



دانشگاه علم و صنعت ایران

دانشگاه علم و صنعت
دانشکده مهندسی کامپیوتر

الگوریتم‌های جستجوی محلی و مسائل بهینه‌سازی

«هوش مصنوعی: رهیافتی نوین»، فصل ۴

مدرس: آرش عبدی هجراندoust

نیمسال اول ۱۴۰۲-۱۴۰۱

رؤس مطالب

مقدمه

معرفی جستجوهای محلی

(Hill-climbing search)

شبیه‌سازی ذوب فلزات

(Local beam search)

(Genetic algorithms)

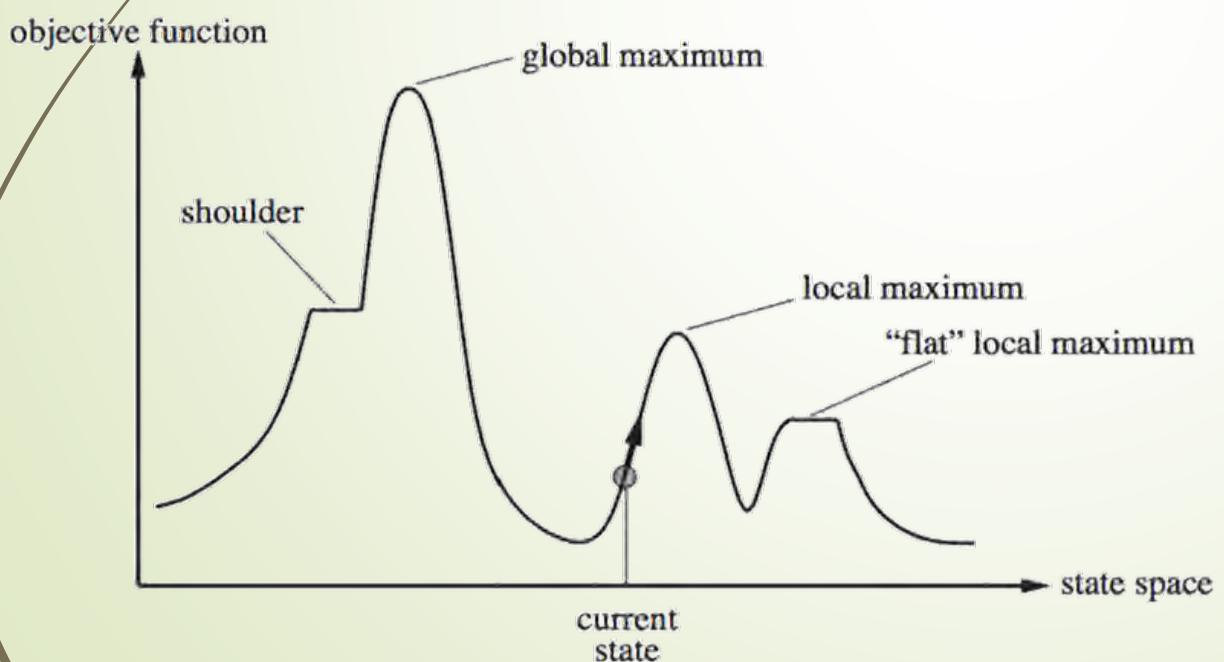
مقدمه

- ❖ آنچه که تا کنون دیدیم، الگوریتم‌های جستجویی بودند که به دنبال یافتن یک مسیر در فضای حالت برای رساندن عامل از یک حالت اولیه به یک حالت هدف بودند.
- ❖ مانند مسئله‌ی جاده‌های رومانی، پازل ۸تایی و ...
- ❖ در بسیاری از مسائل، مسیر راه حل (یا نحوه‌ی رسیدن به هدف) اهمیت ندارد؛ خود حالت هدف پاسخ مسئله است. در چنین مواردی می‌توان از الگوریتم‌های جستجوی محلی بهره گرفت.
- ❖ مانند مسئله ۷- وزیر
- ❖ در این‌گونه مسائل: فضای حالت = مجموعه پیکربندی‌های کامل
- ❖ هدف = یافتن یک پیکربندی که محدودیت‌های مسئله را ارضاء کند

معرفی جستجوی محلی

- ❖ جستجوی محلی یک (یا چند) حالت را به عنوان حالت فعلی در نظر می‌گیرد و تنها به حالت‌های همسایه‌ی آن انتقال می‌یابد.
- ❖ یک حالت "فعلی" را به تنها یابی در نظر بگیر؛ سعی کن آن را بهبود ببخشی.
- ❖ مزایا
- ❖ استفاده از حافظه بسیار کم ($O(C)$ که C یک عدد ثابت است)
- ❖ زیرا این الگوریتم‌ها مسیر طی شده را ذخیره نمی‌کنند.
- ❖ یافتن راه حل‌های معقول در اغلب موارد در فضاهای حالت بزرگ و یا نامحدود
- ❖ مفید برای مسائل بهینه‌سازی محض
- ❖ یافتن بهترین حالت بر طبق تابع هدف (objective function)

نمای فضای حالت (State Space Landscape)



❖ الگوریتم‌های جستجوی محلی landscape را کاوش می‌کنند.

❖ نما، شامل «محل» (حالت) و «ارتفاع» (مقدار تابع هیوریستیک هزینه یا تابع هدف) است.

❖ اگر ارتفاع متناظر با هزینه باشد، آن‌گاه هدف یافتن عمیق‌ترین دره (یک کمینه سراسری) است

❖ اگر ارتفاع متناظر با تابع هدف باشد، آن‌گاه هدف یافتن بلندترین قله (یک بیشینه سراسری) است.

- **کامل و بهینه** بودن الگوریتم‌های جستجوی محلی به چه معناست؟

- کامل: اگر جوابی وجود دارد، یکی را پیدا می‌کند.

- بهینه: بهترین را پیدا می‌کند.

جستجوی تپه‌نوردی (Hill-climbing search)

- ❖ یک وضعیت دلخواه را به عنوان وضعیت فعلی در نظر می‌گیرد سپس از میان همسایه‌های وضعیت فعلی، **بهترین همسایه** را انتخاب می‌کند.
- ❖ مدام در جهت افزایش مقدار حرکت می‌کند (یعنی به سمت بالای تپه).
- ❖ زمانی متوقف می‌شود که به قله‌ای برسد که در آن جا هیچ همسایه‌ای مقدار بیشتری نداشته باشد.
- ❖ ساختمن داده گرہ فعلی تنها نیاز به ثبت حالت و مقدار تابع هدفش دارد. (نه مسیر)

function HILL-CLIMBING(*problem*) returns a state that is a local maximum

current \leftarrow **MAKE-NODE(*problem.INITIAL-STATE*)**

loop do

neighbor \leftarrow a highest-valued successor of *current*

if *neighbor.VALUE* \leq *current.VALUE* **then return** *current.STATE*

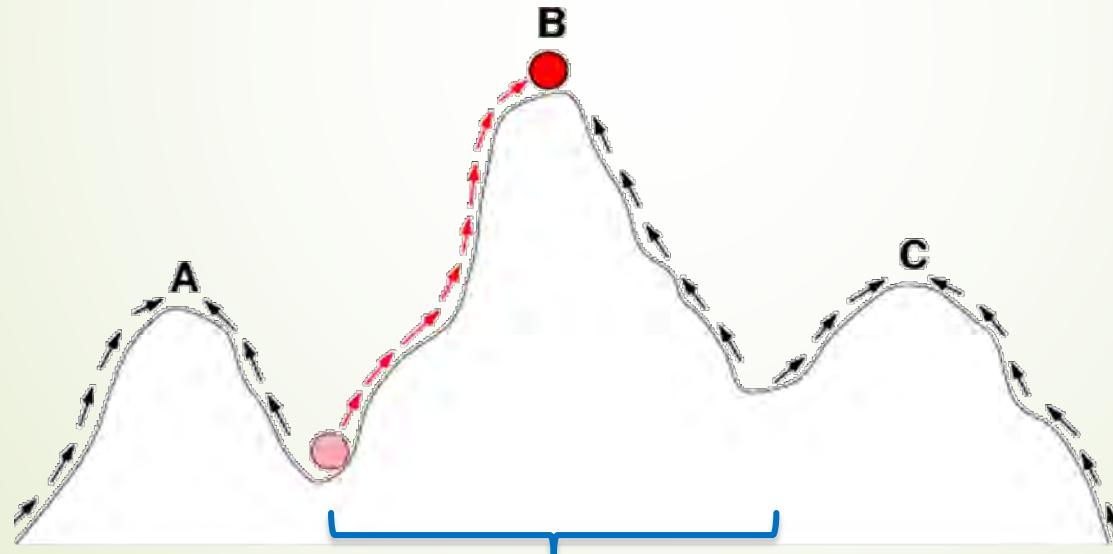
current \leftarrow *neighbor*

جستجوی تپه‌نورده...

❖ یک جستجوی محلی حریصانه است زیرا تپه نورده، فراتر از همسایه‌های مجاور حالت فعلی را نگاه نمی‌کند.

❖ گرادیان صعودی/نزولی (Gradient Ascent/Descent)

❖ به سرعت به سمت هدف پیش می‌رود زیرا به راحتی حالت بد را بهبود می‌بخشد.



هنگامی بهینه است که از
یکی از این حالات شروع به
جستجو کند.

جستجوی تپه‌نوردی – مثال

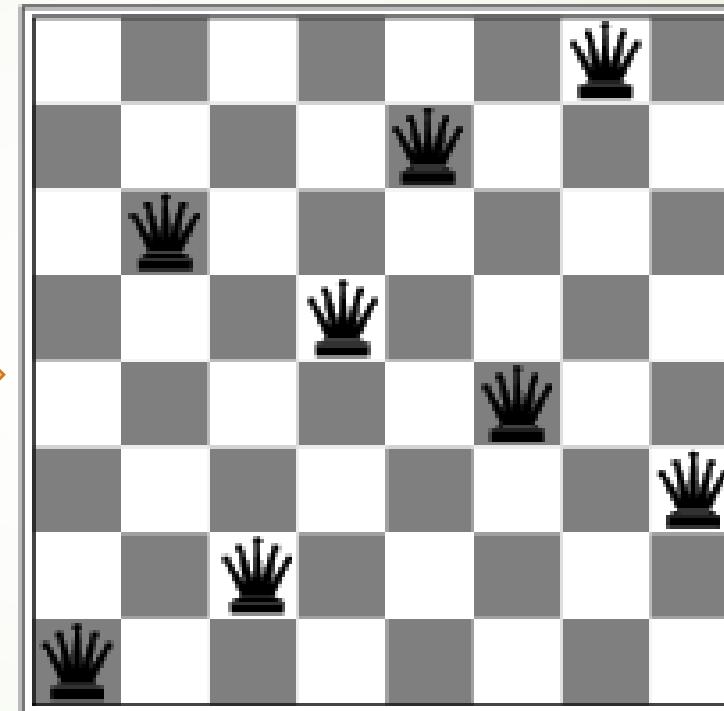
- ❖ n وزیر را در یک صفحه شطرنج $n \times n$ به‌گونه‌ای قرار بده که هیچ دو وزیری در یک سطر، ستون و یا قطر قرار نگیرند.
- ❖ ۸-وزیر
 - ❖ فرمول‌بندی حالت کامل: هر حالت شامل ۸ وزیر، یک وزیر در هر ستون (8^8 حالت)
 - ❖ پسین‌های یک حالت، همه حالت‌های ممکنی است که با جابه‌جایی یک وزیر به مربع دیگری در همان ستون تولید می‌شود.
 - ❖ هر حالت $8 \times 7 = 56$ پسین دارد.
- ❖ تابع هیوریستیک h تعداد جفت وزیرهایی است که در حال حمله به یکدیگر هستند.
- ❖ کمینه سراسری این تابع صفر است که فقط در راه حل کامل اتفاق می‌افتد.

جستجوی تپه‌نوردی – مثال ...

18	12	14	13	13	12	14	14
14	16	10	15	12	14	12	16
14	12	18	13	15	12	14	14
15	14	14	14	15	13	16	13
17	14	17	15	15	14	16	16
17	15	16	18	15	15	15	17
18	14	15	16	15	14	15	16
14	14	10	17	12	14	12	18

$h=17$

بهترین پسین دارای $h=12$ است.

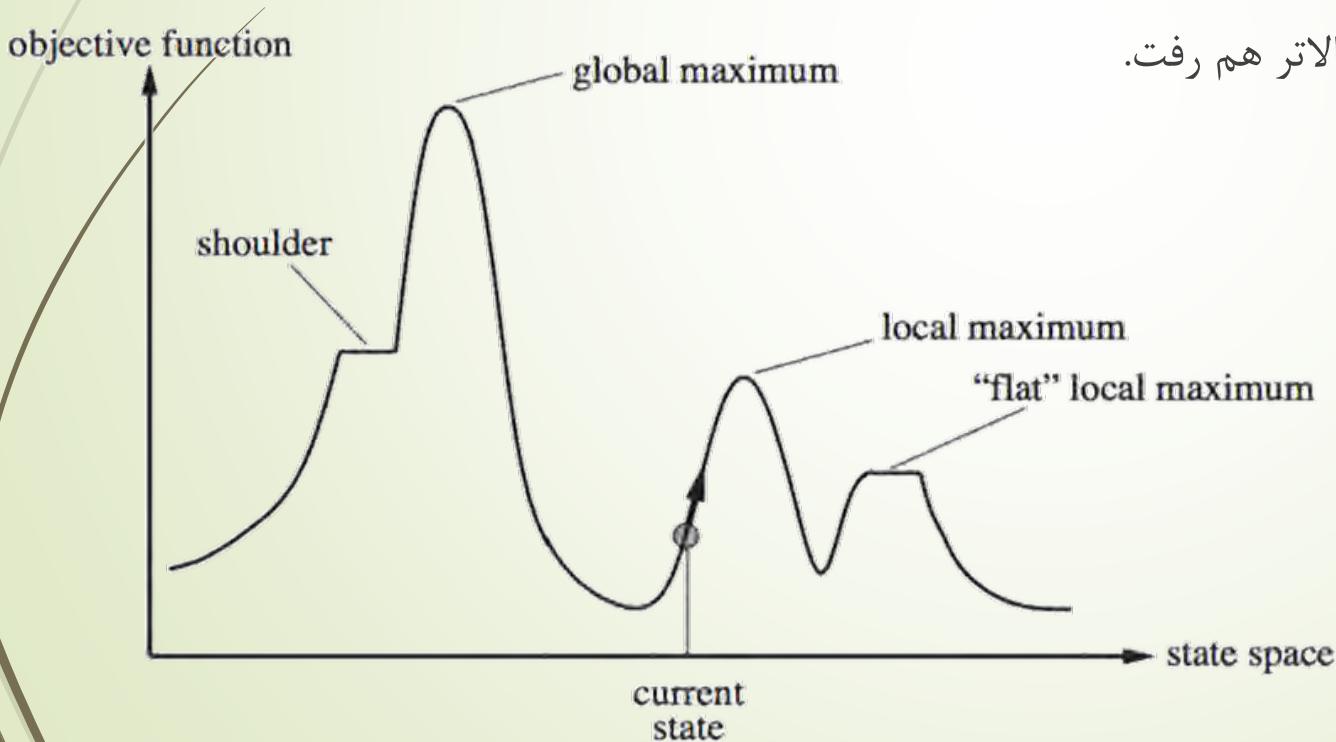


$h=1$

$h>1$

مشکلات جستجوی تپه‌نوردی

- ❖ بیشینه‌های محلی (Local Maxima): قله‌ای که از تمامی حالت همسایگانش بلندتر باشد اما از بیشینه‌ی سراسری پایین‌تر باشد.
- ❖ فلات‌ها (Plateau): ناحیه‌ای از نمای فضای حالت که در آن تابع ارزیاب، ثابت است.
 - ❖ بیشینه محلی مسطح: هیچ مسیر رو به بالایی وجود ندارد.



مشکلات جستجوی تپه‌نوردی ...

❖ دماغه Ridge (Ridge): یک رشته بیشینه محلی که گذشتن از آن‌ها برای الگوریتم‌های حریصانه بسیار مشکل است.



- ❖ با شروع از یک حالت اولیه‌ی ۸ وزیر که به‌طور تصادفی تولید شده است، تپه‌نوردی در ۸۶ درصد اوقات گیر می‌کند.
- ❖ با این حال در موارد موفق، به‌طور متوسط تنها ۴ گام و در مواردی که گیر می‌کند تنها ۳ گام برمی‌دارد.

نسخه‌های مختلف جستجوی تپه‌نوردی

❖ حرکت به کناره (Sideways move)

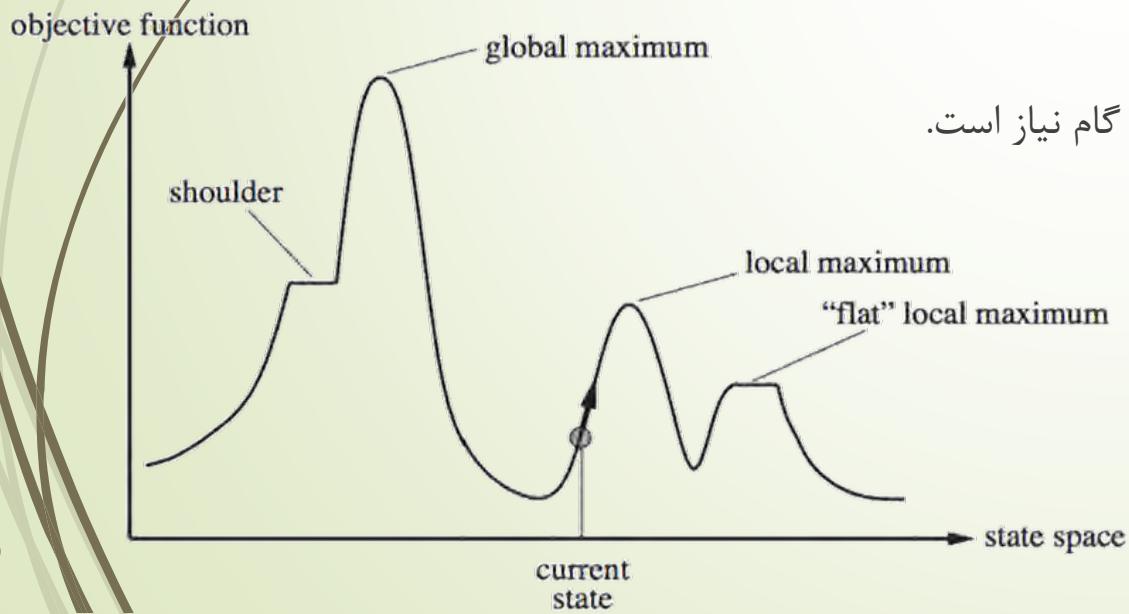
❖ از آنجا که الگوریتم هنگام رسیدن به یک فلات گیر می‌کند، انجام حرکت به اطراف به امید آن که فلات یک شانه باشد مناسب است.

❖ مشکل: اگر الگوریتم به یک ماکزیمم محلی صاف برسد در یک حلقه بینهایت گیر خواهد کرد.

❖ راه حل: محدود کردن تعداد حرکات متوالی به اطراف

❖ برای مثال در مورد مسئله‌ی ۸ وزیر اگر تعداد حرکات به ۱۰۰ محدود شده باشد
درصد موفقیت از ۹۴ درصد به ۶۴ درصد افزایش می‌یابد.

❖ اما در موارد موفقیت به طور متوسط ۲۴ گام و در موارد شکست ۶۴ گام نیاز است.



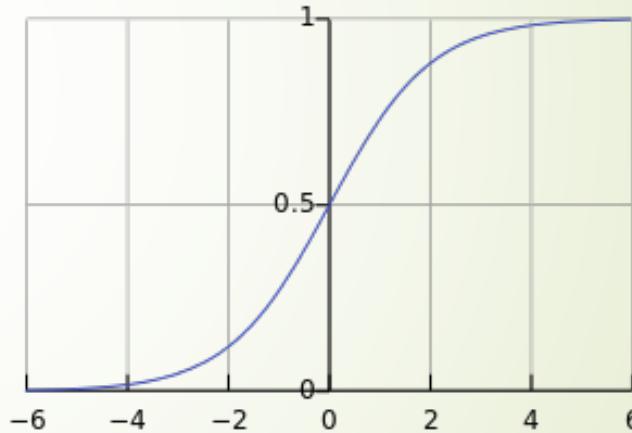
نسخه‌های مختلف جستجوی تپه‌نوردی ...

تپه‌نوردی اتفاقی (Stochastic hill climbing) ♦

- ❖ از میان همسایه‌هایی که بهتر از نقطه‌ی فعلی هستند یکی را به تصادف انتخاب می‌کند.
- ❖ احتمال این انتخاب می‌تواند براساس شیب حرکت‌های رو به بالا تغییر کند.
- ❖ توابع احتمال مختلفی وابسته به کاربرد می‌توان در نظر گرفت. برای مثال تابع احتمال سیگموید به صورت زیر تعریف می‌شود:

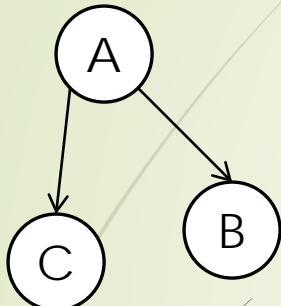
$$p = \frac{1}{1 + e^{-\Delta V}}$$

$$\Delta V = V_{next} - V_{current}$$



- ❖ جمع احتمال‌ها یک است؟ باید باشد؟
- ❖ نرمال سازی احتمال‌ها
- ❖ تولید عدد تصادفی یکنواخت

نسخه‌های مختلف جستجوی تپه‌نوردی ...



... Stochastic hill climbing

❖ برای مثال در شکل روبرو اگر احتمال محاسبه شده برای رفتن از گرهی A به گرههای B و C به ترتیب برابر با 0.4 و 0.6 باشد، می‌توان در پیاده‌سازی عددی تصادفی بین صفر و یک تولید کرد و در صورتی که عدد تولید شده در بازه [0,0.4] باشد به گرهی A و اگر در بازه [0.4,1] باشد به گرهی B منتقل شد.

❖ این روش معمولاً کندتر از بیشترین شبیه به نتیجه می‌رسد، اما در بعضی از نمایهای حالت، بهترین راه حل را پیدا می‌کند.

First choice hill climbing

❖ به صورت تصادفی پسین تولید می‌کند تا زمانی که پسینی تولید شود که از حالت فعلی بهتر باشد. سپس به آن نقطه می‌رود و این کار را تکرار می‌کند.

❖ معادل آن است که از بین پسینهای بهتر، یکی را تصادفی انتخاب کند.

❖ این راهبرد هنگامی مناسب است که یک حالت دارای تعداد زیادی (مثلاً هزار) پسین باشد.

نسخه‌های مختلف جستجوی تپه‌نوردی ...

- ❖ همه‌ی نسخه‌های قبلی ناکامل هستند
- ❖ گیر کردن در بیشینه‌ی محلی
- ❖ تپه نوردی با شروع مجدد تصادفی (Random restart hill climbing)
 - ❖ هر بار جستجوی تپه نوردی را از یک حالت تصادفی شروع می‌کند و هنگامی که یک هدف پیدا شد متوقف می‌شود. (الگوریتم تپه‌نوردی را چندین بار اجرا می‌کند).
 - ❖ این روش با احتمال نزدیک به یک، کامل می‌باشد.
 - ❖ سرانجام حالت هدف به عنوان یک حالت اولیه برگردانده می‌شود.

```
while state ≠ goal do
    run hill-climbing search from a random initial state
```

نسخه‌های مختلف جستجوی تپه‌نوردی ...

❖ p : احتمال موفقیت هر جستجوی تپه‌نوردی

❖ تعداد شروع مجدد مورد انتظار: $1/p$

❖ تعداد کل گام‌های مورد انتظار تا رسیدن به جواب بهینه: $(1/p - 1) \times n_f + n_s$

❖ n_f : میانگین تعداد گام‌ها در یک تپه‌نوردی ناموفق

❖ n_s : میانگین تعداد گام‌ها در یک تپه‌نوردی موفق

❖ منظور از «گام» رفتن از یک نقطه به بهترین همسایه‌ی آن است.

❖ در مورد مسئله‌ی ۸ وزیر

❖ در ۱۴ درصد از موارد وضعیت هدف پیدا می‌شود $\Leftrightarrow 1/p = 7.14$ و $p = 0.14$

❖ تعداد گام‌ها در موارد موفق و ناموفق به ترتیب ۴ و ۳ است $\Leftrightarrow n_s = 4$ و $n_f = 3$

❖ و نسخه sideways move با محدودیت ۱۰۰ حرکت متوالی به اطراف:

❖ درصد موفقیت $1/p = 1.06 - ٪.۹۴$

❖ تعداد گام‌ها در موارد موفق ۲۴ و در موارد شکست ۶۴ $\Leftrightarrow (1.06 - 1) \times 64 + 24 = 28$

تأثیر شکل نمای فضای حالت بر روی تپه‌نوردی

- ❖ شکل نمای فضای حالت اهمیت دارد
- ❖ اگر تعداد ماقزیمم‌های محلی و فلات‌ها کم باشد، تپه‌نوردی شروع مجدد تصادفی سریعاً راه حل خوبی را می‌یابد.
- ❖ نمای مسائل واقعی معمولاً از قبل شناخته شده نیست.
- ❖ مسائل NP-hard معمولاً دارای تعداد نمایی بیشینه‌ی محلی هستند.
- ❖ با این وجود الگوریتم می‌تواند یک بیشینه‌ی محلی خوب پس از تعداد کمی شروع مجدد بیابد.

جستجوی قدم زدن تصادفی (Random walk)

- ❖ یک نقطه را به عنوان نقطه‌ی فعلی در نظر می‌گیرد و به طور تصادفی به یکی از همسایه‌های نقطه‌ی فعلی می‌رود و این کار را تکرار می‌کند.
- ❖ احتمال رفتن به تمام نقاط همسایه (چه بهتر از نقطه فعلی و چه بدتر) یکسان است.
- ❖ بهترین نقطه‌ای که تاکنون دیده است را به خاطر سپرده و بر می‌گرداند.
- ❖ کامل ولی ناکارآمد (زمان زیاد برای یافتن جواب)
- ❖ چه زمانی قابل توصیه است؟



شبیه‌سازی ذوب فلزات (Simulated Annealing)

- ❖ تلاش در جهت ترکیب تپه نوردی با قدم زدن تصادفی که هم کارآمد و هم کامل باشد، منجر به الگوریتم شبیه‌سازی ذوب فلزات شد.
- ❖ ایده
 - ❖ از ماقزیمم‌های محلی با انجام حرکت‌های **بد** فرار کن.
 - ❖ اما به تدریج اندازه و تعداد حرکات بد (**به سمت پایین**) را کم کن.
- ❖ الگوریتم
 - ❖ یک حالت را به عنوان حالت فعلی در نظر بگیر.
 - ❖ هر بار یکی از همسایه‌های حالت فعلی را به طور تصادفی انتخاب کن.
 - ❖ اگر حالت بعدی انتخاب شده بهتر از حالت فعلی باشد، همواره به آن حالت بعدی خواهیم رفت.
 - ❖ در غیر این صورت، تنها با احتمال $e^{\Delta E/T}$ به آن حالت خواهیم رفت.

ΔE : دما و T : اختلاف سطح انرژی یا اختلاف مقدار حالت

شبیه‌سازی ذوب فلزات ...

function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state

inputs: *problem*, a problem

schedule, a mapping from time to “temperature”

current \leftarrow MAKE-NODE(*problem*.INITIAL-STATE)

for *t* = 1 **to** ∞ **do**

T \leftarrow *schedule*(*t*)

if *T* = 0 **then return** *current*

next \leftarrow a randomly selected successor of *current*

$\Delta E \leftarrow$ *next*.VALUE – *current*.VALUE

if $\Delta E > 0$ **then** *current* \leftarrow *next*

else *current* \leftarrow *next* only with probability $e^{\Delta E/T}$

شبیه‌سازی ذوب فلزات ...

- ❖ T به مرور کاهش می‌یابد (فلز به مرور سرد می‌شود)
- ❖ در مقادیر بالاتر T احتمال انجام حرکات بد (رفتن به سمت پایین) بیشتر است (مانند جستجوی تصادفی رفتار می‌کند).
- ❖ با کاهش T این احتمال کاهش یافته و در $T=0$ این احتمال به صفر می‌رسد. (مانند تپه‌نوردی رفتار می‌کند)
- ❖ می‌توان ثابت کرد که اگر T به اندازه کافی آرام کاهش یابد با احتمال نزدیک به یک، SA یک پاسخ بهینه را خواهد یافت.
- ❖ فرمول $e^{\Delta E/T}$ وقتی اعمال می‌شود که ΔE منفی است.
- ❖ کاهش $T \leftarrow$ کاهش احتمال حرکت بد

تست

کدام یک از گزینه‌های زیر در مورد الگوریتم تپه‌نوردی و simulated annealing غلط است؟

- ۱) الگوریتم تپه‌نوردی نزدیک‌ترین ماکریم را پیدا می‌کند.
- ۲) وقتی حرارت خیلی کم شود الگوریتم simulated annealing تبدیل به الگوریتم تپه‌نوردی می‌شود.
- ۳) اگر حرارت خیلی زیاد باشد و در طول الگوریتم کم نشود الگوریتم simulated annealing تبدیل به الگوریتم تصادفی می‌شود.
- ۴) الگوریتم simulated annealing مستقل از این‌که حرارت چه مقداری داشته باشد می‌تواند از مینمیم محلی فرار کند.

جستجوی شعاع محلی (Local Beam Search)

- ❖ به جای نگه داری تنها یک گره در حافظه، چندین گره را در حافظه نگه داری می کند.
- ❖ شروع با k حالت که به طور تصادفی ایجاد شده اند.
- ❖ در هر تکرار، تمام فرزندان برای هر k حالت تولید می شوند.
- ❖ اگر یکی از آنها حالت هدف بود جستجو متوقف می شود و در غیر این صورت از میان لیست کامل فرزندان، k تا از بهترین ها انتخاب می شوند و مرحله بالا تکرار می شود.
- ❖ تفاوت با جستجوی با شروع مجدد تصادفی
- ❖ در جستجوی با شروع مجدد تصادفی، هر فرآیند جستجو به طور مستقل از بقیه اجرا می شود.
- ❖ در جستجوی شعاع محلی، اطلاعات سودمند در بین k جستجوی موازی رد و بدل می گردد.
- ❖ اگر همسایه های یک نقطه از همسایه های نقاط دیگر بهتر باشد الگوریتم بر روی همسایه های آن نقطه متوجه می شود.

جستجوی شعاع محلی ...

❖ مشکل جستجوی شعاع محلی

❖ این روش ممکن است از نبود تنوع بین k حالت رنج ببرد. به سرعت تمامی آن‌ها در محدوده کوچکی از فضای حالت جمع شوند.

❖ جستجوی شعاع اتفاقی (Stochastic Beam Search)

❖ به جای انتخاب k بهترین از مجموعه تمام همسایه‌ها، k همسایه را به صورت تصادفی انتخاب می‌کند.
❖ احتمال انتخاب هر همسایه تابعی صعودی از میزان ارزش آن است.
❖ پیاده سازی؟

جستجوی محلی در محیط پیوسته

- ❖ روش‌های جستجوی قبلی در محیط گستته بودند
- ❖ محیط مسائل واقعی عمدتاً پیوسته است
- ❖ مشکل تعداد عمل/همسایه (ضریب انشعاب) بینهایت در محیط پیوسته
- ❖ ایده؟

مثال محیط پیوسته

- ❖ انتخاب محل تاسیس ۳ فرودگاه
- ❖ به طوری که مجموعه شهرهای فواصل اطراف به نزدیکترین فرودگاه، کمینه گردد.
- ❖ فضای حالت: $(x_1, y_1), (x_2, y_2), (x_3, y_3)$
- ❖ فضای ۶ متغیره
- ❖ فضای ۶ بعدی
- ❖ بردار ۶ بعدی = X
- ❖ حرکت در فضا = تغییر در یک یا چند متغیر از ۶ متغیر
- ❖ تابع هدف: $f(x_1, y_1, x_2, y_2, x_3, y_3)$

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

- ❖ C_i = مجموعه شهرهایی که نزدیکترین فرودگاه بدانها در وضعیت فعلی (مکان فعلی فرودگاه‌ها)، فرودگاه است.
- ❖ تابع فوق به صورت محلی درست است، اما به طور سراسری خیر: مجموعه C_i تابعی گستته بر حسب وضعیت است.

روش حل (پیوسته)

- ❖ راه اول: گسته سازی
 - ❖ هر بار فقط یکی از ابعاد یکی از فرودگاهها را به اندازه ای ثابت در جهت ثبت یا منفی جابجا می کنیم.
 - ❖ ۶ متغیر، هر یک دو انتخاب = ۱۲ وضعیت همسایه
 - ❖ حال می توان روش‌های گسته سازه قبلی را اعمال کرد.
- ❖ اعمال مستقیم (بدون گسته سازی) روش‌های تپه نوردی اتفاقی و شبیه سازی ذوب فلزات در محیط پیوسته
 - ❖ انتخاب تصادفی همسایه در این الگوریتم ها (در محیط گسته)
 - ❖ تولید تصادفی برداری با طول ثابت و جهت تصادفی (در محیط پیوسته)
- ❖ تپه نوردی اتفاقی (نسخه «انتخاب تصادفی همسایه بهتر») : پیاده سازی با نسخه «تپه نوردی اولین گزینه»

روش حل (پیوسته): بهینه سازی پیوسته

❖ استفاده از گرادیان (Gradient)

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

❖ در مسائل ساده : صفر کردن گرادیان \leftarrow جواب

❖ مثلا اگر فقط یک فرودگاه داشته باشیم

❖ در حالت سه فرودگاه، گرادیان وابسته به مجموعه شهرهای نزدیک به هر فرودگاه است.

❖ گرادیان محلی (و نه سراسری)

❖ با داشتن گرادیان محلی، الگوریتم تپه نوردی (در جهت بیشترین شبیب) قابل اعمال است:

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

❖ α = طول گام (ثابت؟)

❖ ممکن است مشتق تابع موجود نباشد

❖ مقدار تابع هدف برای مساله ۳ فرودگاه، با اعمال توابع شبیه ساز اقتصادی و توابع پیچیده دیگر به دست می آید

❖ استفاده از empirical gradient

❖ ارزیابی تابع هدف در مسیرهای کاهشی و افزایشی هر بعد از ابعاد فضای حالت

❖ با داشتن گرادیان محلی، الگوریتم تپه نوردی (در جهت بیشترین شبیب) قابل اعمال است:

❖ α = طول گام (ثابت؟)

❖ ممکن است مشتق تابع موجود نباشد

❖ مقدار تابع هدف برای مساله ۳ فرودگاه، با اعمال توابع شبیه ساز اقتصادی و توابع پیچیده دیگر به دست می آید

❖ استفاده از empirical gradient

❖ ارزیابی تابع هدف در مسیرهای کاهشی و افزایشی هر بعد از ابعاد فضای حالت

Line Search

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

❖ طول گام (α):

❖ کوچک \leftarrow تعداد گام زیاد

❖ بزرگ \leftarrow عبور از هدف

❖ الگوریتم Line Search

❖ طول گام را مرتبا بزرگتر می کند (۲ برابر)

❖ در جهت گرادیان حرکت می کند (بدون محاسبه مجدد گرادیان)

❖ تا وقتی که مقدارتابع هدف شروع به کم شدن کند.

❖ در نقطه مذکور مجددا گرادیان محاسبه می شود

❖ تکرار مراحل فوق

حضرت نیوتن را فسون! Newton-Raphson

❖ روشنی نیوتن:

❖ ایده: با یک گام برو به اکسترمم محلی محتمل!

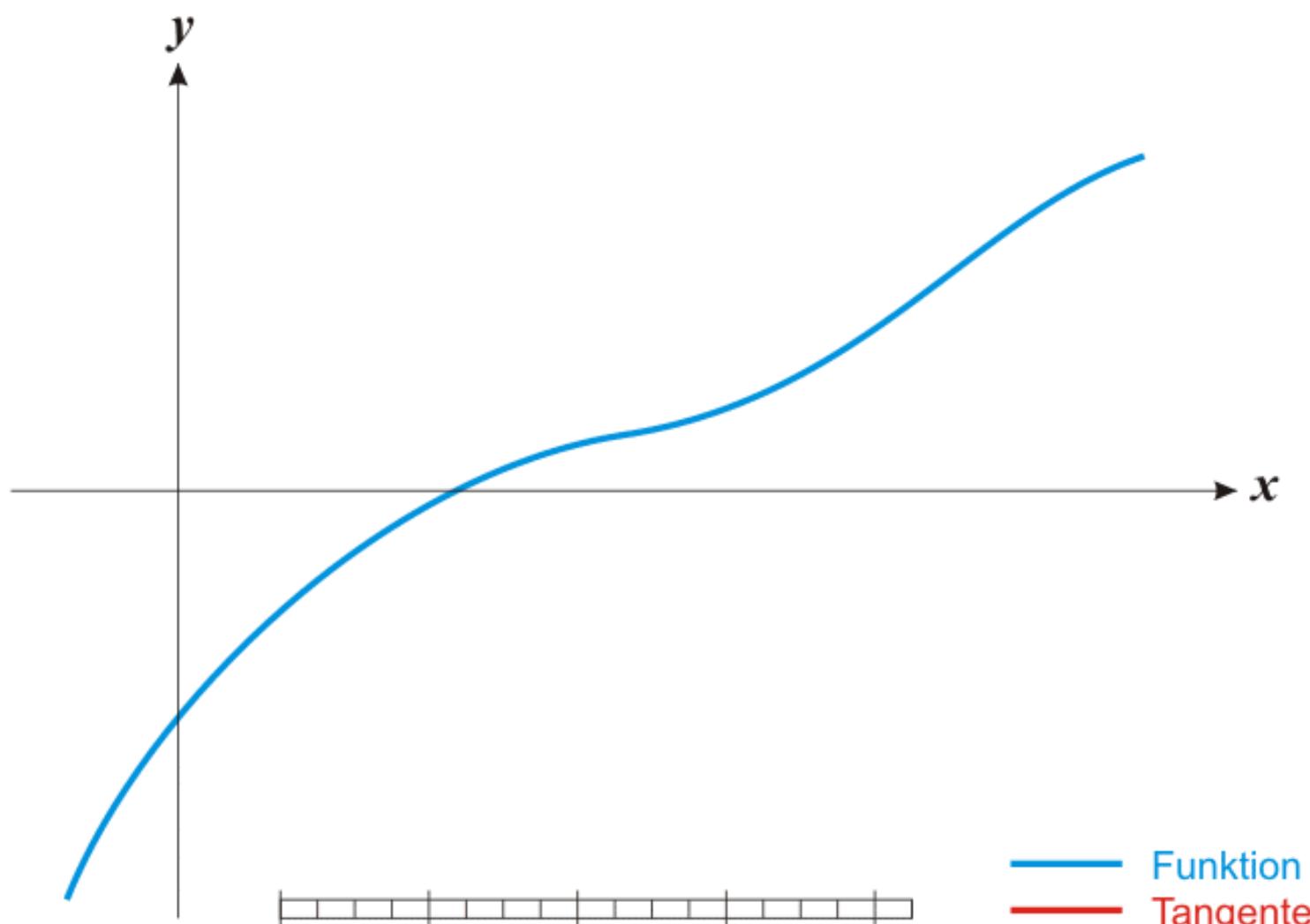
❖ کاربرد: یافتن ریشه توابع

❖ تخمین ریشه تابع طبق فرمول نیوتن

$$f(x) = f(x_0) + \frac{f'(x_0)(x - x_0)}{1!}$$

$$0 = f(x_0) + \frac{f'(x_0)(x - x_0)}{1!}$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$



$$f(x_k + t) \approx f(x_k) + f'(x_k)t + \frac{1}{2}f''(x_k)t^2.$$

❖ برای یافتن نقطه بیشینه/کمینه، ریشه مشتق تابع هدف را باید یافت

$$0 = \frac{d}{dt} \left(f(x_k) + f'(x_k)t + \frac{1}{2}f''(x_k)t^2 \right) = f'(x_k) + f''(x_k)t,$$

$$t = -\frac{f'(x_k)}{f''(x_k)}.$$

$$x_{k+1} = x_k + t = x_k - \frac{f'(x_k)}{f''(x_k)}$$

❖ فضای چند بعدی:

$$f'(x) = \nabla f(x) = g_f(x) \in \mathbb{R}^d$$

$$f''(x) = \nabla^2 f(x) = H_f(x) \in \mathbb{R}^{d \times d}$$

Hessian ❖

$$H_{ij} = \partial^2 f / \partial x_i \partial x_j$$

$$x_{k+1} = x_k - [f''(x_k)]^{-1} f'(x_k)$$

$$\mathbf{x} \leftarrow \mathbf{x} - \mathbf{H}_f^{-1}(\mathbf{x}) \nabla f(\mathbf{x})$$

$$H_{ij} = \partial^2 f / \partial x_i \partial x_j$$

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

❖ مساله فرودگاه ها:

❖ هر بار حرکت طبق روش نیوتن محل فرودگاه ارا به مرکز ثقل شهرهای Ci منتقل می کند.

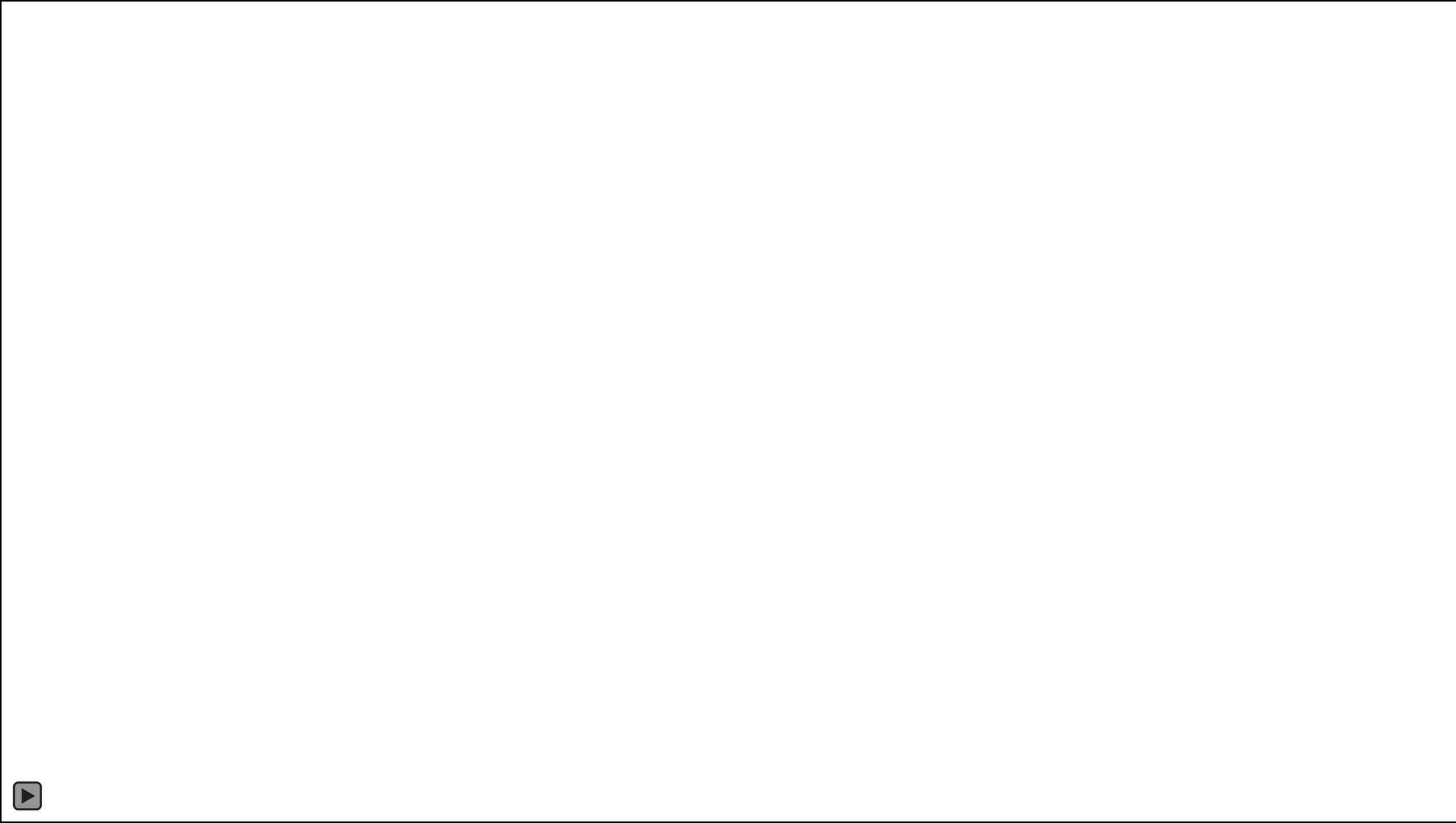
❖ به طور کلی، بروزرسانی طبق روش نیوتن، معادل آن است که یک ابرصفحه (سطح) درجه ۲ بر روی تابع f در نقطه فعلی fit شود (تابع f در آن نقطه به سطحی درجه ۲ تقریب زده شود) و سپس با یک گام حرکت، به نقطه کمینه آن سطح (نقطه مشتق صفر) منتقل شده مجددا سطح دیگری fit شود.

❖ مشکل: سختی محاسبه ماتریس H در ابعاد بالا و نیز محاسبه معکوس آن

❖ ورژنهای تقریب زننده نیوتن

جستجوی محلی در مقابل جستجوی سیستماتیک

جستجوی محلی	جستجوی سیستماتیک	
خود وضعیت به عنوان راه حل برگردانده می شود.	مسیر از وضعیت اولیه به هدف	راه حل
یک یا چند وضعیت فعلی را نگه داری می کند و سعی می کند آنها را بهبود دهد.	به طور سیستماتیک مسیرهای مختلف را امتحان می کند.	روش
پیکربندی کامل (Complete)	معمولًا افزایشی (Incremental)	فضای حالت
معمولًا خیلی کم (ثابت)	معمولًا خیلی بالا	حافظه
پیدا کردن راه حل های معقول در فضاهای حالت بزرگ یا نامتناهی (پیوسته)	پیدا کردن راه حل های بهینه در فضاهای حالت کوچک	اندازه فضا
جستجو و مسائل بهینه سازی	جستجو	حوزه



پردازش تکاملی (Evolutionary Computing)

- ❖ براساس ایده‌ی تکامل طبیعی و فرضیه‌ی داروین (قرن ۱۹ میلادی)
- ❖ در هر جامعه‌ای، معمولاً افراد قوی‌تر می‌توانند بیشتر از منابع استفاده کنند و با احتمال بیشتری زنده می‌مانند، اما افراد ضعیفتر با احتمال کمتری باقی می‌مانند.
- ❖ به دلیل محدودیت منابع زیستی (خوراک و)، ممکن است امکان بقای همه موجودات فراهم نباشد و تنابع بقا سبب حذف ضعیفترها شود.
- ❖ افراد باقی‌مانده از هر نسل یک جامعه، تولید مثل کرده و فرزندانی تولید می‌کنند که نسل بعد را تشکیل می‌دهند.
- ❖ هر فرزند خصوصیات خود را از والدینش به ارث می‌برد.
- ❖ انتظار داریم افراد هر نسل از جامعه، قوی‌تر از افراد نسل قبل خود باشند.
- ❖ ممکن است جهش ژنتیکی رخ داده باشد و یکی از ژن‌های فرد به طور تصادفی تغییر کند.
- ❖ باعث تنوع و پراکندگی در افراد یک نسل می‌شود.

❖ تقابل فرضیه تکامل انواع و ثبوت انواع

❖ امکان اثبات فرضیه تکامل

❖ عقلی و فلسفی؟

❖ تجربی (مشاهده و تجربه)؟

انواع پردازش‌های تکاملی

Genetic Algorithm (GA) ❖

❖ بازنمایی فضای حلت با آرایه ای با طول ثابت از بیت‌ها (یا integer)

Crossover ❖ سادگی

Genetic Programming (GP) ❖

❖ بازنمایی با ساختار درخت (اندازه متغیر)

Evolutionary Programming (EP) ❖

❖ بازنمایی گرافی

Evolutionary Strategy (ES) ❖

❖ بازنمایی با اعداد حقیقی (پیوسته)

❖ طول جهش خود تطبیقی برای هر فرد

❖ جستجوی دسته جمعی پرندگان / مورچه‌ها / زنبورها / سایر جک و جانورها (!)

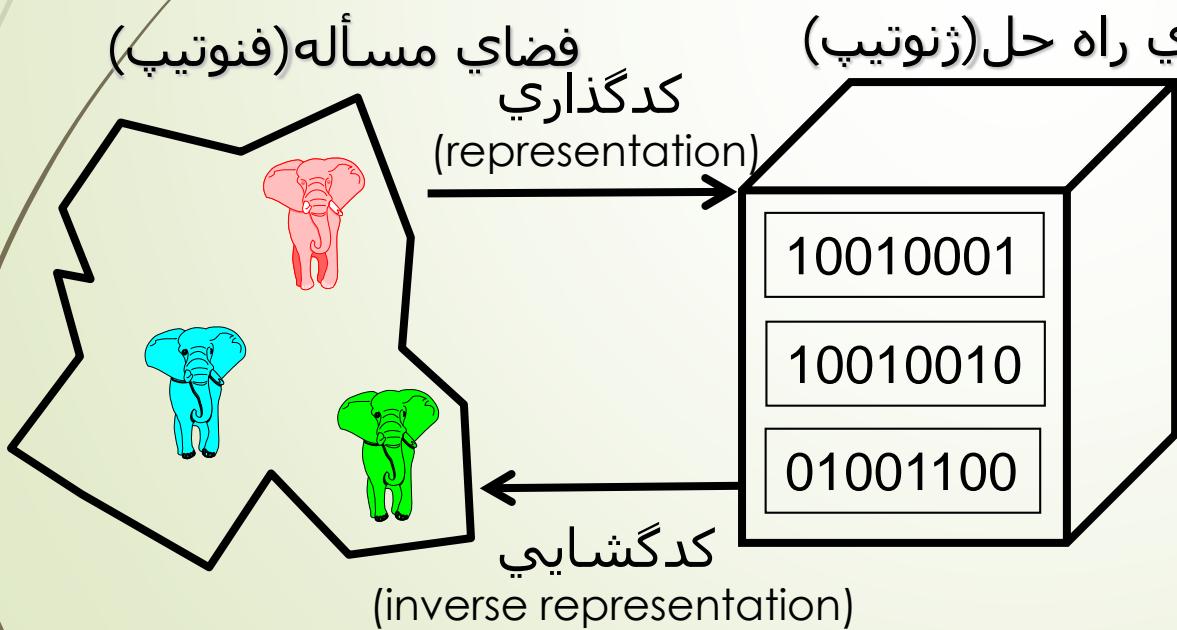
❖ در عمل: ترکیب ایده‌های همه الگوریتم‌ها ممکن است و اتفاق می‌فتند.

الگوریتم ژنتیک

- ❖ گونه‌ای از جستجوی شعاع اتفاقی
- ❖ به جای تولید حالت‌های پسین با تغییر یکی از حالات، پسین‌ها از ترکیب دو حالت والد به دست می‌آیند.
- ❖ چارچوب کلی یک الگوریتم ژنتیک (و همه الگوریتم‌های تکاملی)
 - ۱) تعیین روش بازنمایی و تابع برازش (*fitness*)
 - ۲) ایجاد جمعیت اولیه
 - ۳) تکرار تا برقراری شرط خاتمه
 - ۱-۳) ارزیابی جمعیت
 - ۲-۳) انتخاب والدین
 - ۳-۳) اعمال عملگرهای ژنتیکی و تولید فرزندان
 - ۴-۳) انتخاب بازماندگان

تعیین روش بازنمایی

- ❖ هر وضعیت از مسئله را باید به صورت مناسبی مدل و ذخیره کرد.
- ❖ در فضای حالت گسترش: آرایه‌ای از صفرها و یک‌ها، آرایه‌ای از اعداد صحیح
- ❖ فنوتیپ (*phenotype*): هر یک از وضعیت‌های واقعی مسئله
- ❖ ژنوتیپ (*genotype*) یا کروموزوم: هر یک از آرایه‌های مدل‌کننده هر وضعیت
- ❖ ژن: هر یک از فیلدهای یک کروموزوم
- ❖ برای آن که الگوریتم امکان یافتن جواب بهینه را داشته باشد روش بازنمایی باید تمام راه حل‌های ممکن را پوشش دهد.



تعیین تابع برازش

- ❖ هر حالت توسط تابع ارزیاب یا تابع برازش رتبه‌بندی می‌شود.
- ❖ یک تابع برازش باید برای حالت‌های بهتر، مقادیر بزرگ‌تر را برگرداند.
- ❖ هر چه یک کروموزوم به وضعیت هدف نزدیک‌تر باشد، برازنده‌گی آن بیشتر خواهد بود.

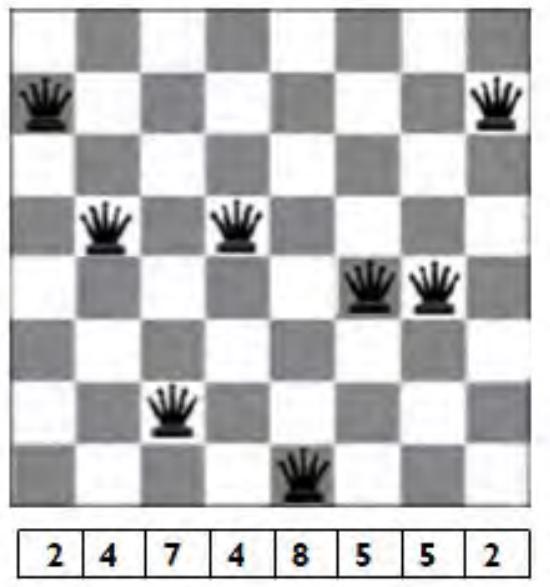
❖ مثال ۸- وزیر

❖ بازنمایی: آرایه‌ای یک بعدی از ۸ المان که المان آم آن نشان‌دهنده‌ی سطر وزیر ستون آم است.

❖ تابع برازش حالت n : تعداد زوج وزیرهایی که در وضعیت n هم‌دیگر را تهدید نمی‌کنند.

❖ در حالت نشان داده شده برابر با ۲۴ است.

❖ ماکزیمم برازنده‌گی (برازندگی وضعیت هدف)?

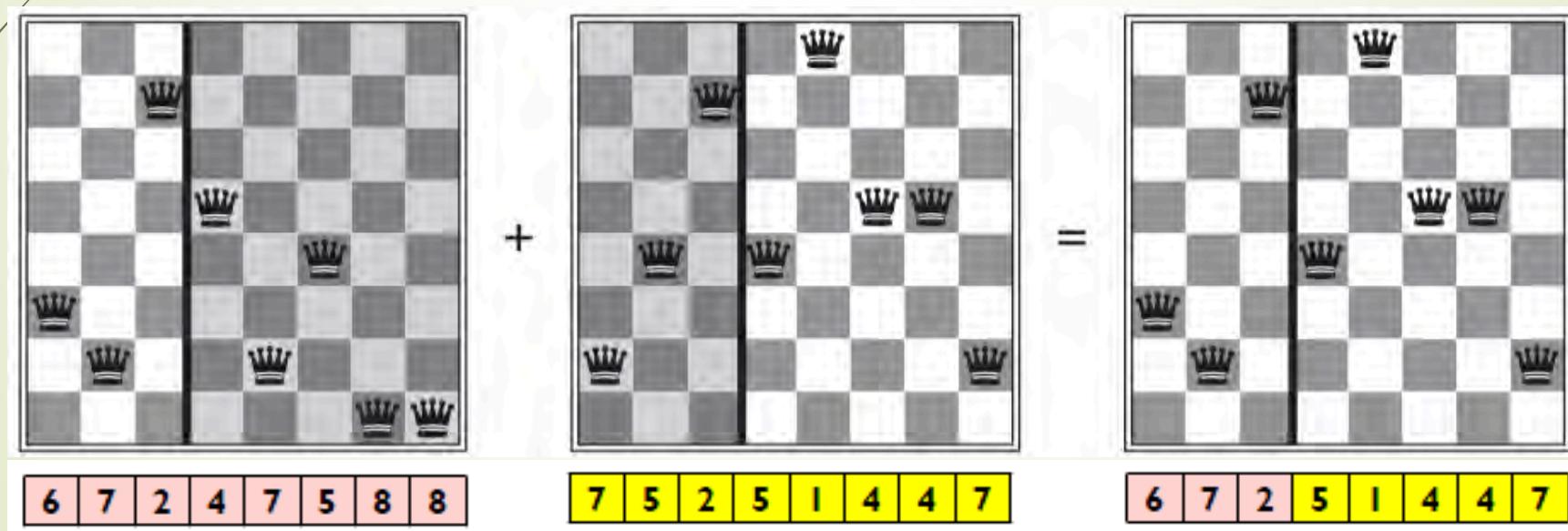


الگوریتم ژنتیک

- ۱- k وضعیت را به طور تصادفی، به عنوان کروموزومهای نسل فعلی در نظر بگیر.
- ۲- تا وقتی که هیچ یک از افراد نسل فعلی، معادل هدف نیست مراحل زیر را تکرار کن. (یا هر شرط منطقی دیگر برای خاتمه حلقه)
 - ۱-۱- مقدار برازنده‌گی هر یک از کروموزومهای نسل فعلی را محاسبه کن.
 - ۲-۲- به هر یک از کروموزومهای نسل فعلی، با توجه به مقدار برازنده‌گی اش، یک احتمال انتخاب نسبت بدده، به طوری که کروموزوم با برازنده‌گی بیشتر، احتمال انتخاب بیشتری داشته باشد.
 - ۳-۲- کروموزوم را به طور تصادفی و با توجه به احتمال انتخاب‌شان، انتخاب کن (عمل selection). یک کروموزوم چند بار می‌تواند انتخاب شود.
 - ۴-۲- کروموزومهای انتخاب شده را دو به دو با هم ترکیب کن و از ترکیب هر دو تای آنها، دو فرزند در نسل بعدی اضافه کن (عمل crossover یا ترکیب متقطع).
 - ۵- ۲- برخی ژن‌های فرزندان را به طور تصادفی تغییر بده (عمل mutation یا جهش).
 - ۶- ۲- فرزندان تولید شده را به عنوان نسل فعلی در نظر بگیر و حلقه را تکرار کن.

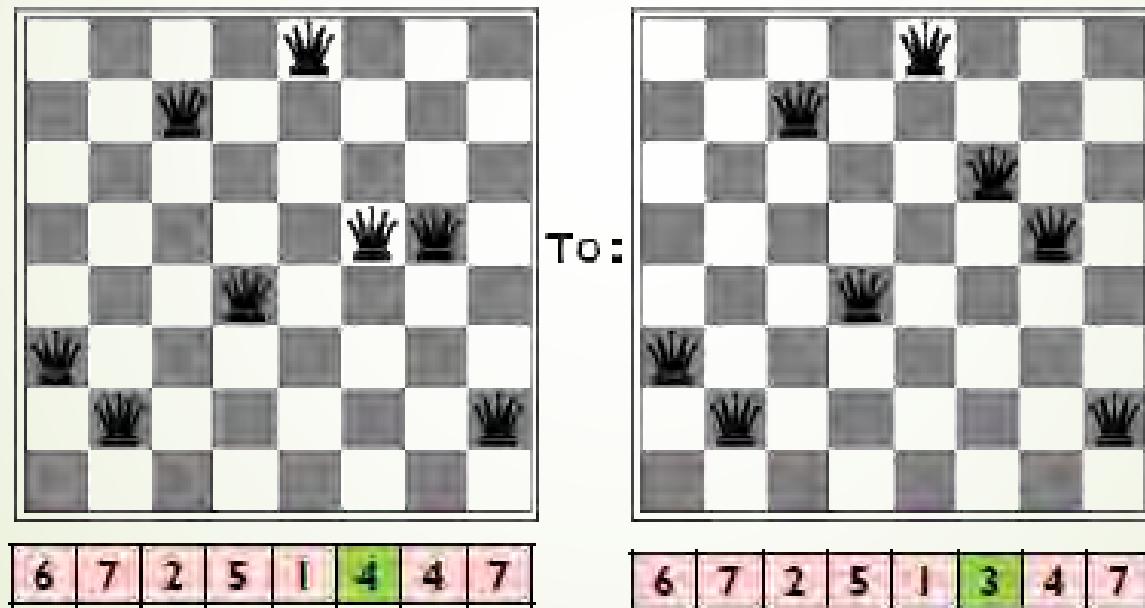
عمل ترکیب متقارطع (Crossover)

- ❖ دو کروموزوم والد به روش‌های مختلفی می‌توانند با هم ترکیب شوند و دو فرزند تولید کنند.
- ❖ ترکیب یک نقطه‌ای
- ❖ انتخاب یک نقطه از موقعیت‌های داخل کروموزوم‌ها به‌طور تصادفی
- ❖ تولید فرزندان از برخورد کروموزوم‌های والد در نقطه‌ی پیوند

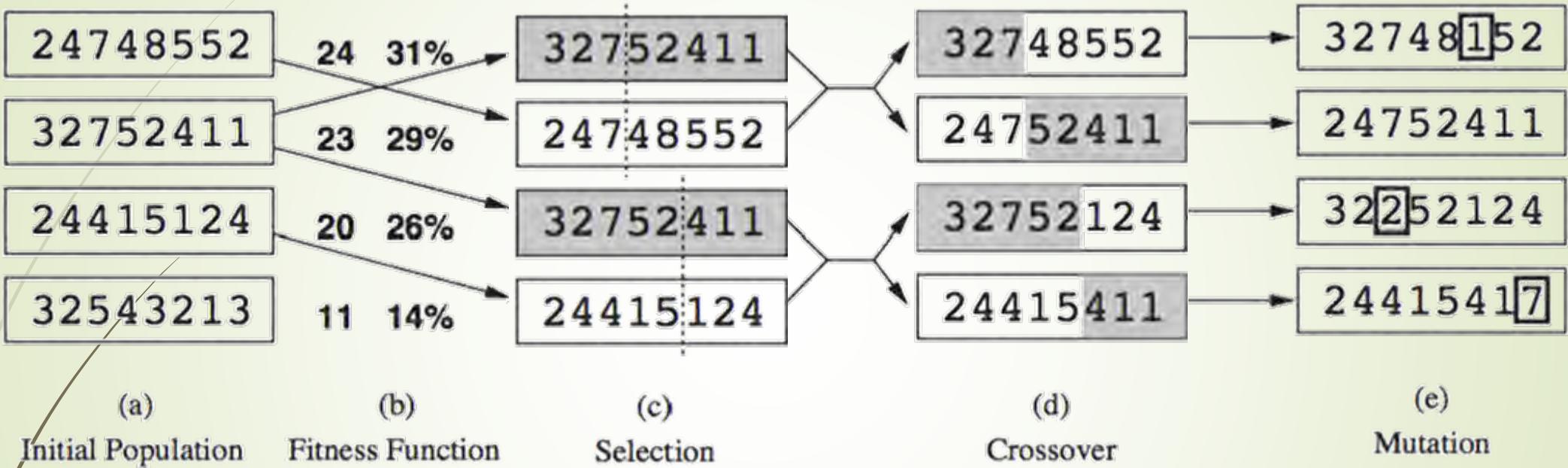


عمل جهش (Mutation)

- با یک احتمال مشخص، امکان دارد هر کروموزوم مورد جهش واقع شود.
- هم زن و هم مقدار جدید آن به طور تصادفی انتخاب می‌شوند.



چرخه‌ای از الگوریتم ژنتیک برای مسئله ۸ وزیر



$$fitness(i) / \sum_{k=1}^n fitness(k)$$

❖ احتمال انتخاب کروموزوم i برابر است با

$$\%31 = 24/(11+20+23+24)$$

$$\%29 = 23/(11+20+23+24)$$

الگوریتم ژنتیک ...

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 \leftarrow empty set

for *i* = 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 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*))

ویژگی‌های الگوریتم ژنتیک (و سایر الگوریتم‌های تکاملی)

- ❖ الگوریتم‌های ژنتیک معمولاً در نسل‌های ابتدایی، گام‌های بزرگ برداشته و در نسل‌های بعدی گام‌های کوچک‌تر برمی‌دارند.
- ❖ در ابتدا معمولاً، افراد جامعه پراکندگی خوبی دارند.
- ❖ عملگر ترکیب بر روی وضعیت‌های والد مختلف، می‌تواند حالتی متفاوت نسبت به هر دو والد تولید کند.
- ❖ به تدریج، افراد با شباهت بیشتر در جمعیت ظهرور می‌یابند.
- ❖ عملگر ترکیب به عنوان ویژگی مهم الگوریتم ژنتیک شناخته شده است.
- ❖ قابلیت ترکیب بلوک‌های بزرگ از ژن‌ها را دارد که به طور مستقل تکامل یافته‌اند.
- ❖ بازنمایی نقش مهمی در سودمندی عملگر ترکیب مورد استفاده دارد.