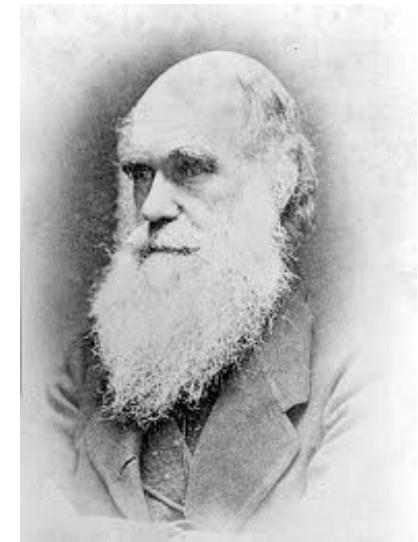
این الگوریتم مثل الگوریتم تپه نوردی است ولی به جای اینکه با یک حالت کار بکنه با  $k$  تا حالت کار میکنه یعنی میگه  $k$  تا حالت رو در نظر بگیر و هر کدام از این  $k$  تا حالت دوباره یک تعدادی فرزند دارند

حالا میخوایم گام بعدی رو برداریم --> توی تپه نوردی بین اینا بهترین رو انتخاب میکردیم یا به تصادف یکی رو انتخاب می کردیم یا ... --> الان توی بیم سرج میایم بین این ها جستجو انجام میدیم یعنی بین همه این ها یک مقایسه ای انجام میدیم و  $k$  تا بهترینش رو انتخاب می کنیم و اینا رو میداریم مبنا و می ریم گام بعدی

???

# Local beam search

- Basic idea:
  - $K$  copies of a local search algorithm, initialized randomly
  - For each iteration
    - Generate ALL successors from  $K$  current states
    - Choose best  $K$  of these to be the new current states
- Or,  $K$  chosen randomly with a bias towards good ones
- Why is this different from  $K$  local searches in parallel?
  - The searches **communicate**! “Come over here, the grass is greener!”
- What other well-known algorithm does this remind you of?
  - Evolution!



فرق این بیم سرچ با اینکه ما بخوایم الگوریتم تپه نوردی رو  $k$  بار انتخاب بکنیم چیه؟  
حافظه اینا خیلی کم است

# biological Evolution

---

- The organisms that are ill-suited for an environment have little chances to reproduce (natural selection)
- The best fitting have more chances to survive and reproduce



تکامل:

# Evolutionary algorithm

---

- Imitating Nature
  - **Reproduction:** Offspring are similar to their parents
  - **Random mutations** occur and they can bring to better (or worse) fitting individuals
  - “The Origin of the Species on the Basis of Natural Selection” C. Darwin (1859)
  - **Encoding:** An organism is fully represented by its DNA string, that is a string over a finite alphabet (4 symbols) Each element of this string is called gene
  - Inspire from nature: GA, Ant colony, Bee colony, PSO,....

الگوریتم ژنتیک:

یکسری والدینی وجود دارن که فرزندانشون؟؟؟

انگار این پدران و مادران یه ویژگی هایی دارن که این ویژگی ها توی یک DNA کد شده و ما اگر اون DNA رو داشته باشیم انگار اون پدر رو داریم پس ما یک بازنمایی داریم ینی یک Encoding داریم

بحث جهش: یک شرایطی برای ژن پیش میاد که این ژن ها تکامل پیدا می کنن ینی جهش پیدا می کنن

؟؟؟

# Genetic algorithms

---

- Search algorithms based on the mechanics of natural selection.
- A highly simplified computational model of biological evolution.
- GA: Developed by John Holland in the ‘60s.(pioneer of GA)

John Henry Holland



---

# Genetic algorithms

---

- A **state** (an individual) is represented as a **string over a finite alphabet (often a string of 0s and 1s)**, just as DNA that is a string over the alphabet **ACGT**.
- A **successor state** is generated by **combining two parent states**
- Evaluation function (fitness function). Higher values for better states.

توی این روش استیت ها یک رشته هستن  
می خواستیم از یک استیت بریم به یک استیت دیگه --> اینجا میگه استیت ها یک فرزندانی دارند  
می خوایم کیفیت استیت ها هم بسنجیم --> با Evaluation function می تونیم این کار رو بکنیم

???

# Genetic algorithms

---

- Start with  $k$  randomly generated states (**population**)
- Produce the next generation of states by

Selection,

Crossover,

And mutation

رویه الگوریتم ژنتیک:

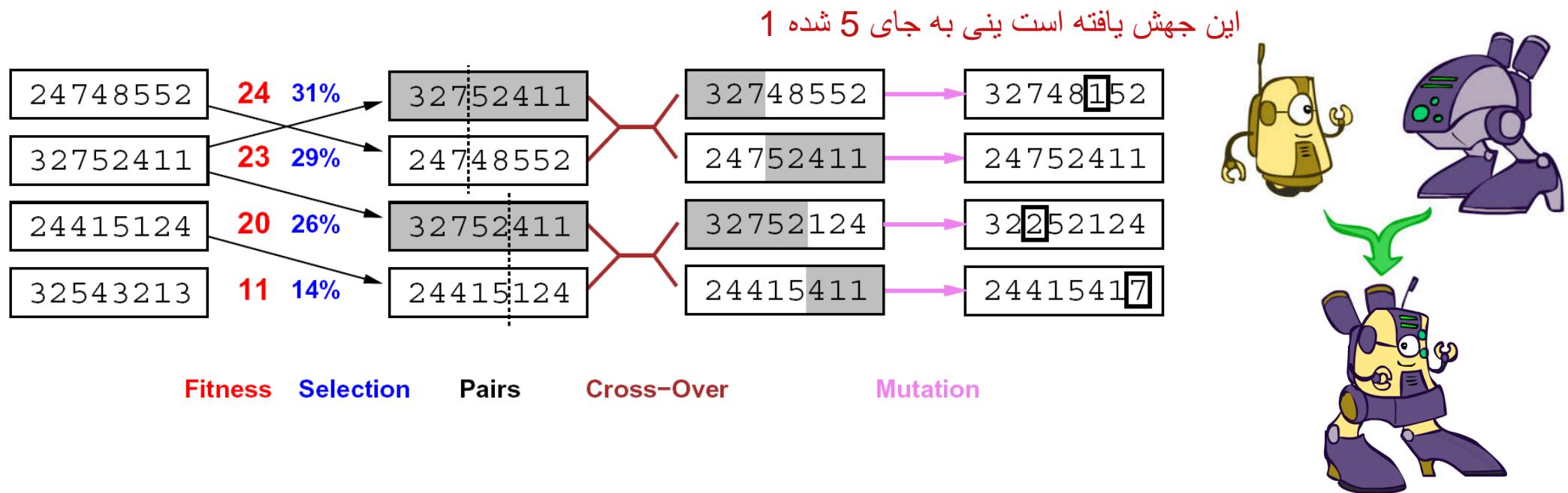
ما ابتدای کار یک جمعیتی تولید میکنیم ینی  $k$  تا استیت تولید میکنیم که میشه جمعیت اولیه ما، ما با این تعداد حالت شروع می خوایم برمی به اون استیت هدف برسیم  
اینجا توی ژنتیک ما محدود می کنیم:

ما  $k$  تا حالت داریم و میخوایم خوب هاشون رو انتخاب بکنیم و چجوری خوب هاشون رو انتخاب بکنیم؟ یه جوری باید انتخاب بکنیم که فقط در جهت ادم خوب ها نریم ینی ادم خوب ها رو انتخاب نکنیم برمی نسل بعد --> پس Selection ینی چجوری از یک نسلی یک تعدادی رو انتخاب بکنیم برمی نسل بعدی

Crossover: تولید مثل ینی ما چجوری بیایم این ها رو با هم ترکیب بکنیم که یک نسل جدیدی بسازیم

mutation: اجازه بدیم مثل طبیعت یه جاهایی این ها جهش پیدا بکن ینی یک سری از ویژگیهاش جهش پیدا بکنه

# Genetic algorithms



- Genetic algorithms use a natural selection metaphor
  - Resample  $K$  individuals at each step (selection) weighted by fitness function
  - Combine by pairwise crossover operators, plus mutation to give variety

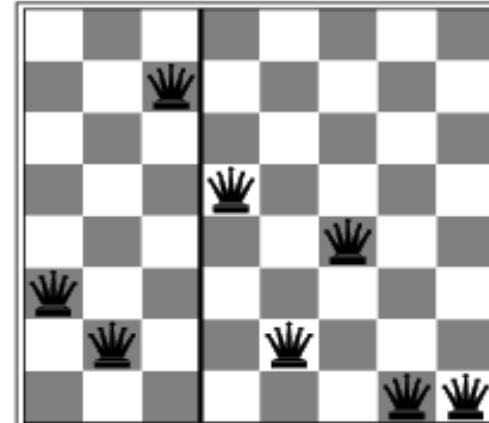
؟؟؟ اصلاً نفهميدم

# Genetic algorithms

---

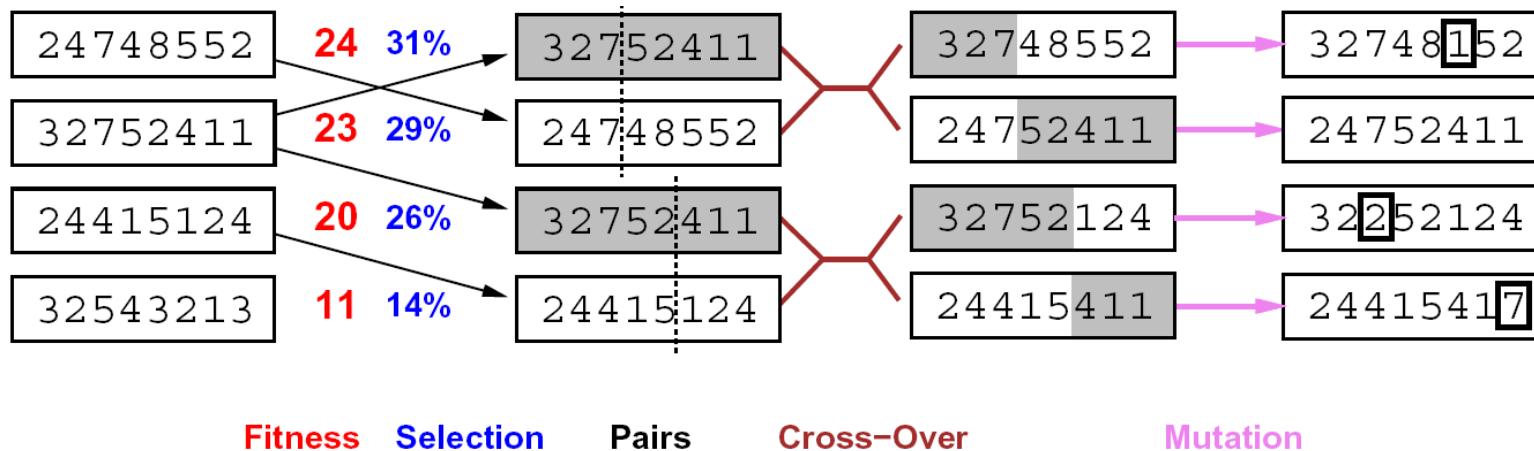
- it is better to have bigger fitness function so create new ones
- Fitness function: number of non-attacking pairs of queens (min = 0, max =  $8 \times 7/2 = 28$ )

24748552



مسئله 8 وزیر با الگوریتم ژنتیک:  
یه جوری باید اینو بازنمایی بکنیم  
حالا میخوایم هیوریستیک فانکشن رو برای این تعریف بکنیم --> هیوریستیک فانکشنی که قبلا  
داشتیم؟؟؟

# Genetic algorithms



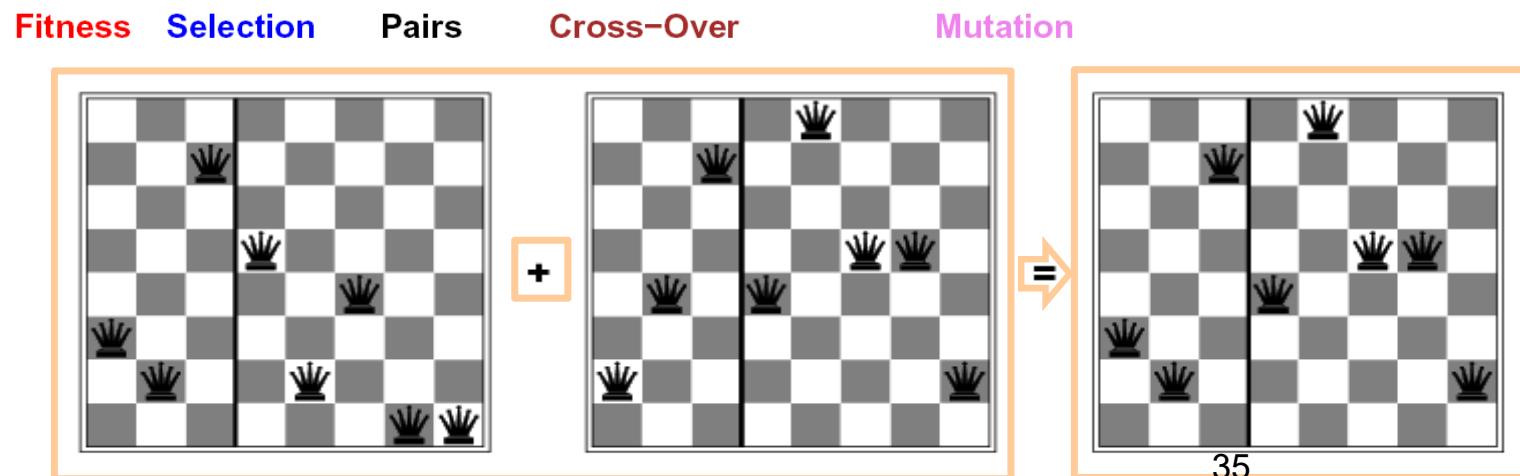
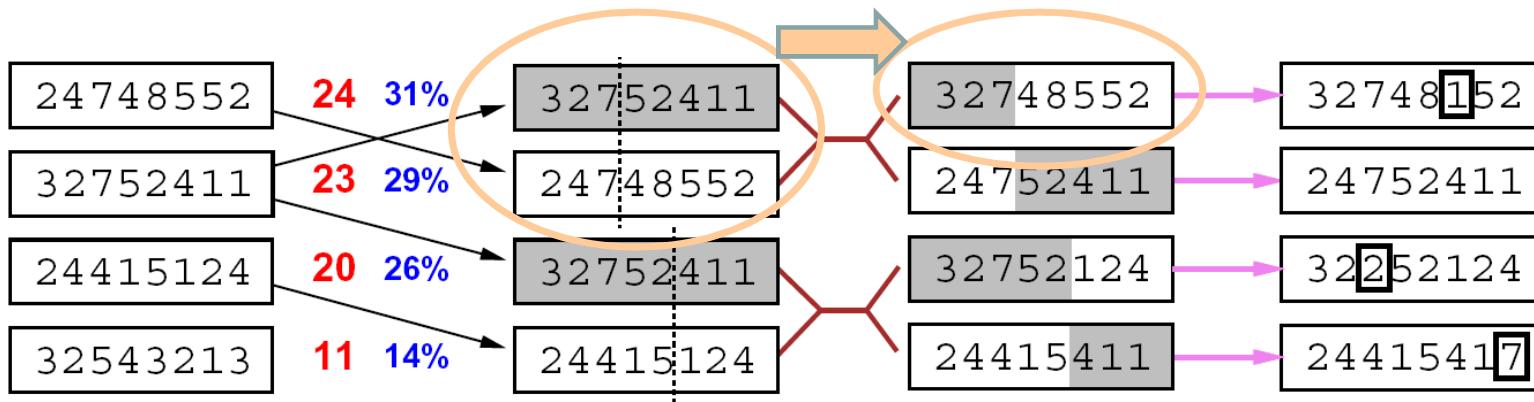
## ■ fitness to selection Probability

- $24/(24+23+20+11) = 31\%$
- $23/(24+23+20+11) = 29\% \text{ etc.}$

از یک حالت اولیه شروع می کنیم:

اون خط تصادفی است و اینا رشتن که 8 تاس

# Genetic algorithms



mutation: یکی از این وزیرها رو به صورت تصادفی بالا و پایین کن

# Decisions to make when applying a GA

---

- *What is the fitness function?*
- *How is an individual represented?*
- *How are individuals selected?*
- *How do individuals reproduce?*

برای اینکه بخوایم ژنتیک الگوریتم رو توی مسائل استفاده کنیم باید به 4 تا سوال جواب بدیم:

# A Genetic algorithm

---

```
function GENETIC_ALGORITHM( population, FITNESS-FN) return an individual
  input: population, a set of individuals
          FITNESS-FN, a function which determines the quality of the individual
  repeat
    new_population  $\leftarrow$  empty set
    loop for i from 1 to SIZE(population) do
      x  $\leftarrow$  RANDOM_SELECTION(population, FITNESS_FN)
      y  $\leftarrow$  RANDOM_SELECTION(population, FITNESS_FN)
      child  $\leftarrow$  REPRODUCE(x,y)
      if (small random probability) then child  $\leftarrow$  MUTATE(child )
      add child to new_population
    population  $\leftarrow$  new_population
  until some individual is fit enough or enough time has elapsed
  return the best individual
```

ما یک جمعیتی داریم و یک fitness function داریم و میخوایم این چرخه رو یک تعداد بار تکرار بکنیم و رویه اینطوری است که:

ما باید بیایم تک تک اعضای این جمعیت رو به اندازه سایز جمعیت میخوایم نسل جدید تولید بکنیم--> جمعیت رو می دیم به یک تابع رندوم سلکشنی که این رندوم سلکشنه میاد از این جمعیت یک نفر رو انتخاب میکنه که این میشه پدرس و با جستجوی دوم مادرش رو مشخص میکنیم و یک تابع دیگه داریم به نام reproduce که این میاد یک فرزند جدید تولید میکنه حالا با یک احتمالی این فرزند رو میدیم به تابع MUTATE یا جهش که با احتمال خیلی کوچیکی روی بعضی از ویژگی های این فرزنده یک جهشی تولید بکنه و در نهایت نتیجه رو به عنوان یک جمعیت new\_population تعریف ش می کنیم و این کامل که انجام شد جمعیت جدید امده است و همین رویه هی ادامه پیدا میکنه به این امید که ما توی جمعیت کسی رو پیدا بکنیم که استیت هدف باشه یا جمعیت به یک شرایطی برسن که همشون مثل هم بشن و هرچی که میخوایم بچه تولید بکنیم بچه هاشون شبیه فرزندانشون بشن و فایده دیگه نداره

# A Genetic algorithm (Cont.)

---

```
function REPRODUCE(x, y) return an individual
```

**input:** *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))
```

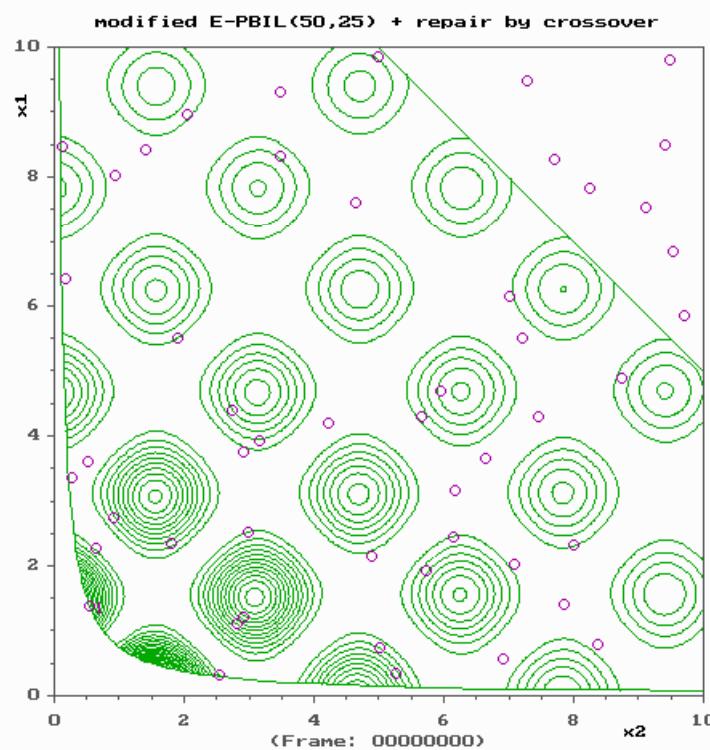
In this more popular version of GA, from each two parents, only one offspring is produced, not two.

دوتا تابع تولید مثل:

ما دوتا رشته  $y$  ,  $x$  داریم و می خوایم یک رشته جدید تولید بکنیم و کافیه که یک عدد تصادفی از موقعیت جایی که میخوایم این دوتا رشته رو پرش بکنیم تولید بکنیم و یک SUBSTRING از  $x$  میگیریم و یک SUBSTRING از  $y$  می گیریم و اینارو بهم می چسبونیم و فرزند نهایی رو تولید میکنیم

# Example

- Estimation of distribution algorithm over Keane's function

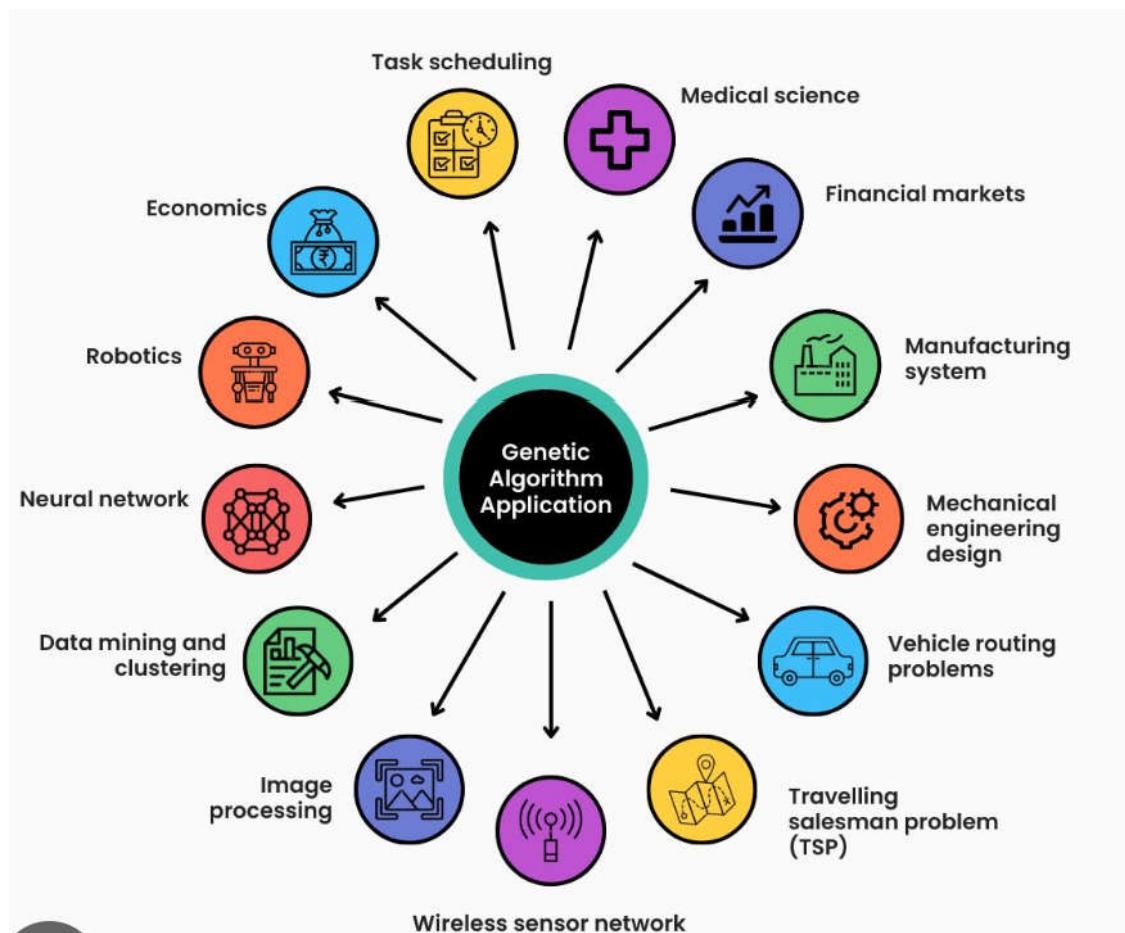


---

# Example



The 2006 NASA ST5 spacecraft antenna. This complicated shape was found by an evolutionary computer design program to create the best radiation pattern. It is known as an evolved antenna.



کاربردهای ژنتیک الگوریتم:

# Summary

---

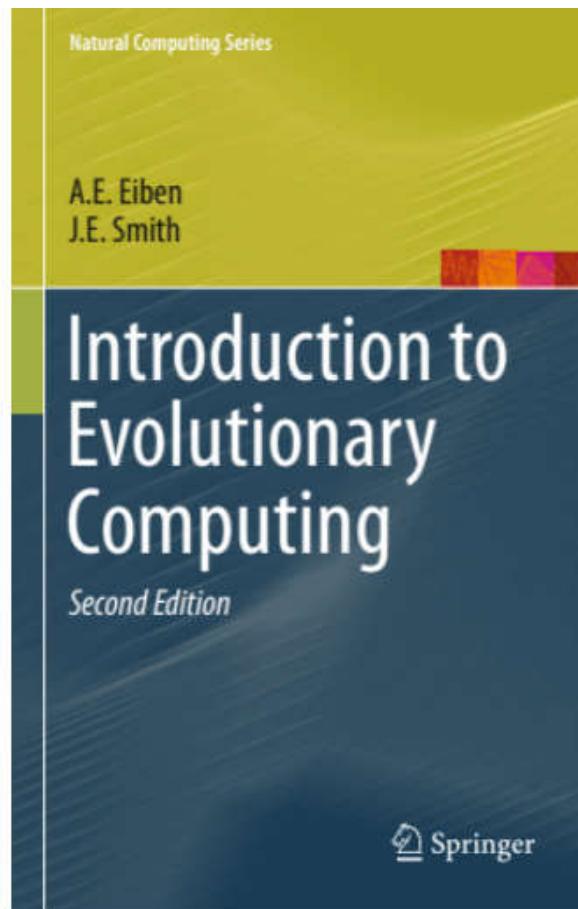
- Many configuration and optimization problems can be formulated as local search
- General families of algorithms:
  - Hill-climbing, continuous optimization
  - Simulated annealing (and other stochastic methods)
  - Local beam search: multiple interaction searches
  - Genetic algorithms: break and recombine states

Many machine learning algorithms are local searches

---

# Good Book

---



---

---

# **LOCAL SEARCH IN CONTINUOUS SPACES**

---

# Introduction

---

- Most real-world environments are continuous
- A continuous action space has an infinite branching factor

چیزایی که تا الان گفتیم همش توی یک محیط گسته بود که می خواستیم سرچ بکنیم  
اما خیلی از مسائل دنیای واقعی فضا، فضای پیوسته است

# Example: Weber Point

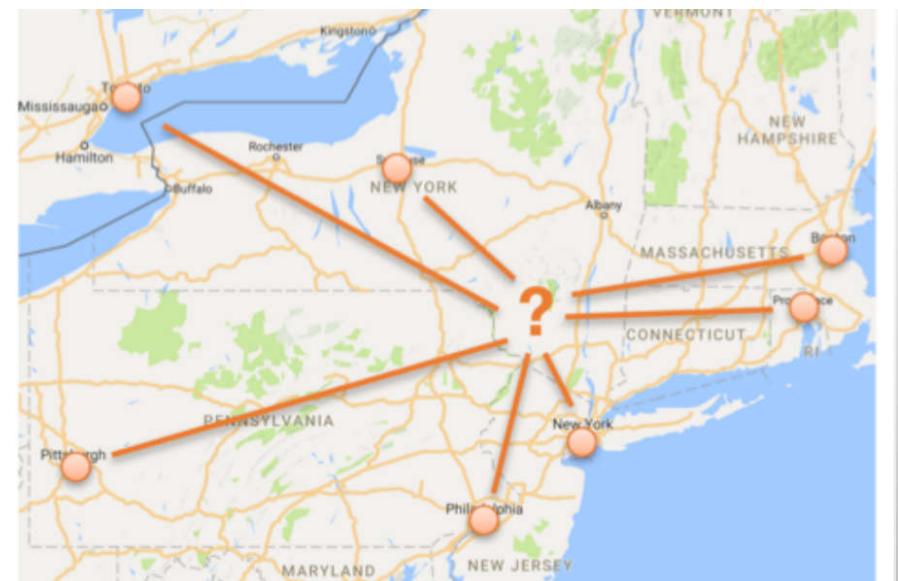
- Given a collection of cities (assume on 2D plane) find Best Airport Location?

how can we find the location that minimizes the sum of distances to all cities?

Denote the locations of the cities

- Write as the optimization problem:

$$\underset{x}{\text{minimize}} \sum_{i=1}^m \|x - y^{(m)}\|_2$$



مثال:

یک فرودگاهی می خواهد تاسیس بشه و ما یک تعداد شهر داریم و نزدیک شهر ها این فرودگاه را میخوایم قرار بدم به طوری که همه شهر ها بتوانن ازش استفاده بکنن هدف پیدا کردن موقعیت این فرودگاه است که این فرودگاه کجا باشه اگر فضای ما فضای 2 بعدی باشه راه حل چیست؟

ما به دنبال یک نقطه ای هستیم که اون نقطه جمع فاصله هاش از بقیه شهرها کمینه باشه --> پس یک تابع میخوایم و این تابع یک ورودی داره و این ورودی این شهر است اینجا ما داریم یک مسئله بهینه سازی رو حل میکنیم و مسائل بهینه سازی حول یک تابع هستن

تابع می تونه این باشه که توی صفحه هست--> ما  $m$  تا شهر داریم --> اگر مینیم این تابع رو پیدا بکنیم حالت بهینه رو پیدا کردیم

# Example: Machine Learning

---

- As we will see in much more detail shortly, virtually all (supervised) machine learning algorithms boil down to solving an optimization problem

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^m \ell(h_{\theta}(x^{(i)}), y^{(i)})$$

---

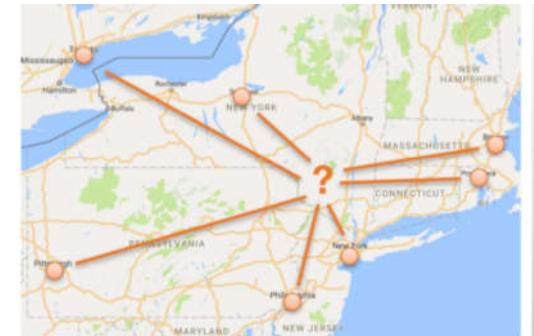
- $x^{(i)} \in \mathcal{X}$  are inputs
- $y^{(i)} \in \mathcal{Y}$  are outputs
- $\ell$  is a loss function
- $h_{\theta}$  is a hypothesis function parameterized by  $\theta$

مثال:

مانطور که به زودی با جزئیات بسیار بیشتری خواهیم دید، تقریباً همه الگوریتم های یادگیری ماشین (با نظرات) به حل یک مسئله بهینه سازی خلاصه می شوند.

# Example: Weber Point

- want to place three new airports anywhere in Romania, such that (the sum of squared straight-line) distances from each city on the map to its nearest airport is minimized.
- State Space: \*  
 $(x_1, y_1)$ ,  $(x_2, y_2)$ , and  $(x_3, y_3)$
- Goal State

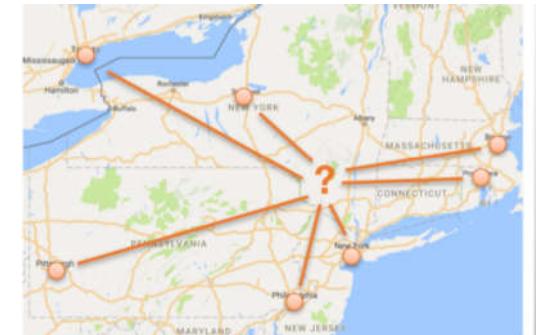


مثال:

می خوایم سه تا فرودگاه بسازیم که فاصلشون از همه اون شهرها توی نقشه کمینه باشه  
سه تا فرودگاه سه تا لوکیشن دارند که فضای استیت میشه این\*

# Example: Weber Point

- want to place three new airports anywhere in Romania, such that (the sum of squared straight-line) distances from each city on the map to its nearest airport is minimized.
- State Space:  
 $(x_1, y_1), (x_2, y_2)$ , and  $(x_3, y_3)$
- Goal State(Minimum f value)



$$f(\mathbf{x}) = f(x_1, y_1, x_2, y_2, x_3, y_3) = \sum_{i=1}^3 \sum_{c \in C_i} (x_i - x_c)^2 + (y_i - y_c)^2 .$$

How to Solve This Problem?

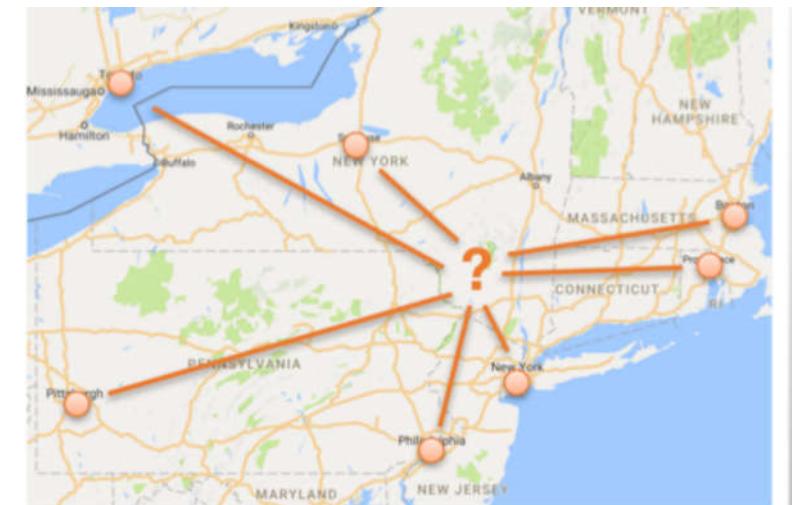
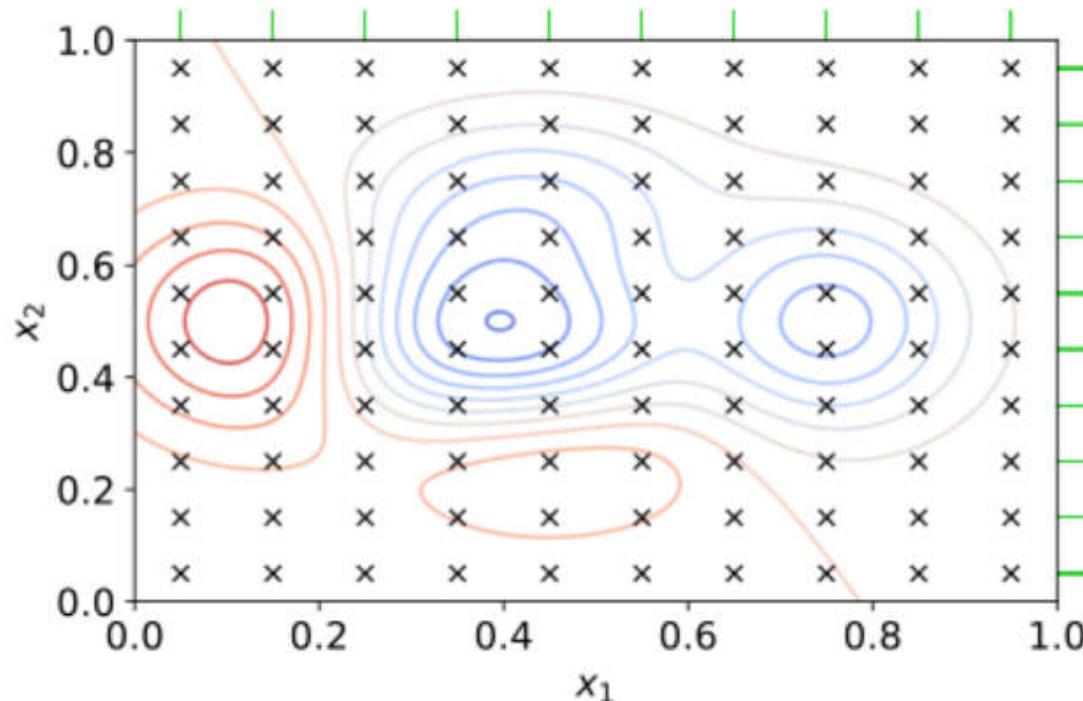
فاصله هر کدام از این فرودگاه ها تا شهرها رو حساب میکنه

از کجا تشخیص بدیم این گل استیت هست یا نه؟  
ما دقیق نمی دونیم کجا نقطه خوب هستن فقط میدونیم اونجایی نقطه خوب هست که یک شرایطی  
داره که اون شرایط مینیم بودنه یا ماکزیم بودنه

ما میخوایم جواب نهایی این تابع رو پیدا بکنیم چی کار باید بکنیم؟  
???

# Continues to discrete

- Search in Discrete (Local Search)



نکته: فرودگاهی رو به ما بده که فاصله اش از همه شهرها کمینه باشه یا اینکه حتی می تونیم جمعیت این شهرها هم بهش اضافه بکنیم ینی فرودگاهی رو بهمون بده که اون شهرهایی که پر جمعیت هستن این فرودگاه نزدیکتر به اون ها باشه تا اون شهرهایی که کم جمعیت هستن

اینجا میخوایم یکم پیوسته هم بهش اضافه کنیم ینی نمی خوایم گسته بکنیم ینی بعضی جاها گسته کردن مسئله ما رو از جواب بهینه دور میکنه و همون حالت پیوسته می تونه یک جواب خوبی بهمون بده

# Gradient Descent

---

- What is goal state: a Minimum by solving the equation  $\nabla f=0$
- Use local information to find the minimum
- Some close form

$$f(\mathbf{x}) = f(x_1, y_1, x_2, y_2, x_3, y_3) = \sum_{i=1}^3 \sum_{c \in C_i} (x_i - x_c)^2 + (y_i - y_c)^2.$$

$$\nabla f = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_3} \right).$$

$$\frac{\partial f}{\partial x_1} = 2 \sum_{c \in C_1} (x_1 - x_c).$$

حالت هدف چیست:  $a$  حداقل با حل معادله  $\nabla f=0$   
از اطلاعات محلی برای یافتن حداقل ها استفاده کنید  
برخی از فرم های نزدیک

-----

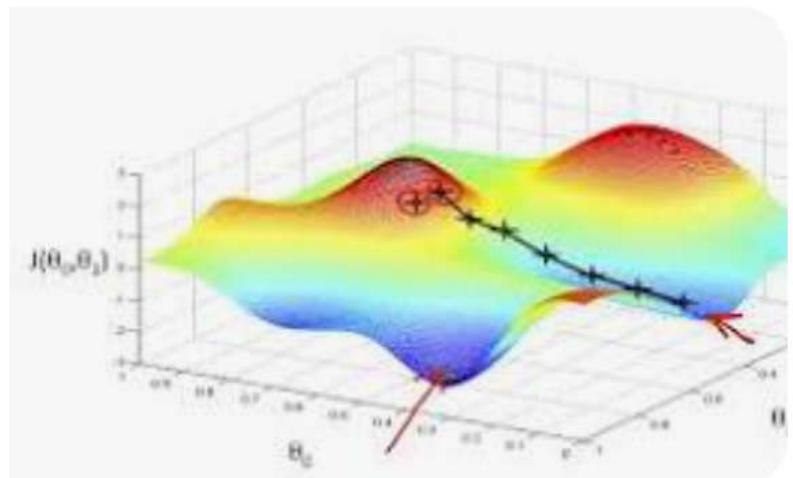
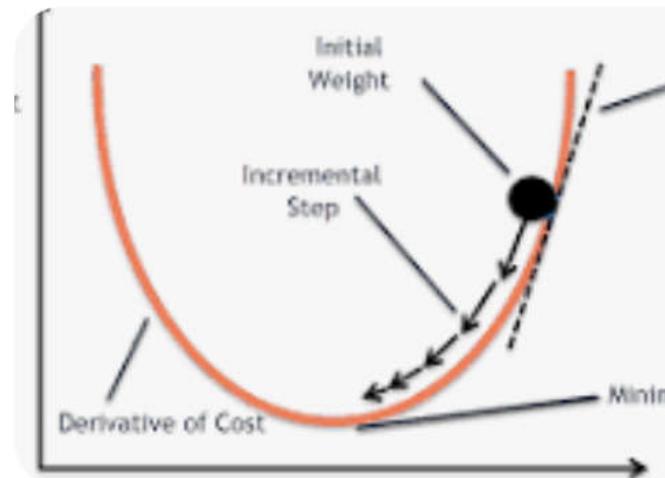
مینیمم و ماکزیمم های محلی توی فضای پیوسته  
مشتق جزئی کمک می گیریم

# Gradient Descent

- What is goal state: a Minimum by solving the equation  $\nabla f=0$
- Steepest ascent

$$\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla f(\mathbf{x}),$$

$$\frac{\partial f}{\partial x_1} = 2 \sum_{c \in C_1} (x_1 - x_c).$$



در جهت شب یک نقطه بساز در جهت اون شب حرکت کن و یک نقطه جدید تولید بکن

از اطلاعات محلی کمک میگیره که این اطلاعات محلی مشتق است در جهت اون شب حرکت بکنیم  
اندازه شب بهمون میگه که چقدر می خوایم پرش بکنیم --> بردار شب در چه جهتی است و با  
توجه به اون مقدار رو جابه جا بکنیم  
اونجاهايی که شب بيشتر است بيشتر حرکت بکنیم

# Gradient Descent

**Algorithm:** Gradient Descent

**Given:**

Function  $f$ , initial point  $x_0$ , step size  $\alpha > 0$

**Initialize:**

$$x \leftarrow x_0$$

**Repeat until convergence:**

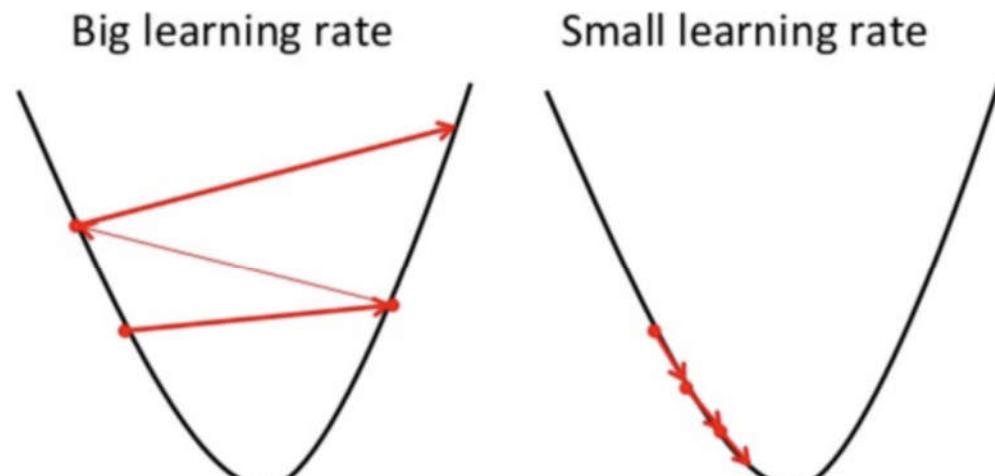
$$x \leftarrow x - \alpha \nabla_x f(x)$$

---

- Step Size (alpha or Learning Rate)

$$\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla f(\mathbf{x}),$$

- Convex Function (local Max = Global Max)



age: Niklas Donges

اين پرش اگر خيلی زياد باشه ميشه شكل و مى خوايم شيب رو هى کوچك کنيم او لش مقادير بزرگ  
بره و بعدش کوچيك بره

# Gradient Map

---

- Gradient\_Descent\_in\_2D.webm

---

# Signal Separation



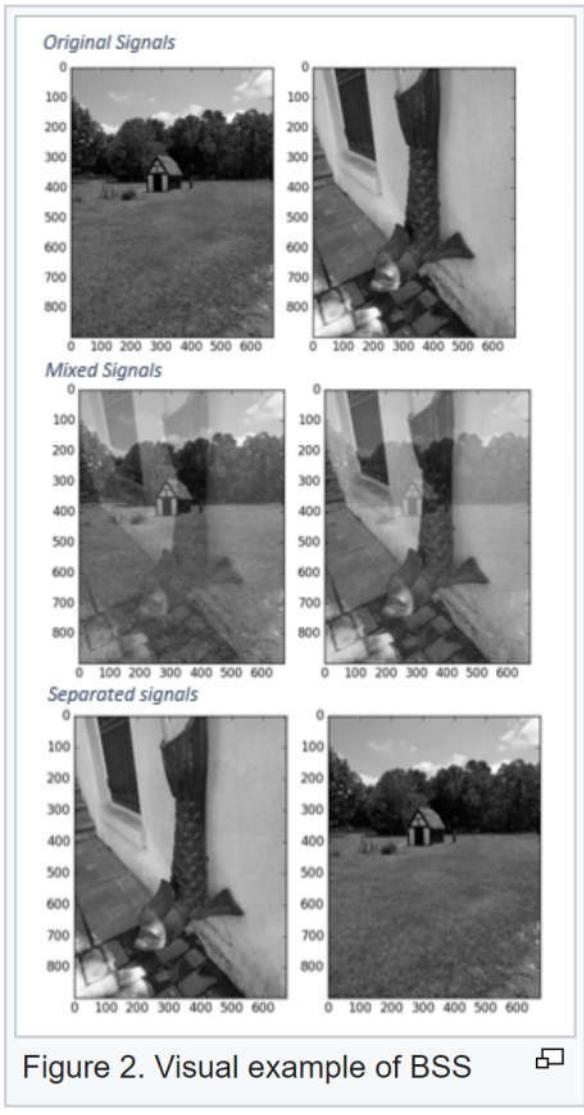
$$\begin{aligned} \mathbf{x}(t) &= \mathbf{A} \quad \mathbf{s}(t) \\ \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} &= \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} s_1(t) \\ s_2(t) \end{pmatrix} \end{aligned}$$

$$\begin{aligned} \mathbf{y}(t) &= \mathbf{W} \quad \mathbf{x}(t) \\ \begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix} &= \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} \end{aligned}$$

ICA estimates a demixing matrix  $W$  that satisfies  $W = A^{-1}$  using only  $\mathbf{x}(t)$ .

---

# Signal Separation



---