



دانشگاه صنعتی امیرکبیر

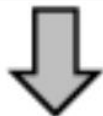
دانشکده مهندسی کامپیوتر و فناوری اطلاعات

الگوریتم‌های جستجوی محلی و مسائل بهینه‌سازی

«هوش مصنوعی: یک رهیافت نوین»، فصل ۴

ارائه‌دهنده: سیده فاطمه موسوی

نیم‌سال دوم ۱۴۰۰-۱۳۹۹



- آن چه 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 ۸ تایی و ...

- در بسیاری از مسائل، مسیر راه‌حل (یا نحوه‌ی رسیدن به هدف) اهمیت ندارد؛ **خود حالت هدف** پاسخ مسئله است.

- مانند مسئله n -وزیر

- ارتباط با بهینه‌سازی

- تعریف مسائل جستجوی فصل قبل به صورت مسئله بهینه‌سازی

- در این گونه مسائل: فضای حالت = مجموعه پیکربندی‌های کامل

هدف = یافتن یک پیکربندی که محدودیت‌های مسئله را ارضاء کند

- در چنین مواردی می‌توان از **الگوریتم‌های جستجوی محلی** بهره گرفت.

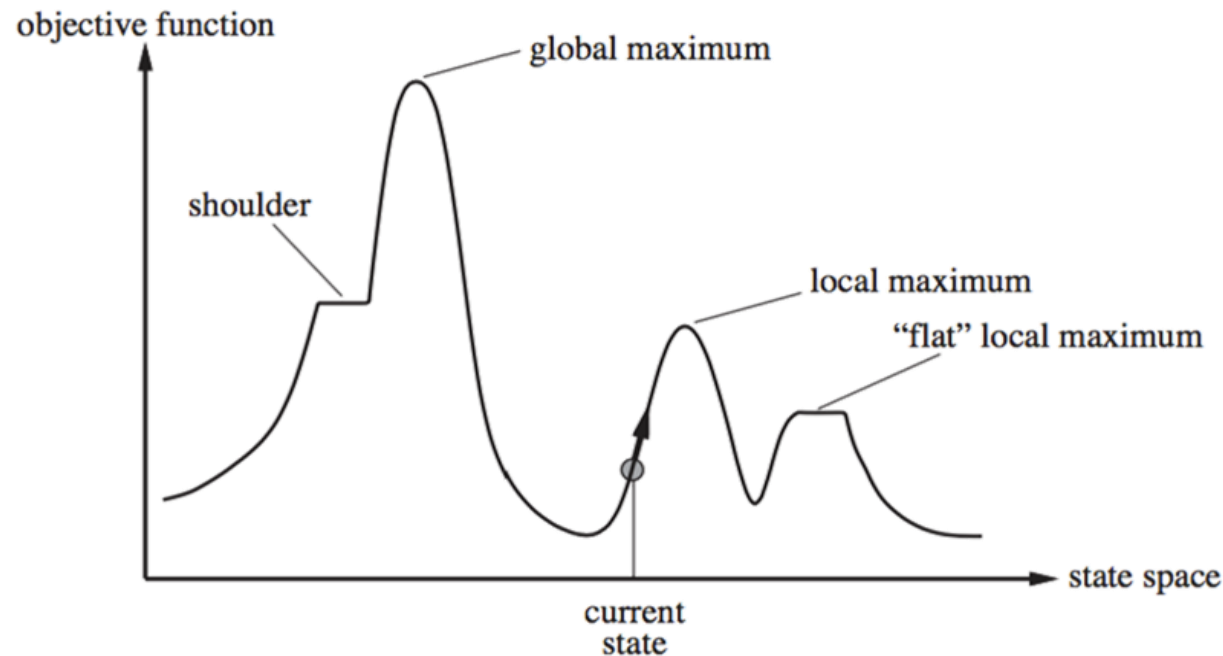
- معرفی جستجوهای محلی
 - جستجوی تپهنوردی (Hill-climbing search)
 - شبیه‌سازی ذوب فلزات (Simulated annealing)
 - جستجوی پرتو محلی (Local beam search)
 - الگوریتم‌های ژنتیک (Genetic algorithms)

معرفی جستجوی محلی

- جستجوی محلی یک (یا چند) حالت را به عنوان حالت فعلی در نظر می گیرد و تنها به حالت های همسایه ی آن انتقال می یابد.
- یک حالت “فعلی” را به تنهایی 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 الگوریتم ها مسیر طی شده را ذخیره نمی کنند.
 - یافتن راه حل های معقول در اغلب موارد در فضا های حالت بزرگ و یا نامحدود
 - مفید برای مسائل بهینه سازی محض
 - یافتن بهترین حالت بر طبق تابع هدف (objective function)

دورنمای فضای حالت (State Space Landscape)

- الگوریتم‌های جستجوی محلی landscape را کاوش می‌کنند.
- دورنما شامل «محل» (حالت) و «ارتفاع» (مقدار تابع هیوریستیک هزینه یا تابع هدف) است.
- اگر ارتفاع متناظر با هزینه باشد، آن گاه هدف یافتن عمیق‌ترین دره (یک کمینه سراسری) است.
- اگر ارتفاع متناظر با تابع هدف باشد، آن گاه هدف یافتن بلندترین قله (یک بیشینه سراسری) است.



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

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.
- ساختمان داده گره 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)

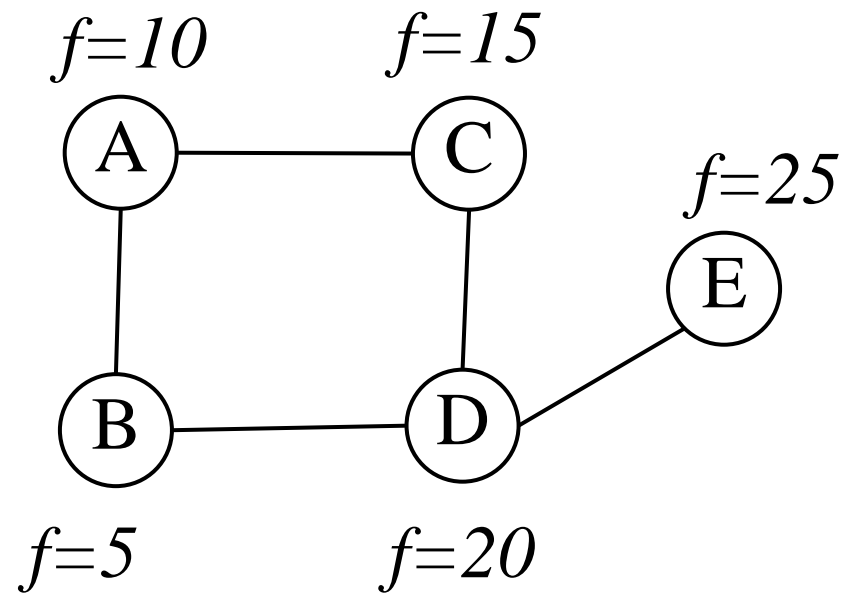
loop do

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

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

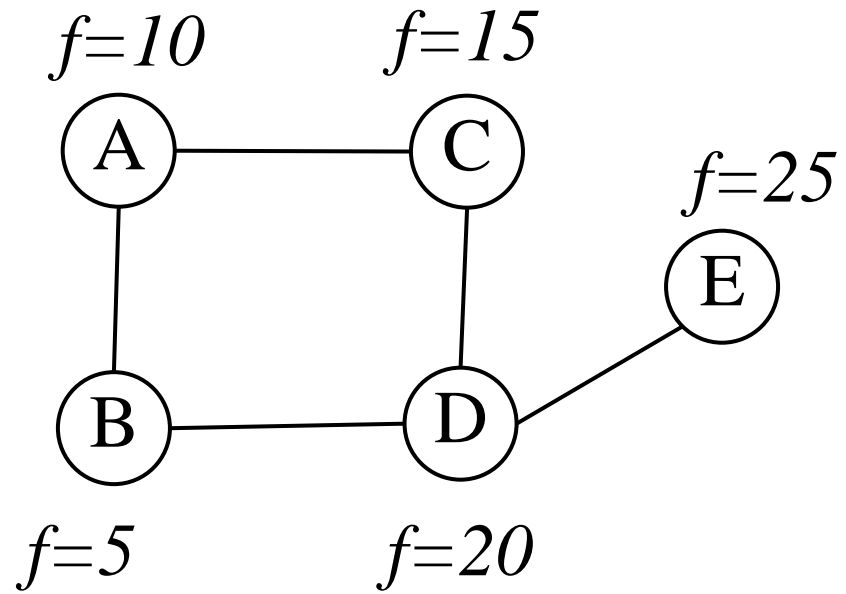
current \leftarrow *neighbor*

جستجوی تپه نوردی - مثال



current = A

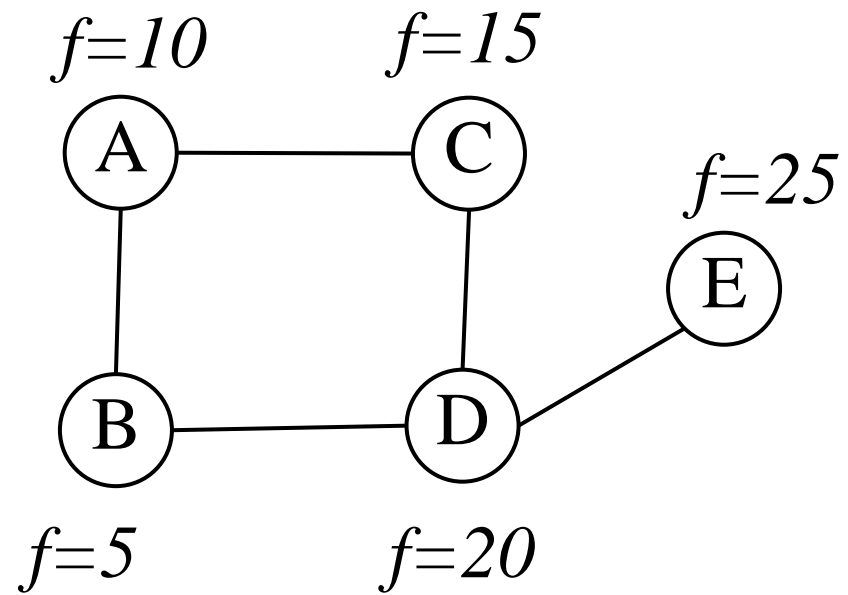
جستجوی تپه نوردی - مثال



current = A

current = C

جستجوی تپه نوردی - مثال

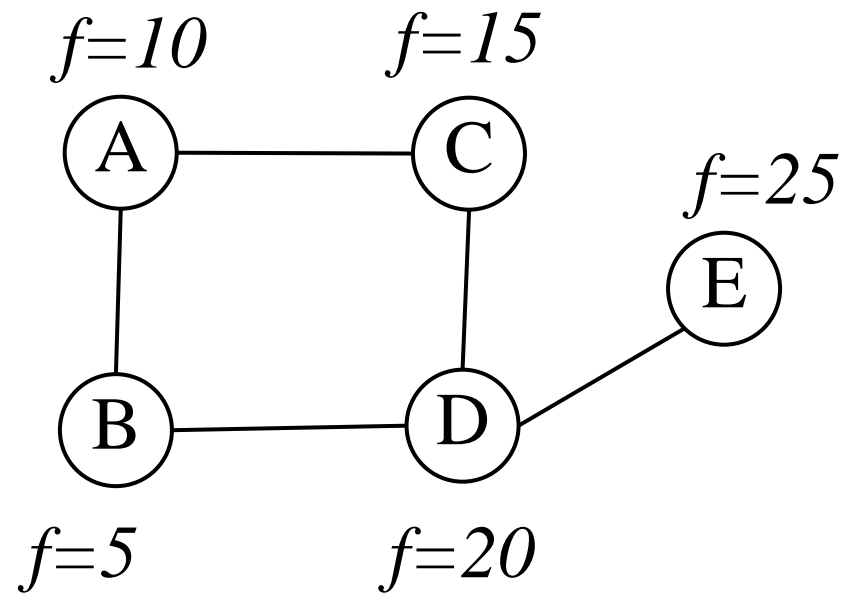


current = A

current = C

current = D

جستجوی تپه نوردی - مثال



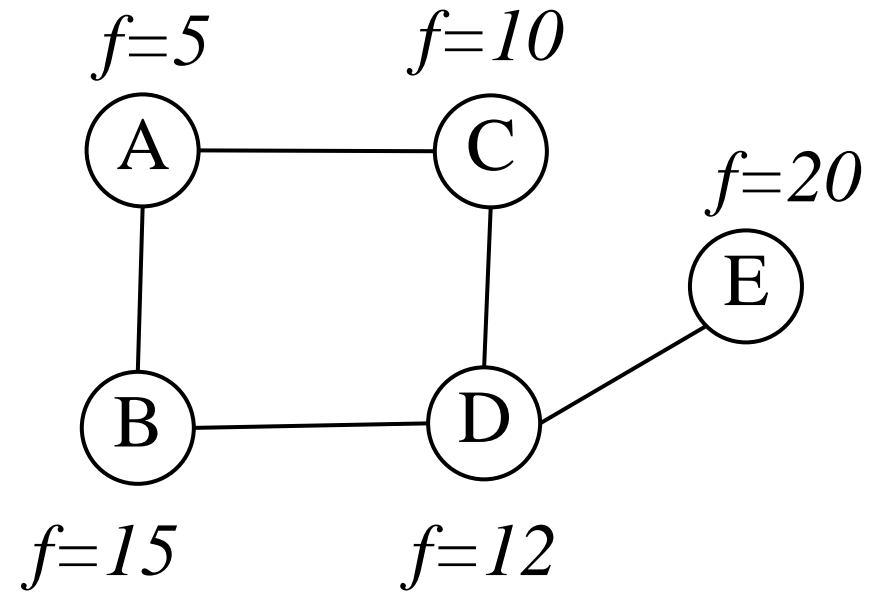
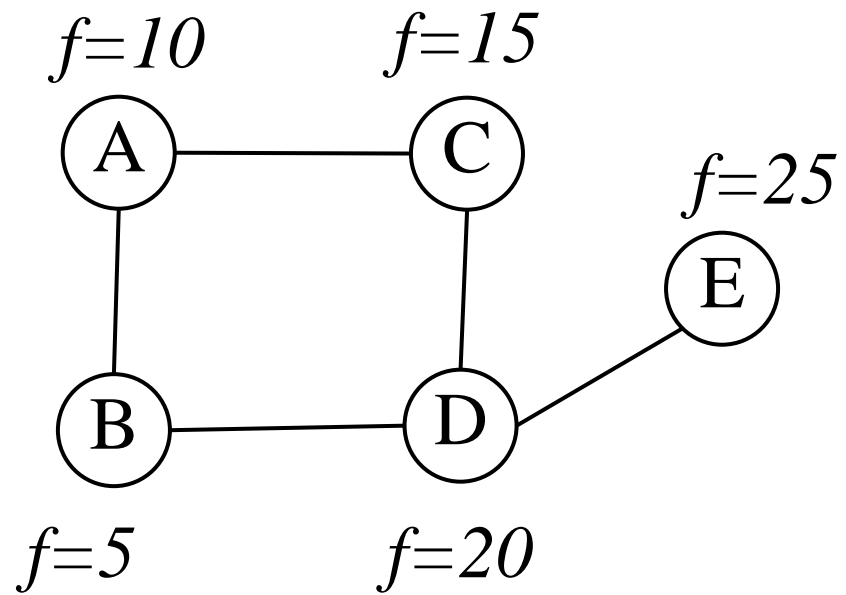
current = A

current = C

current = D

current = E

جستجوی تپه نوردی - مثال



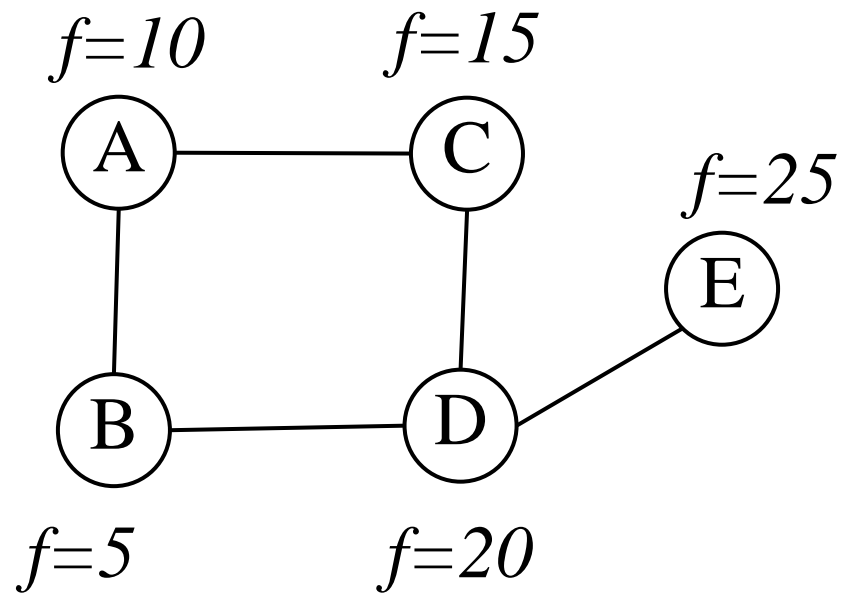
current = A

current = C

current = D

current = E

جستجوی تپه نوردی - مثال

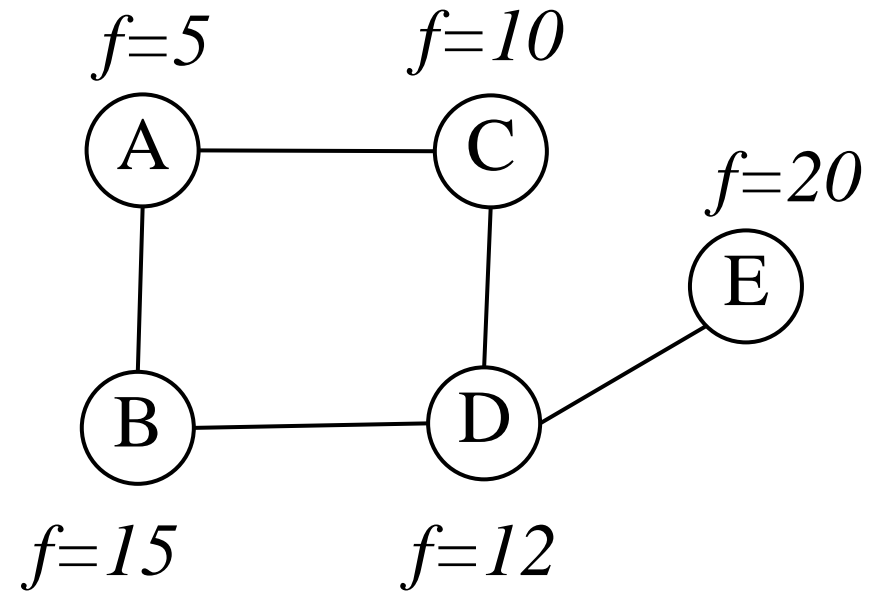


current = A

current = C

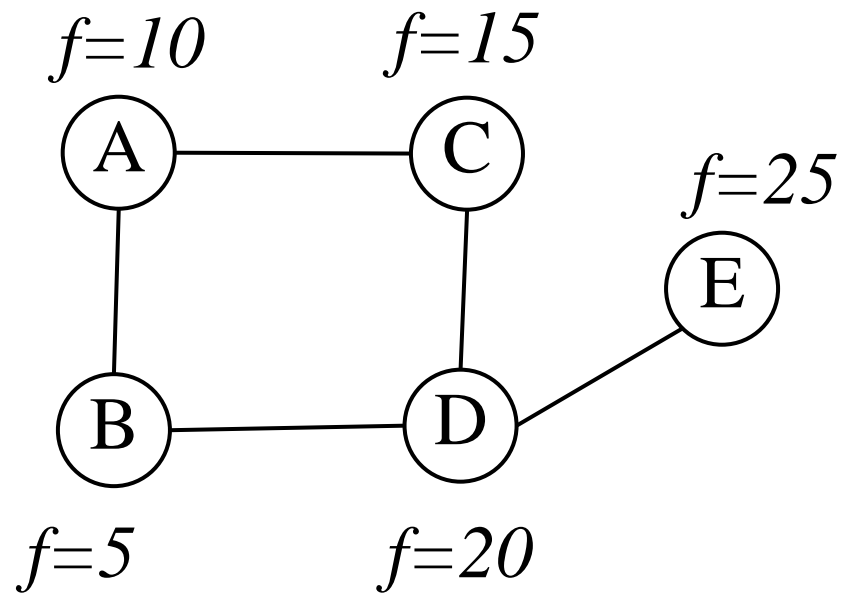
current = D

current = E



current = A

جستجوی تپه نوردی - مثال

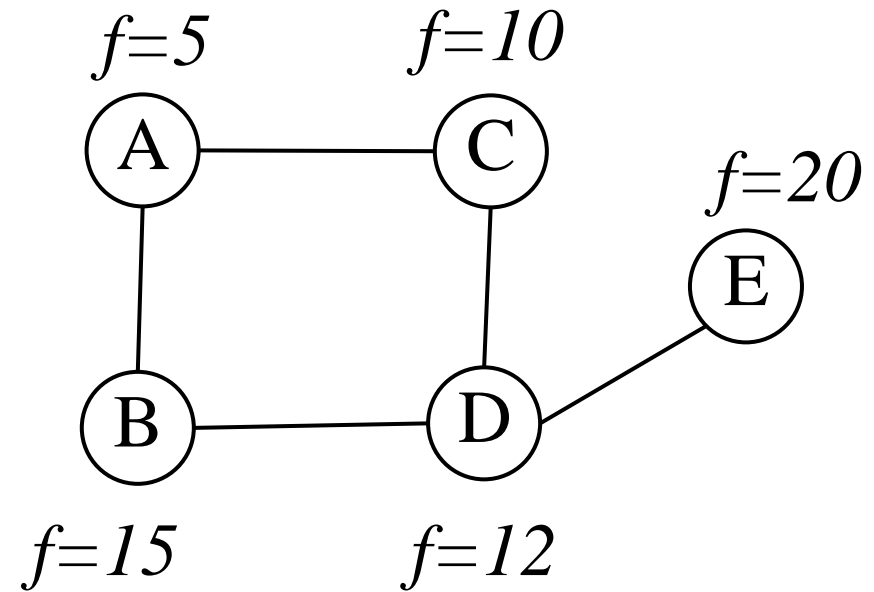


current = A

current = C

current = D

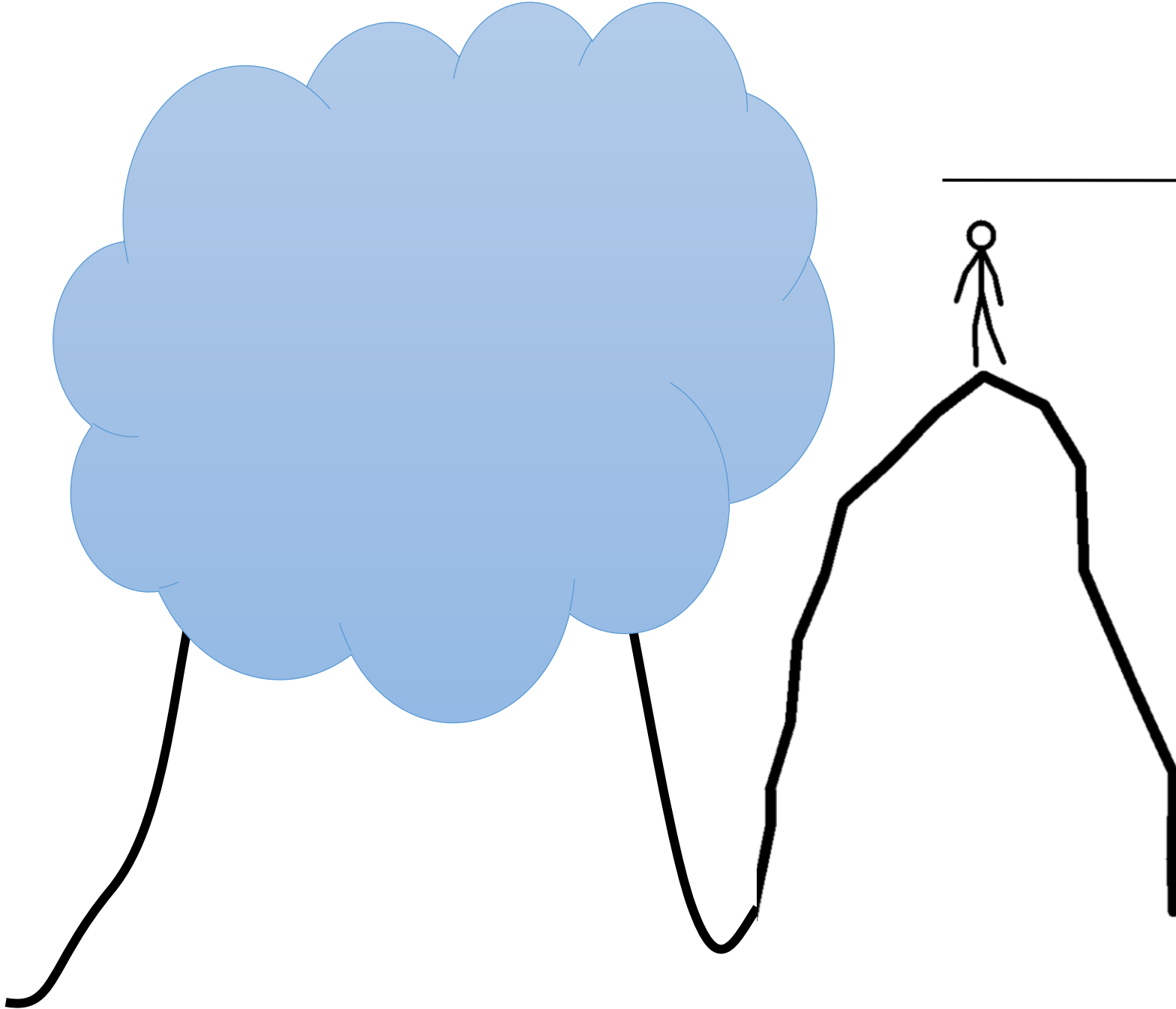
current = E



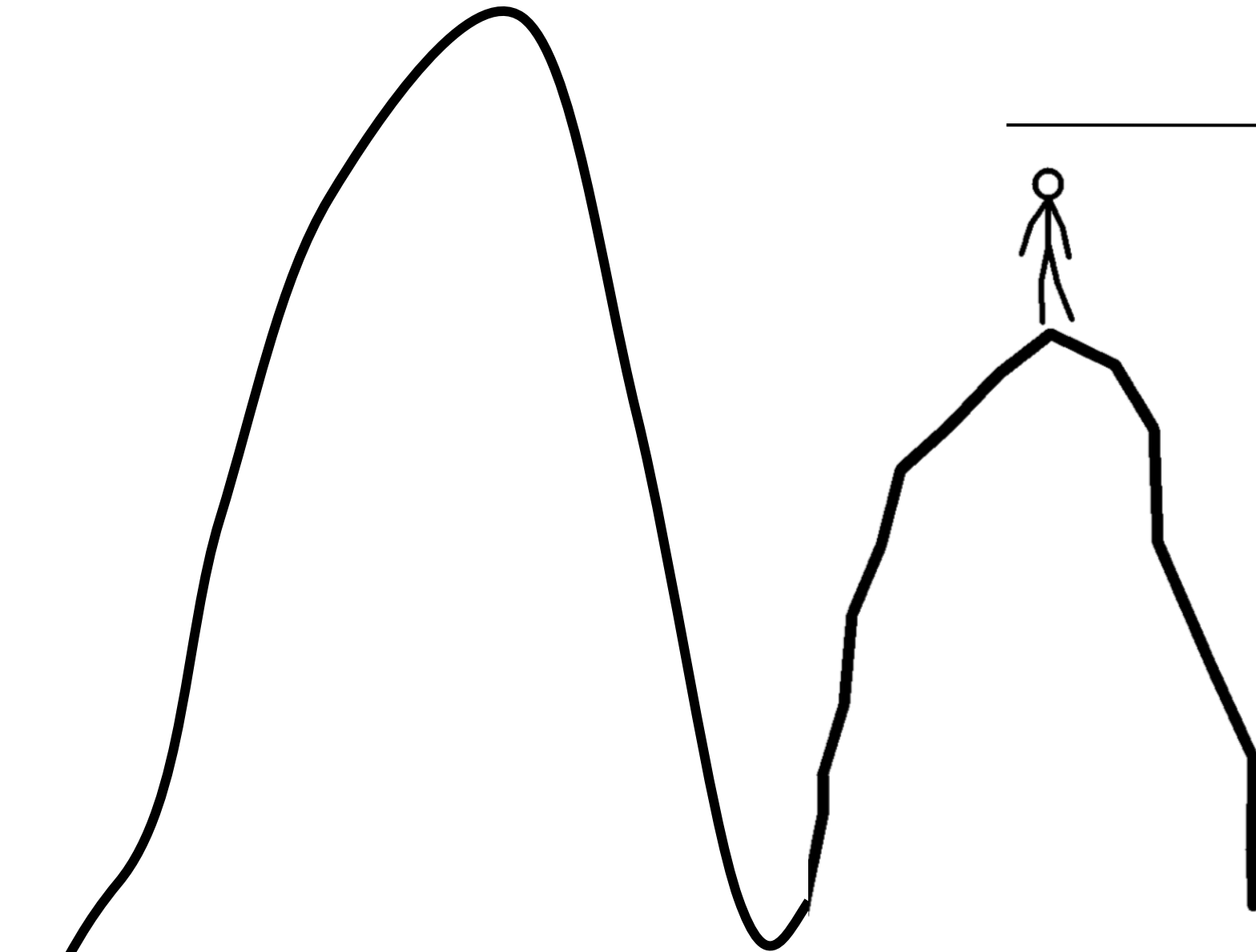
current = A

current = B

جستجوی تپه نوردی



جستجوی تپه نوردی

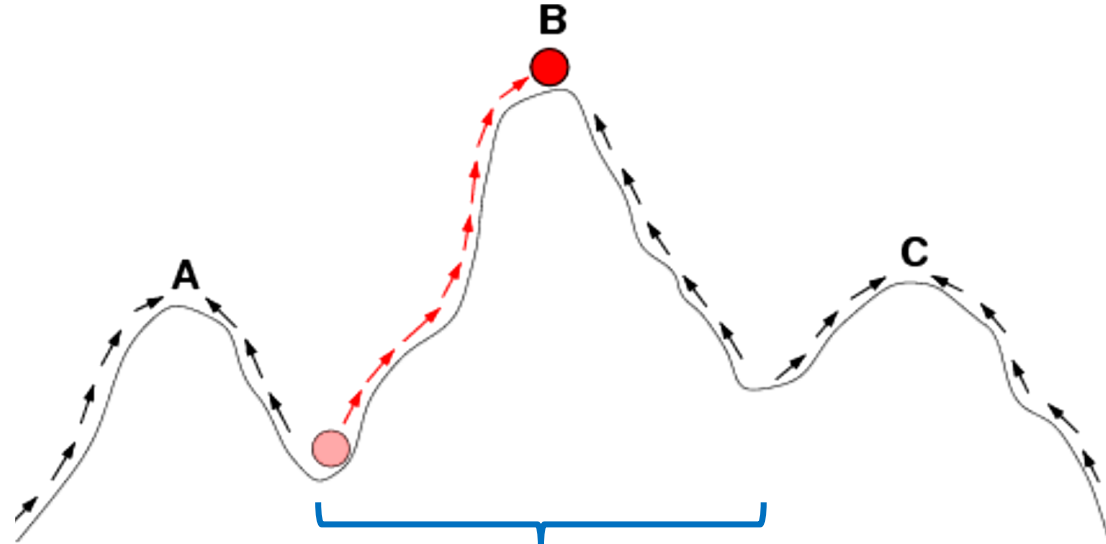


جستجوی تپه‌نوردی ...

- یک جستجوی محلی حریصانه است زیرا تپه‌نوردی، 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 $n \times n$ به گونه‌ای قرار بده که هیچ دو وزیری در یک سطر، ستون و یا قطر قرار نگیرند.

• ۸- وزیر

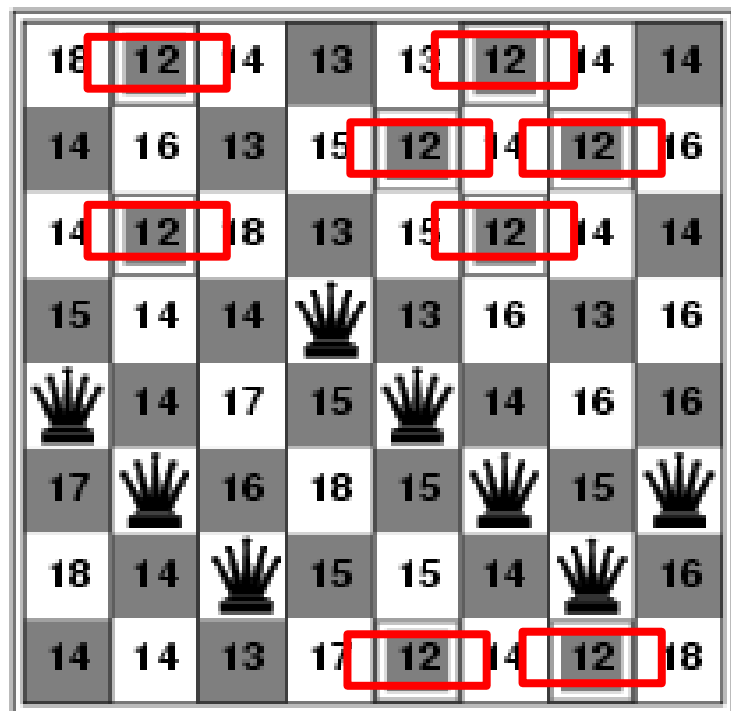
• فرمول‌بندی حالت کامل: هر حالت شامل ۸ وزیر یعنی یکی در هر ستون (8^8 حالت)

• پسین‌های یک حالت، همه حالت‌های ممکن است که با جابه‌جایی یک وزیر به مربع دیگری در همان ستون تولید می‌شود. هر حالت $8 \times 7 = 56$ پسین دارد.

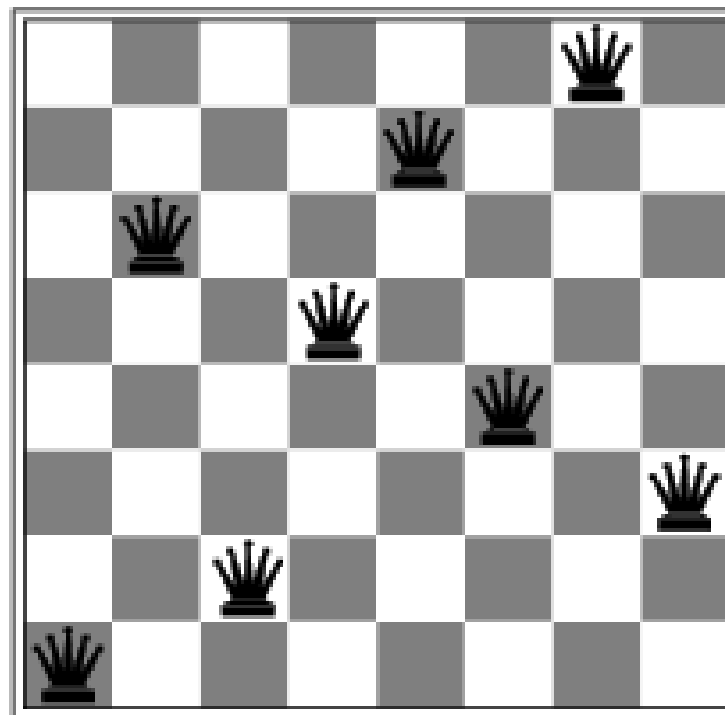
• تابع هیوریستیک h تعداد جفت وزیرهایی است که چه به صورت مستقیم و چه غیر مستقیم در حال حمله به یکدیگر هستند.

• کمینه سراسری این تابع صفر است که فقط در راه حل کامل اتفاق می‌افتد.

جستجوی تپه‌نوردی – مثال ...



$h=17$



$h=1$



$h>1$

بهترین پسین دارای $h=12$ است.

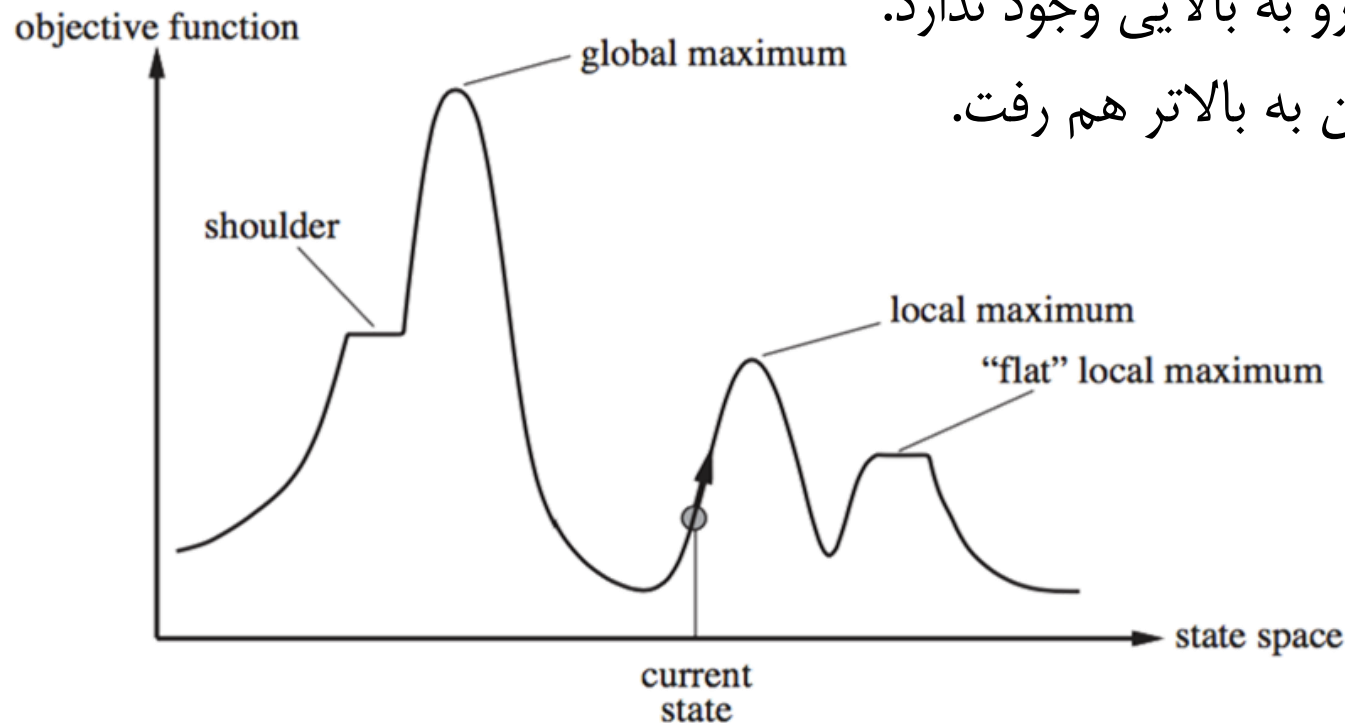
مشکلات جستجوی تپه‌نوردی

- **بیشینه‌های محلی (Local Maxima):** قله‌ای که از تمامی حالات همسایگانش بلندتر باشد اما از بیشینه‌ی سراسری پایین‌تر باشد.

- **فلات‌ها (Plateau):** ناحیه‌ای از دورنمای فضای حالت که در آن تابع ارزیاب، ثابت است.

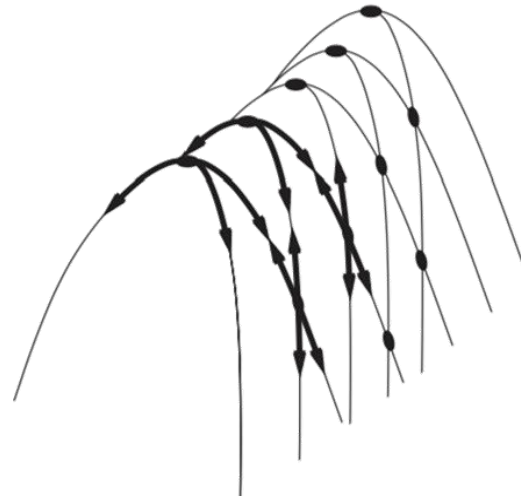
- بیشینه‌ی محلی مسطح: هیچ مسیر رو به بالایی وجود ندارد.

- شانه (shoulder): می‌توان از آن به بالاتر هم رفت.



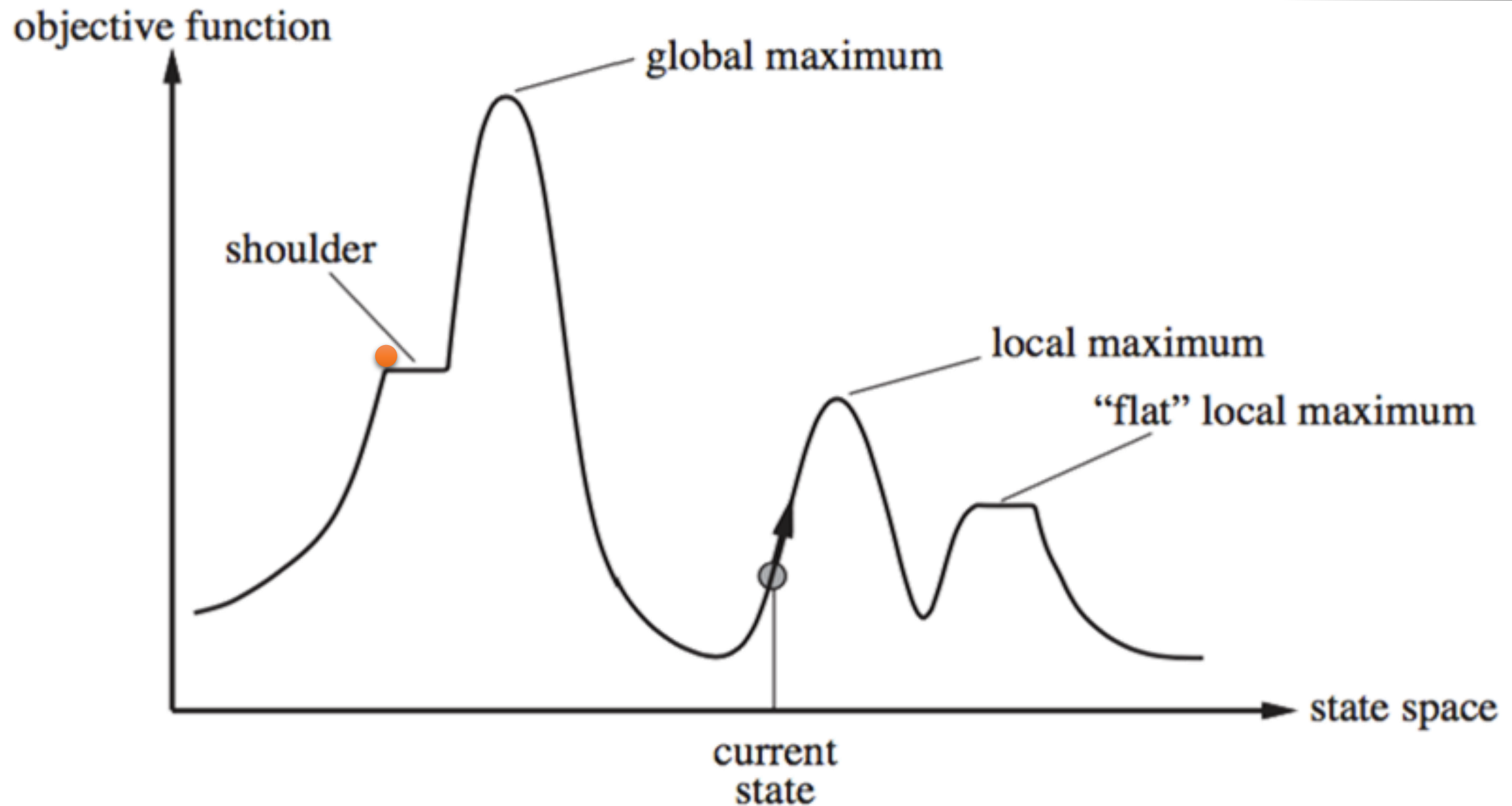
مشکلات جستجوی تپه‌نوردی ...

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

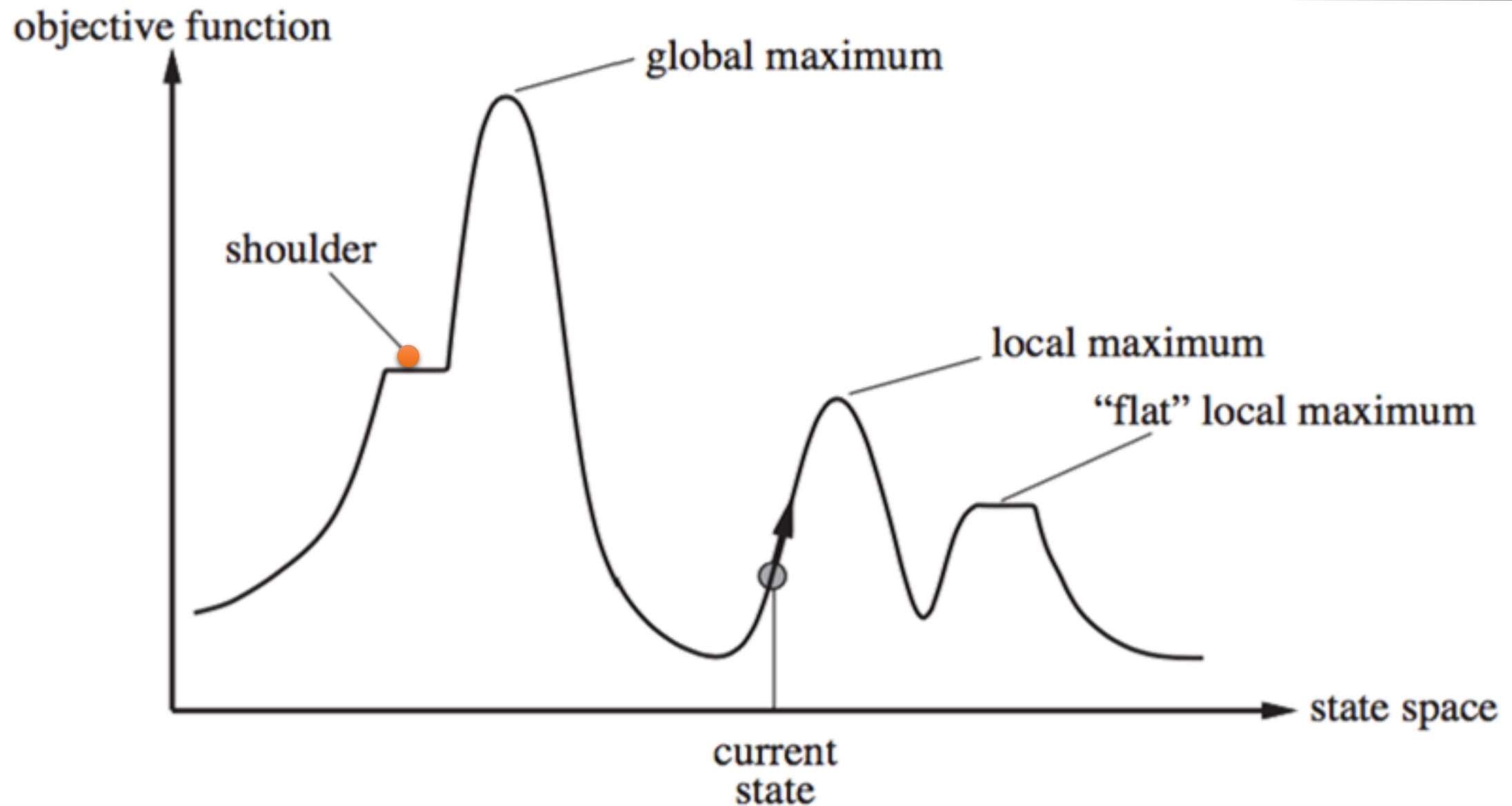


- با شروع از یک حالت اولیه‌ی ۸ وزیر که به‌طور تصادفی تولید شده است، تپه‌نوردی در ۸۶ درصد اوقات گیر می‌کند.
- با این حال در موارد موفق، به‌طور متوسط تنها ۴ گام و در مواردی که گیر می‌کند تنها ۳ گام برمی‌دارد.

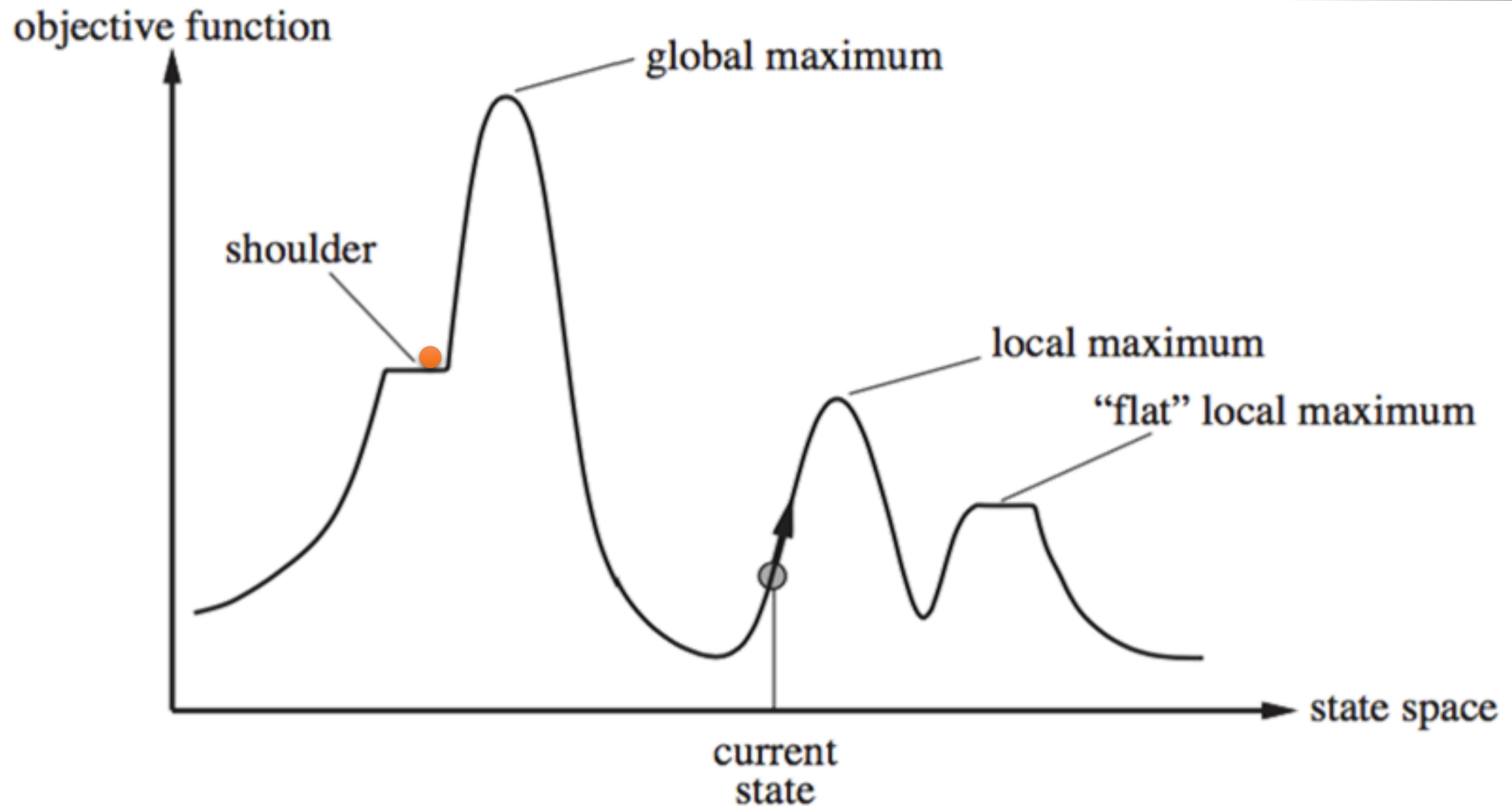
نسخه‌های مختلف جستجوی تپه‌نوردی



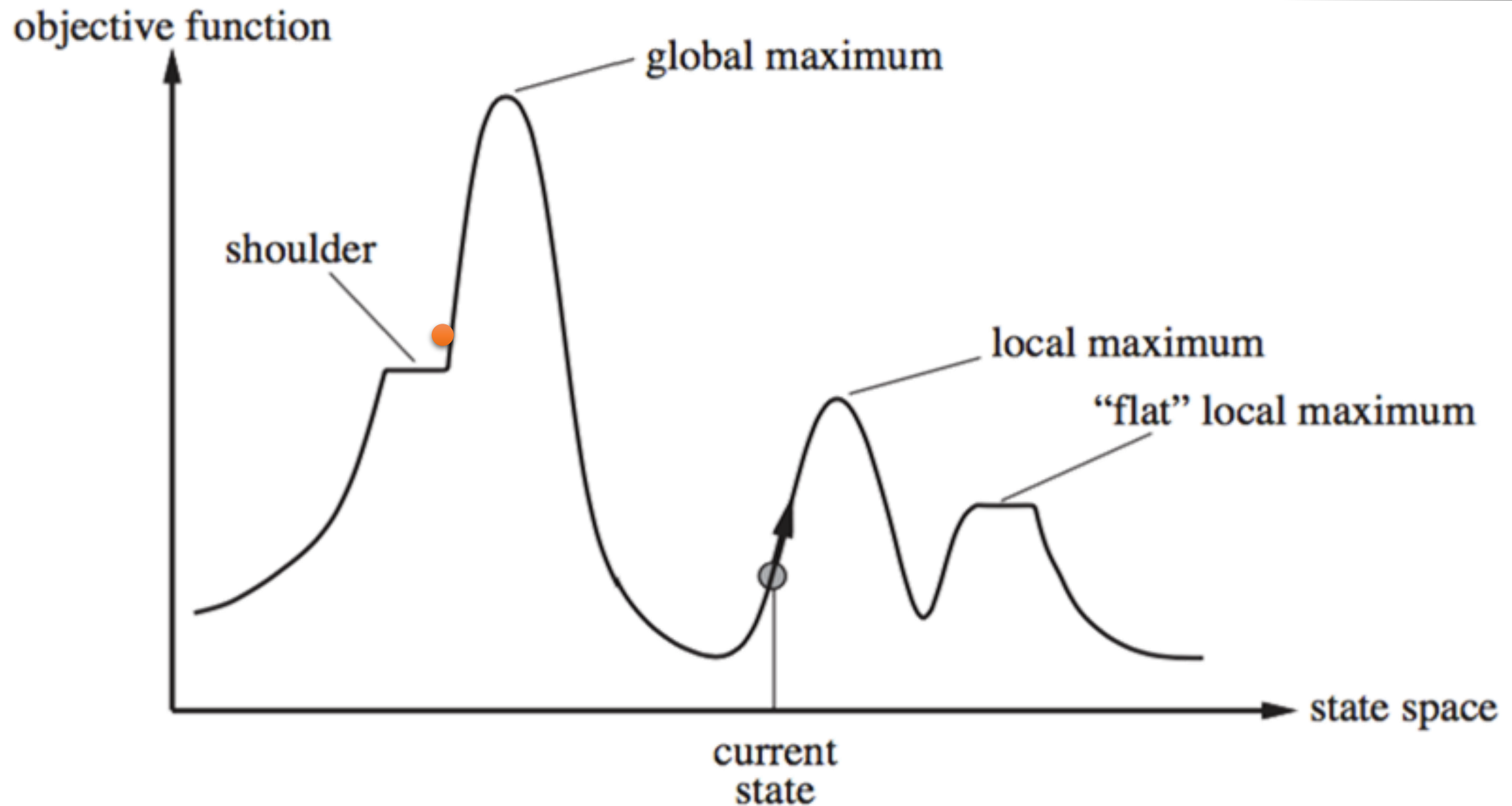
نسخه‌های مختلف جستجوی تپه‌نوردی



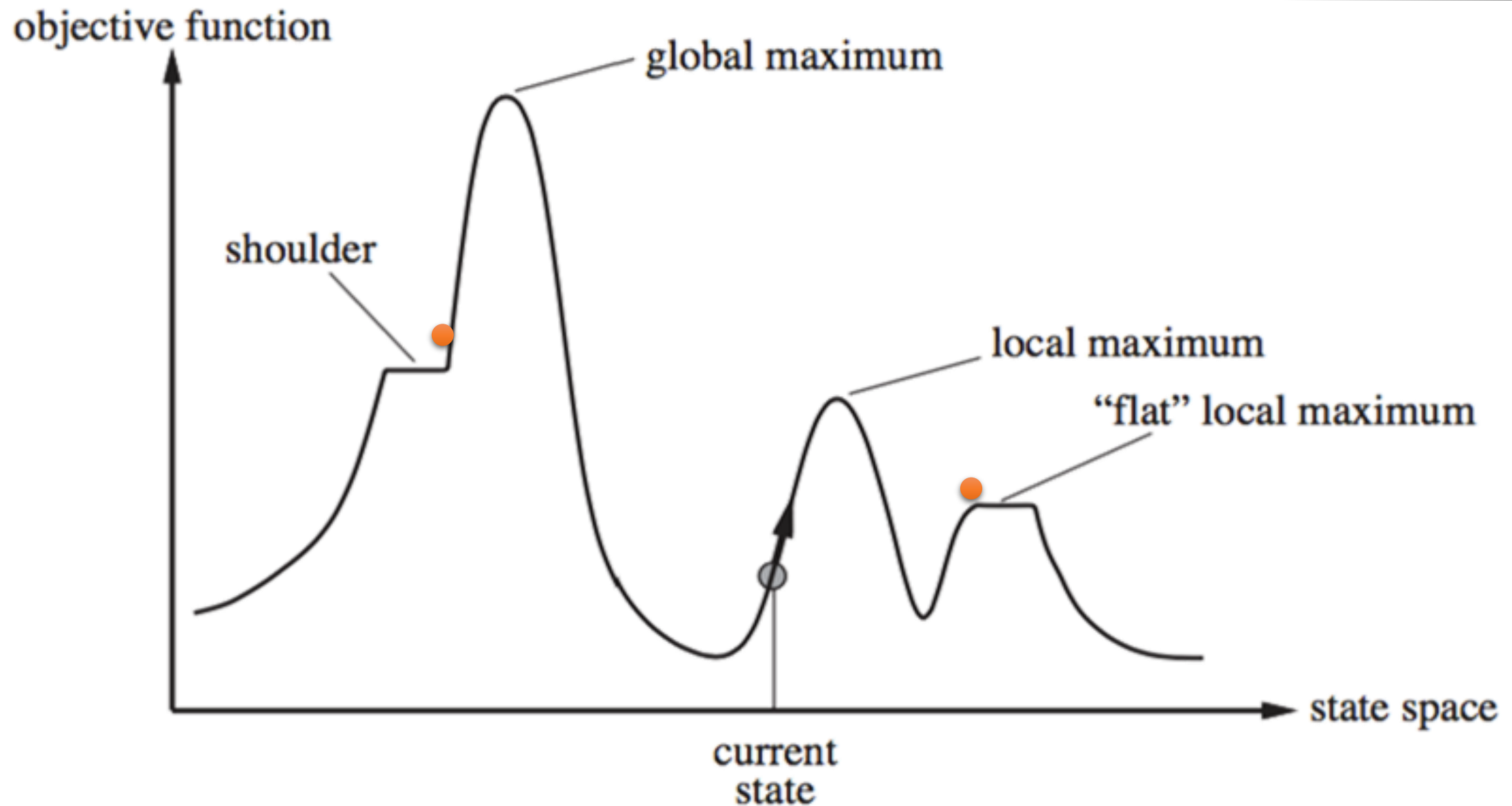
نسخه‌های مختلف جستجوی تپه‌نوردی



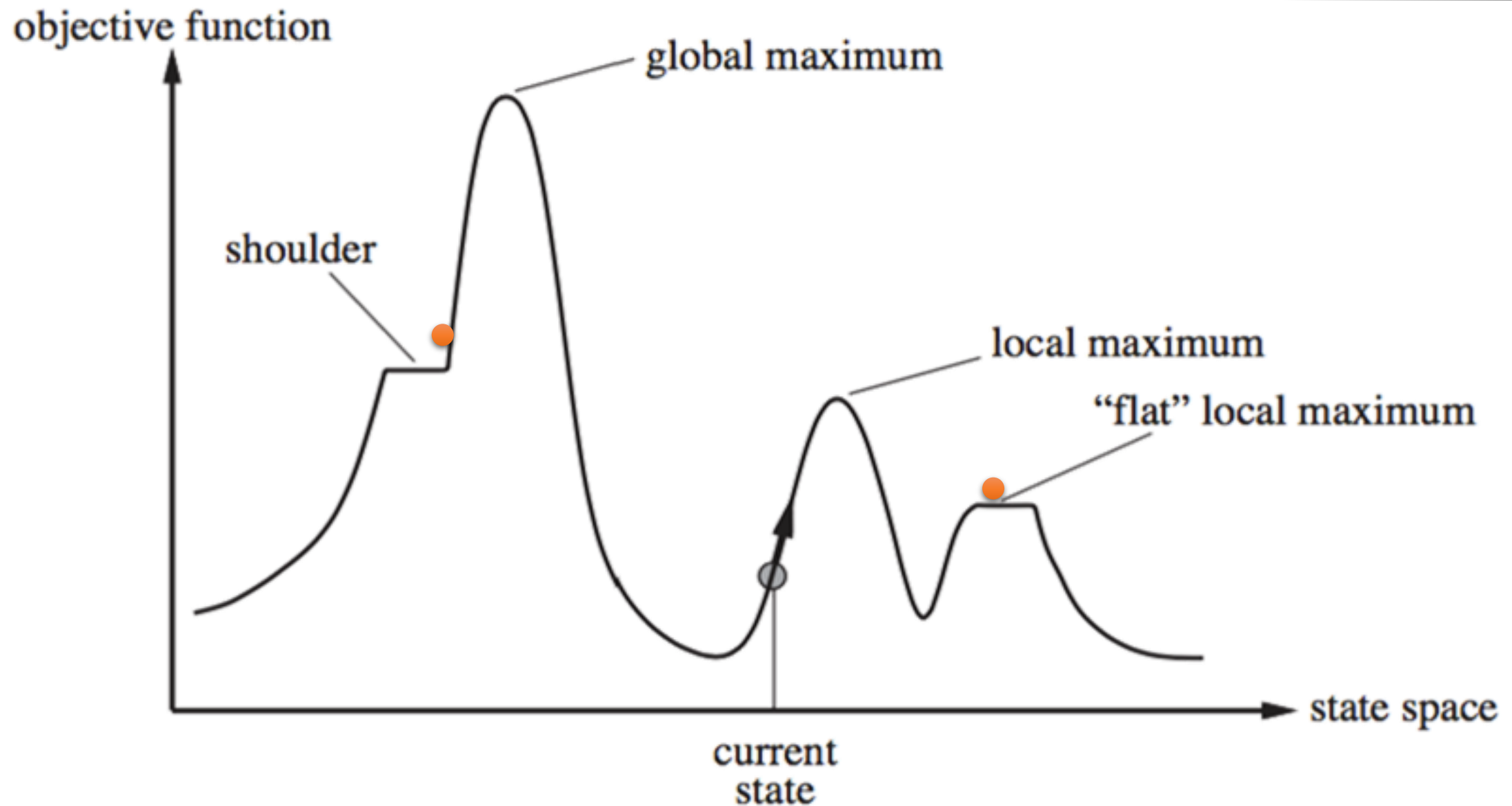
نسخه‌های مختلف جستجوی تپه‌نوردی



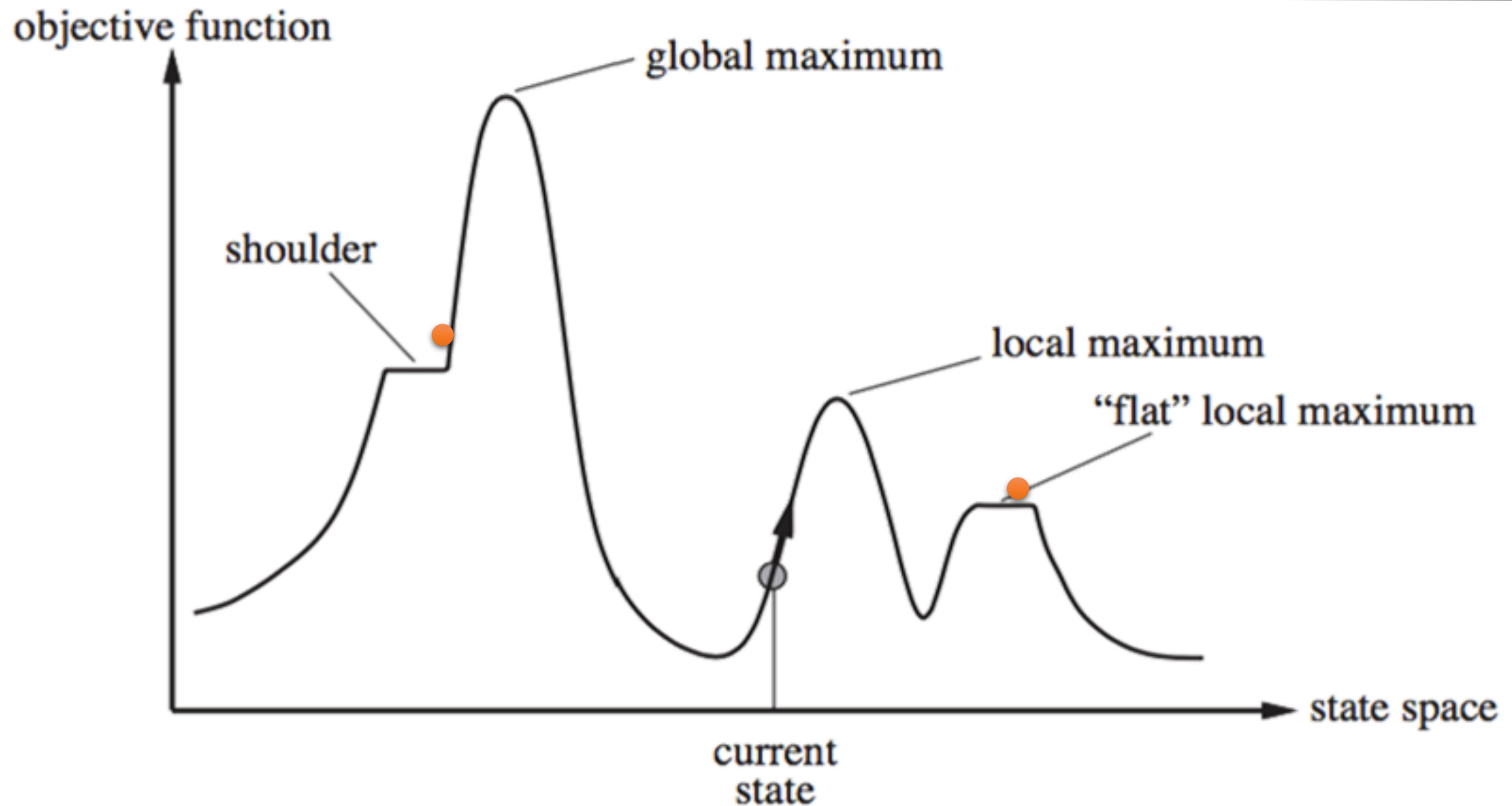
نسخه‌های مختلف جستجوی تپه‌نوردی



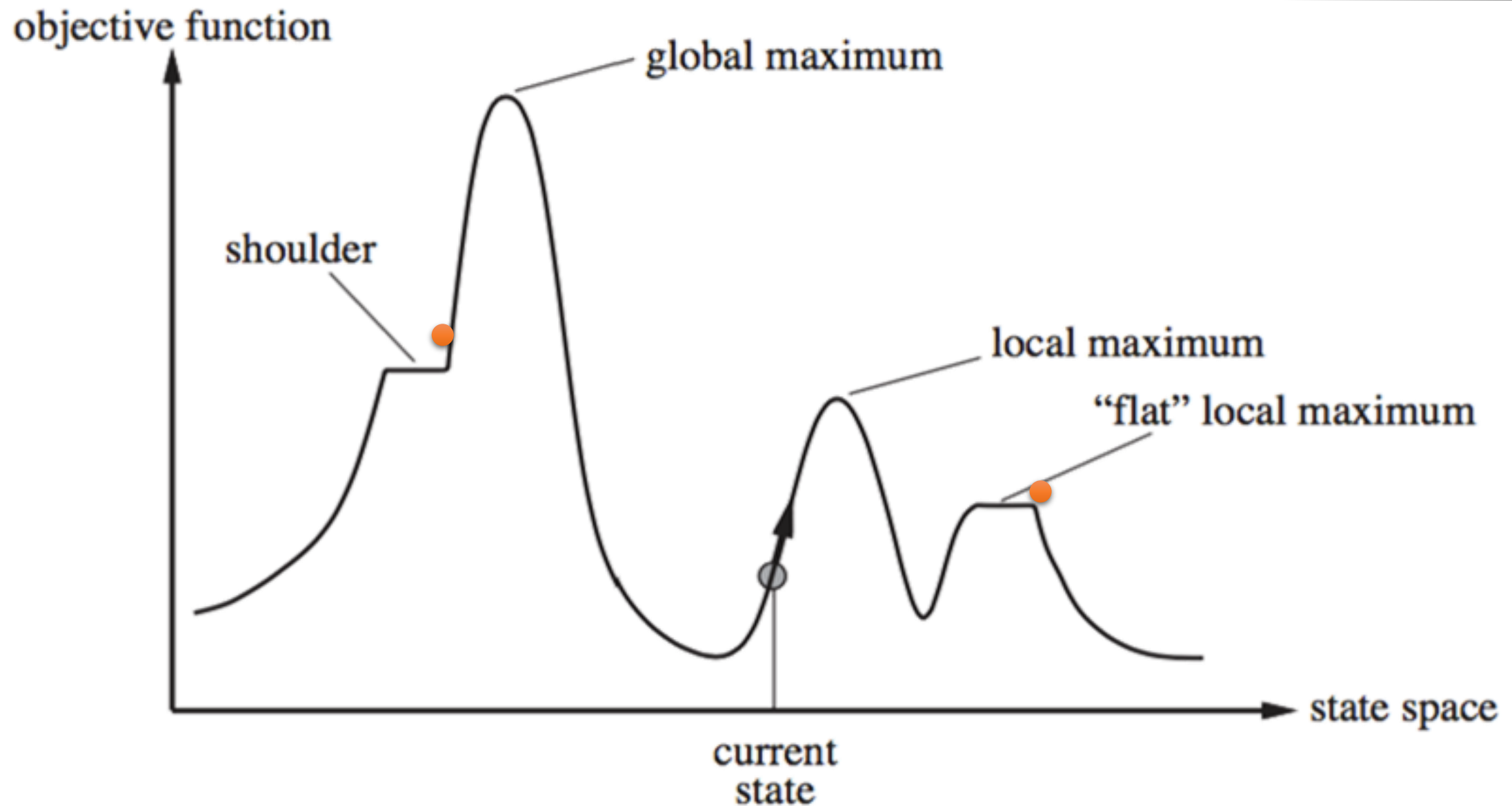
نسخه‌های مختلف جستجوی تپه‌نوردی



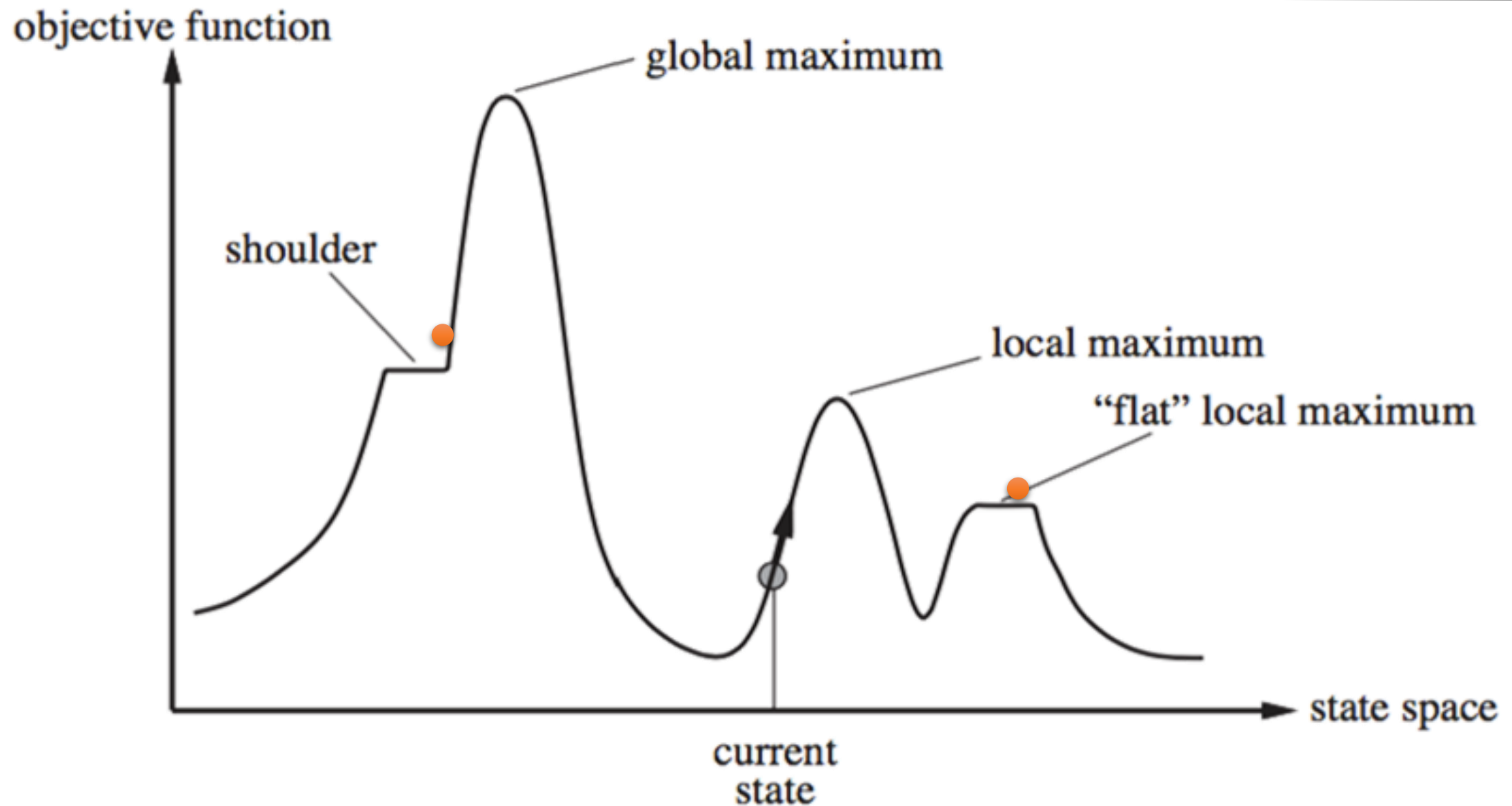
نسخه‌های مختلف جستجوی تپه‌نوردی



نسخه‌های مختلف جستجوی تپه‌نوردی



نسخه‌های مختلف جستجوی تپه‌نوردی



نسخه‌های مختلف جستجوی تپه‌نوردی ...

• حرکت به کناره (Sideways move)

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

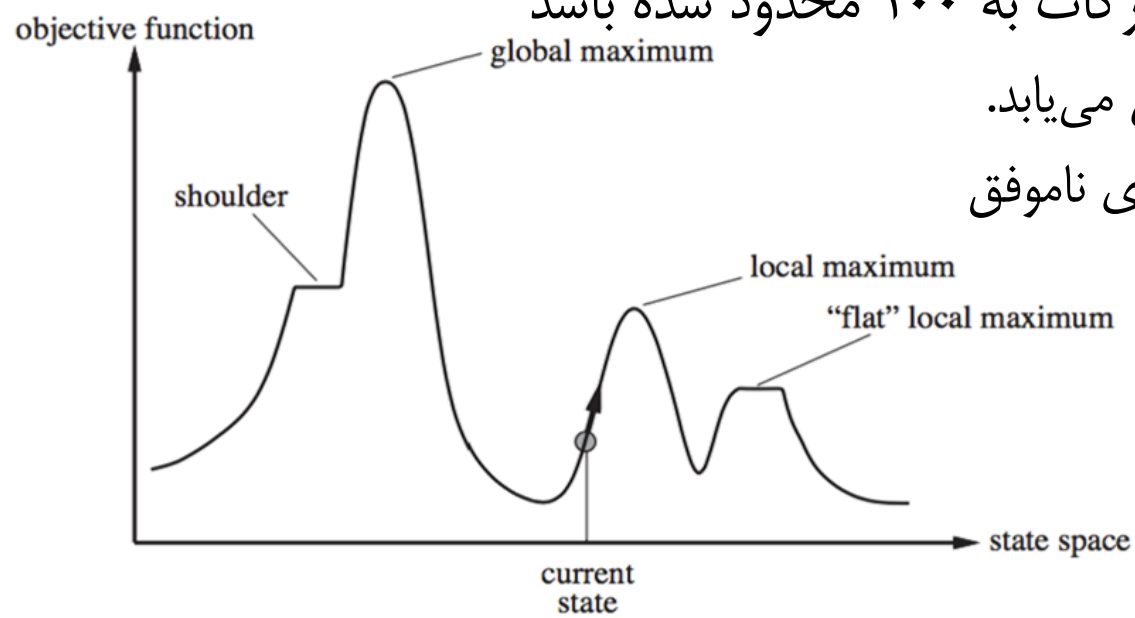
- مشکل: اگر الگوریتم به یک ماکزیمم محلی صاف برسد در یک حلقه بی‌نهایت گیر خواهد کرد.

- راه‌حل: 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 ۱۴ درصد به ۹۴ درصد افزایش می‌یابد.

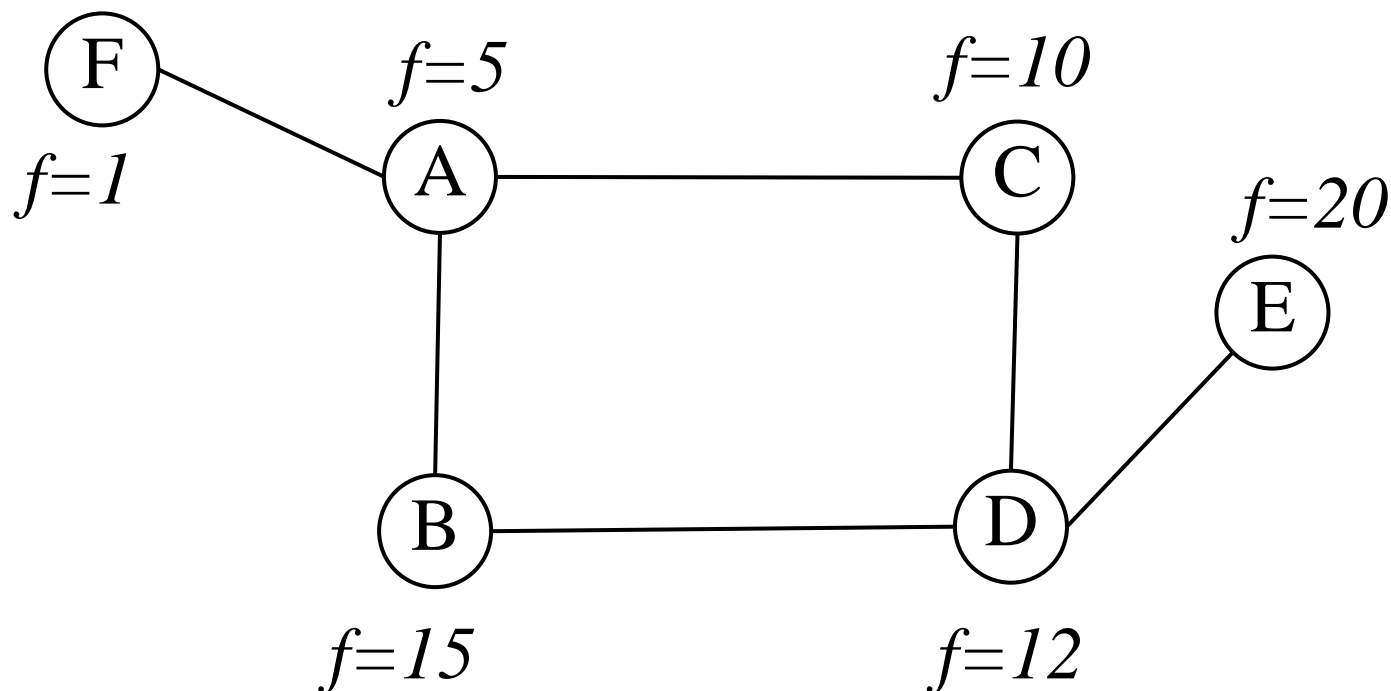
- اما برای مورد موفق به‌طور متوسط ۲۴ گام و برای ناموفق ۶۴ گام نیاز است.



نسخه‌های مختلف جستجوی تپه‌نوردی ...

• تپه‌نوردی اتفاقی (Stochastic hill climbing)

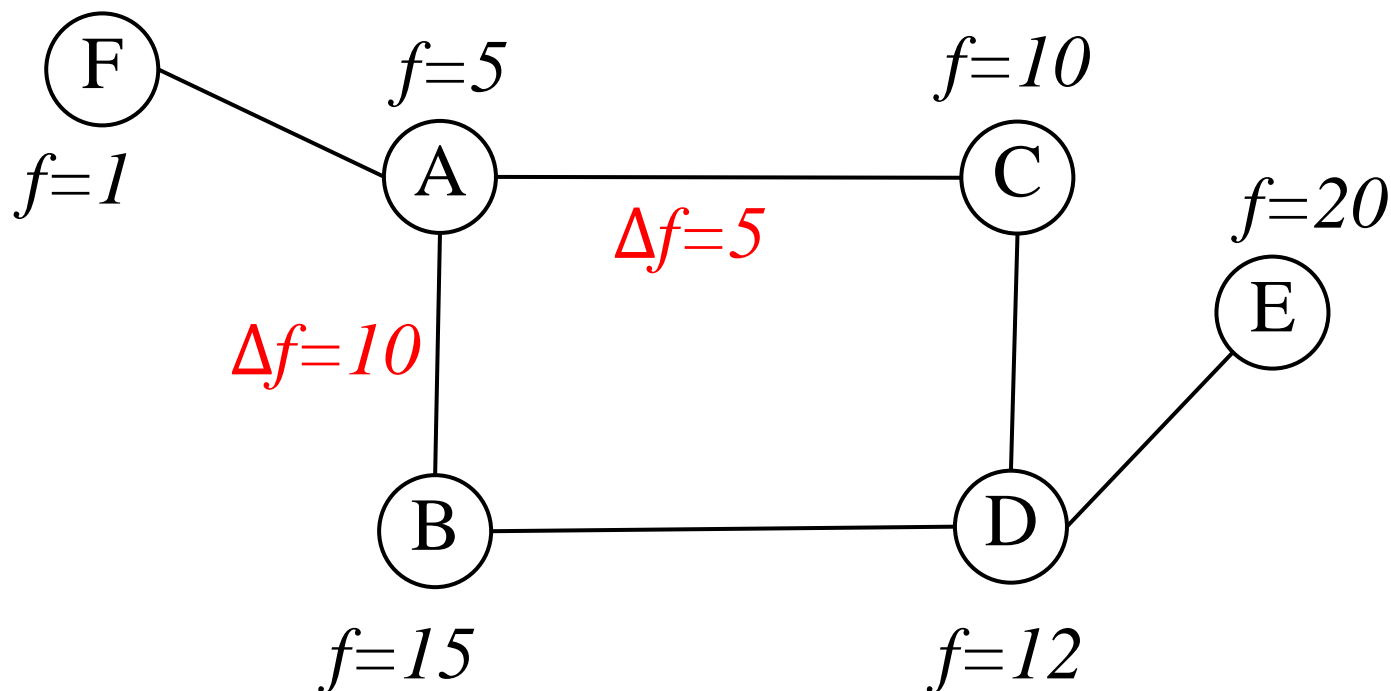
- از میان همسایه‌هایی که بهتر از نقطه‌ی فعلی هستند یکی را به تصادف انتخاب می‌کند.
- احتمال این انتخاب می‌تواند براساس شیب حرکتهای رو به بالا تغییر کند.
- این روش معمولاً کندتر از بیشترین شیب به نتیجه می‌رسد، اما در بعضی از دورنماهای حالت، بهترین راه‌حل را پیدا می‌کند.



نسخه‌های مختلف جستجوی تپه‌نوردی ...

• تپه‌نوردی اتفاقی (Stochastic hill climbing)

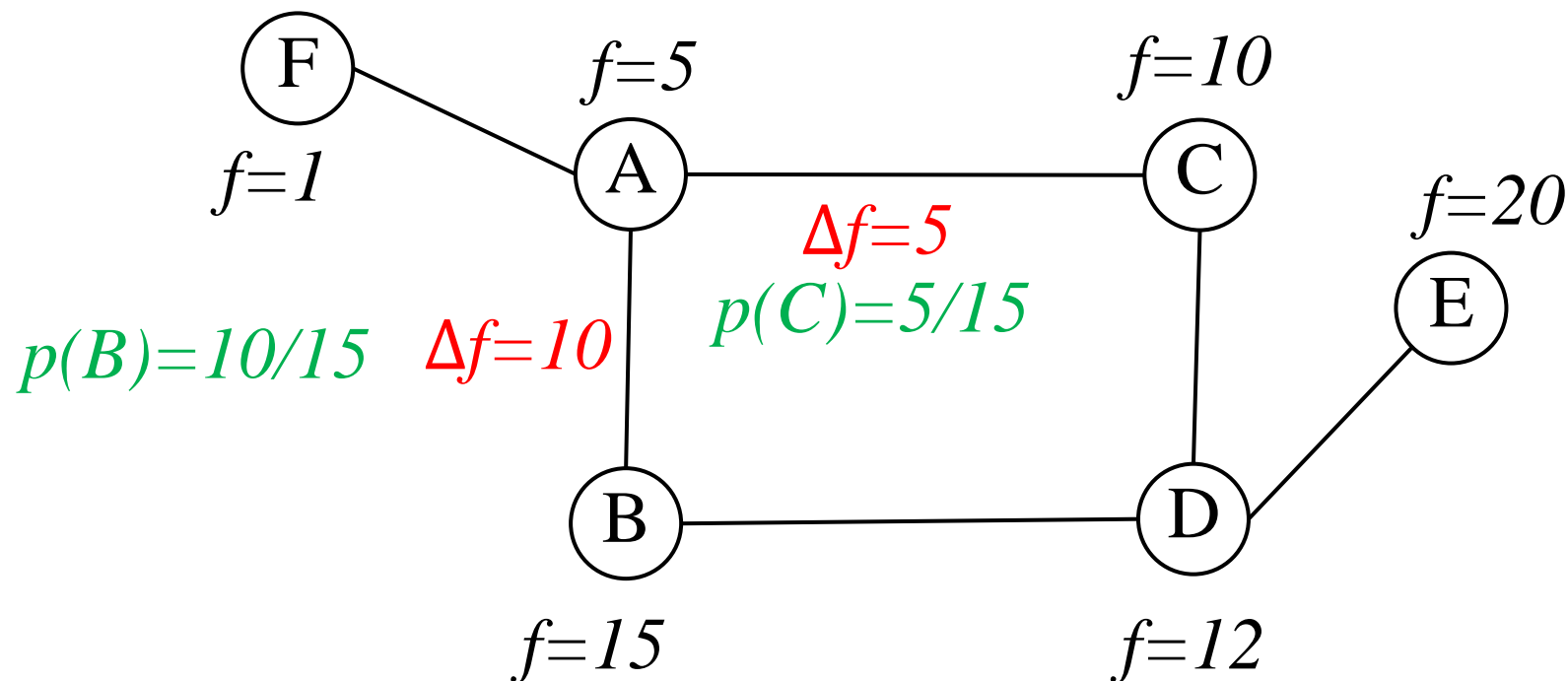
- از میان همسایه‌هایی که بهتر از نقطه‌ی فعلی هستند یکی را به تصادف انتخاب می‌کند.
- احتمال این انتخاب می‌تواند براساس شیب حرکت‌های رو به بالا تغییر کند.
- این روش معمولاً کندتر از بیشترین شیب به نتیجه می‌رسد، اما در بعضی از دورنماهای حالت، بهترین راه‌حل را پیدا می‌کند.



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.
- این روش معمولاً کندتر از بیشترین شیب به نتیجه می‌رسد، اما در بعضی از دورنماهای حالت، بهترین راه‌حل را پیدا می‌کند.



نسخه‌های مختلف جستجوی تپه‌نوردی ...

- تپه‌نوردی اولین گزینه (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.
- این راهبرد هنگامی مناسب است که یک حالت دارای تعداد زیادی (مثلاً هزار) پسین باشد.

نسخه‌های مختلف جستجوی تپه‌نوردی ...

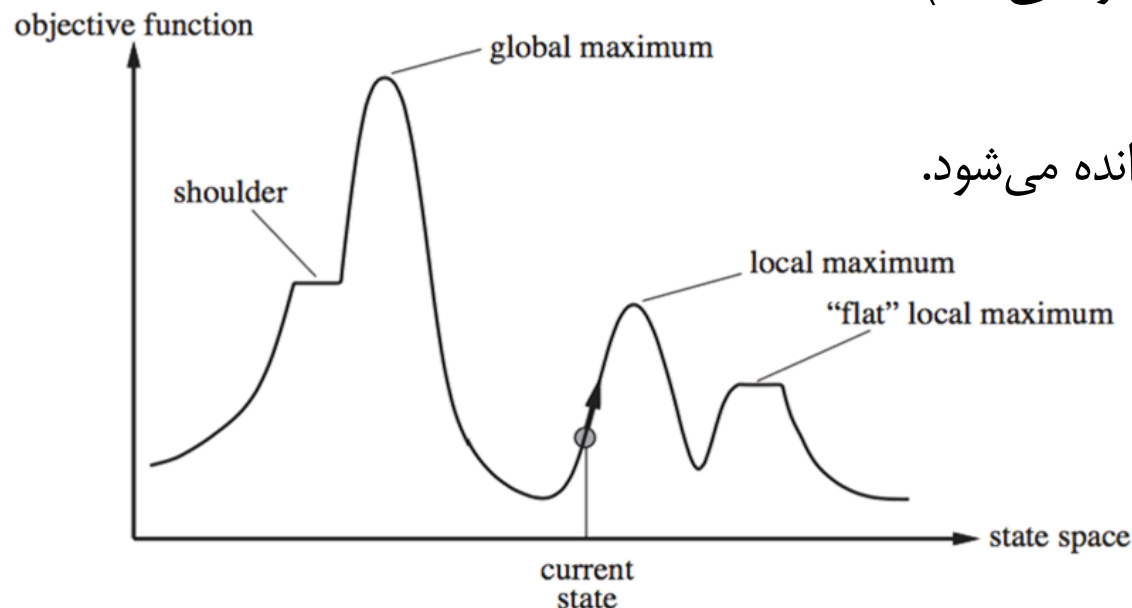
- همه‌ی نسخه‌های قبلی ناکامل هستند
- گیر کردن در بیشینه‌ی محلی

• تپه‌نوردی با شروع مجدد تصادفی (Random restart hill climbing)

- هر بار جستجوی تپه‌نوردی را از یک حالت شروع تصادفی اجرا 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 به‌عنوان یک حالت اولیه برگردانده می‌شود.



while state \neq goal **do**
 run hill-climbing search from a
 random initial state

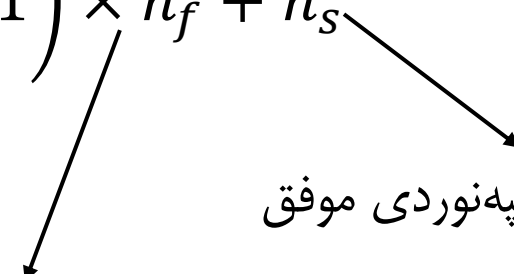
نسخه‌های مختلف جستجوی تپه‌نوردی ...

- p : احتمال موفقیت هر جستجوی تپه‌نوردی

- تعداد شروع مجدد مورد انتظار: $1/p$

- تعداد کل گام‌های مورد انتظار تا رسیدن به جواب بهینه:

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

$$\left(\frac{1}{p} - 1\right) \times n_f + n_s$$


میانگین تعداد گام‌ها در یک تپه‌نوردی موفق

میانگین تعداد گام‌ها در یک تپه‌نوردی ناموفق

- در مورد مسئله‌ی ۸ وزیر

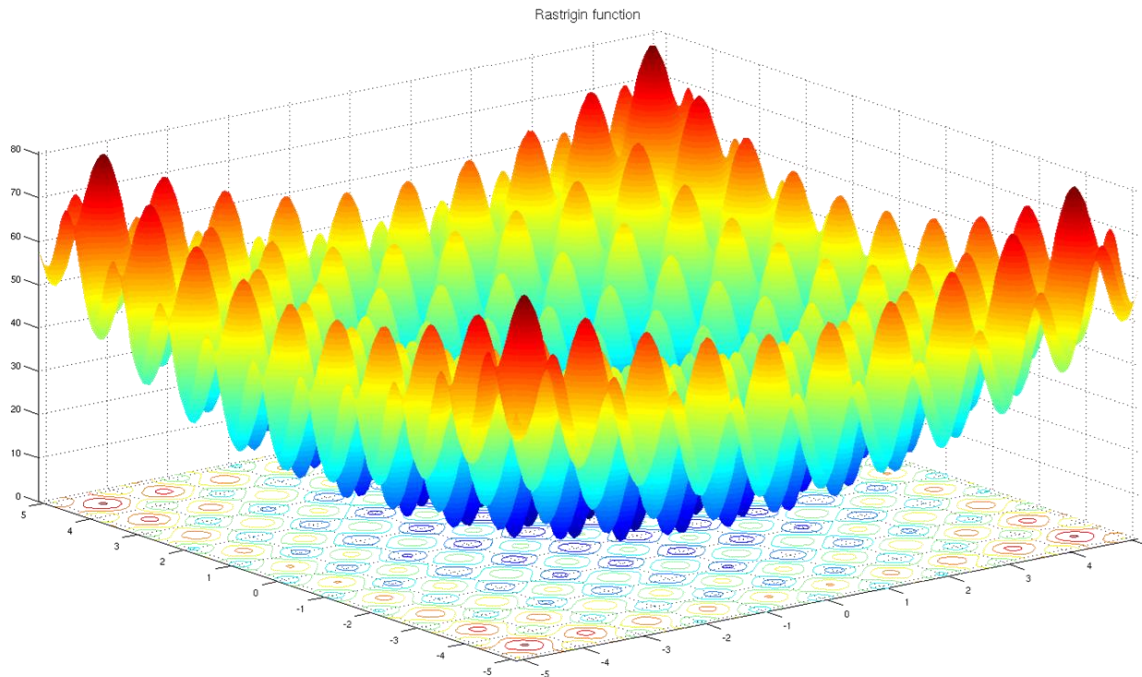
- در ۱۴ درصد از موارد وضعیت هدف پیدا می‌شود $\Leftrightarrow p=0.14$ و $1/p=7.14$

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

$$\Leftrightarrow (7-1) \times 3 + 4 = 22$$

تأثیر شکل دورنمای فضای حالت بر روی تپهنوردی

- شکل دورنمای فضای حالت اهمیت دارد
- اگر تعداد ماکزیمم‌های محلی و فلات‌ها کم باشد، تپهنوردی شروع مجدد تصادفی سریع‌ا راه‌حل خوبی را می‌یابد.
- دورنمای مسائل واقعی معمولاً از قبل شناخته‌شده نیست.



- مسائل NP-hard معمولاً دارای تعداد نمایی بیشینه‌ی محلی هستند.
- با این‌وجود 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 تا کنون دیده است را به خاطر سپرده و برمی‌گرداند.
- کامل ولی ناکارآمد (زمان زیاد برای یافتن جواب)

کدام یک از موارد زیر در مورد مقایسه دو روش جستجوی تپهنوردی ساده (تپهنوردی اولین گزینه) و تپهنوردی از تندترین شیب صحیح است؟
(فناوری اطلاعات ۸۳)

✓ ۱) تپهنوردی ساده کم‌تر در ماکزیمم محلی قرار می‌گیرد.

۲) تپهنوردی ساده با سرعت بیشتری حرکت می‌کند اما مسیر طولانی‌تری را می‌یابد.

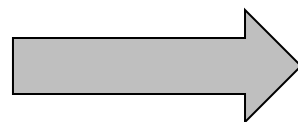
۳) تپهنوردی از تندترین شیب با سرعت بیشتری حرکت می‌کند اما حافظه بیشتری نیز مصرف می‌کند.

۴) تپهنوردی از تندترین شیب پاسخ بهینه را می‌یابد، درحالی‌که تپهنوردی ساده این‌طور نیست.

تمرین

n^2 - سودو کو حالت عمومی تر سودو کو است که در آن یک جدول $n^2 \times n^2$ با زیر جدول های $n \times n$ وجود دارد که با اعداد ۱ تا n^2 پر می شود. فرض کنید m خانه از جدول از قبل پر شده باشد، می خواهیم با استفاده از تپه نوردی جدول را حل کنیم.

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		



	1	2	3	4	5	6	7	8	9
A	4	8	3	9	2	1	6	5	7
B	9	6	7	3	4	5	8	2	1
C	2	5	1	8	7	6	4	9	3
D	5	4	8	1	3	2	9	7	6
E	7	2	9	5	6	4	1	3	8
F	1	3	6	7	9	8	2	4	5
G	3	7	2	6	8	9	5	1	4
H	8	1	4	2	5	3	7	6	9
I	6	9	5	4	1	7	3	8	2

تمرین...

الف) حالت‌ها، تابع هزینه و نحوه به‌دست آوردن همسایه‌ها را برای این مسئله توضیح دهید.

- حالت: برداری از سایز $n^4 - m$ که هر عنصر آن مقداری بین ۱ تا n^2 دارد.

- تابع هزینه: مجموع تعداد تناقض‌ها در سطرها، ستون‌ها و زیرجدول‌ها

- تولید همسایه: تغییر یکی از اعداد از مجموعه $n^4 - m$ عنصری

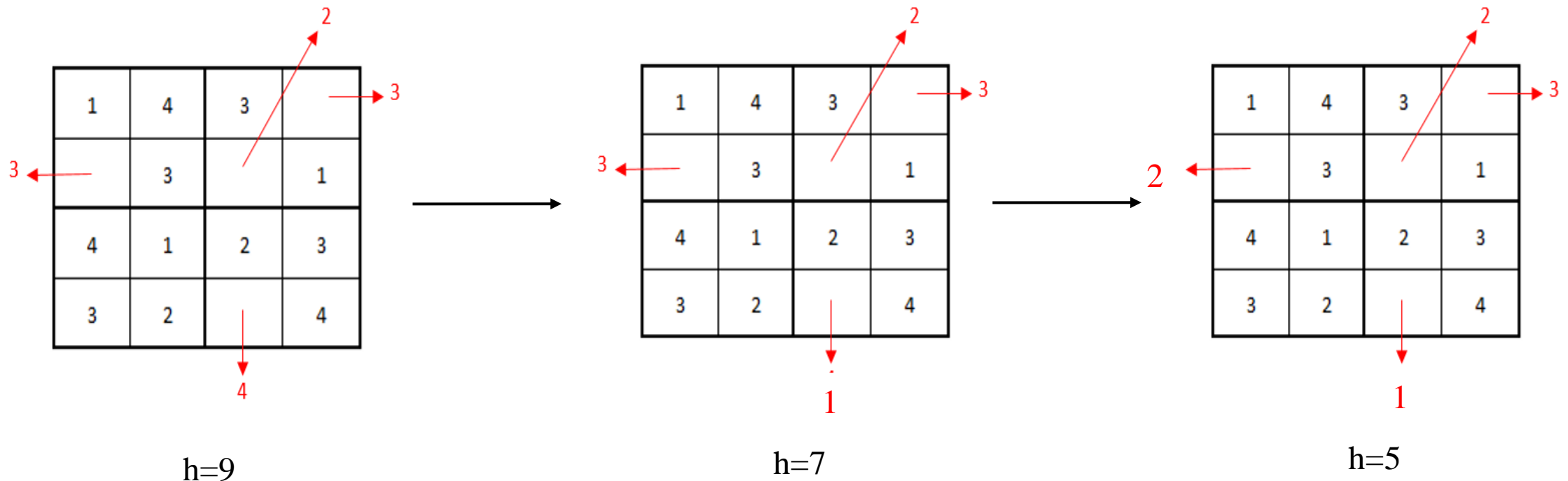
1	4	3	
	3		1
4	1	2	3
3	2		4

ب) تعداد همسایه‌های هر حالت را برحسب m و n حساب کنید.

$$(n^4 - m)(n^2 - 1) \cdot$$

تمرین...

ج) حالت اولیه نشان داده شده در شکل زیر را در نظر بگیرید. الگوریتم تپه‌نوردی را با در نظر گرفتن مقادیر اولیه مشخص شده برای خانه‌های خالی تا دو مرحله انجام دهید.



شبیه‌سازی ذوب فلزات (Simulated annealing)

تپه‌نوردی + قدم‌زن تصادفی

ناکامل و کارآمد \swarrow

کامل و ناکارآمد \swarrow

• ایده

- از ماکزیمم‌های 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 به‌عنوان 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.
- T : دما و ΔE : اختلاف سطح انرژی یا مقدار حالت

شبه‌سازی 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**

$T \leftarrow \text{schedule}(t)$

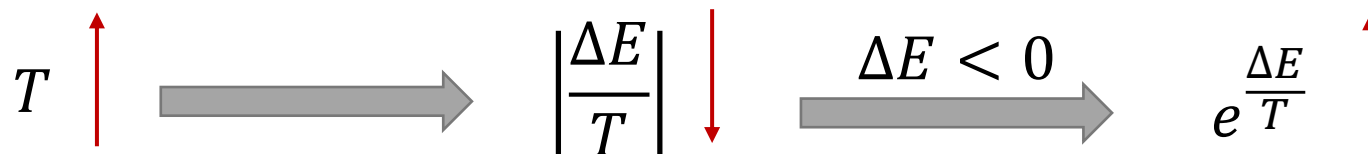
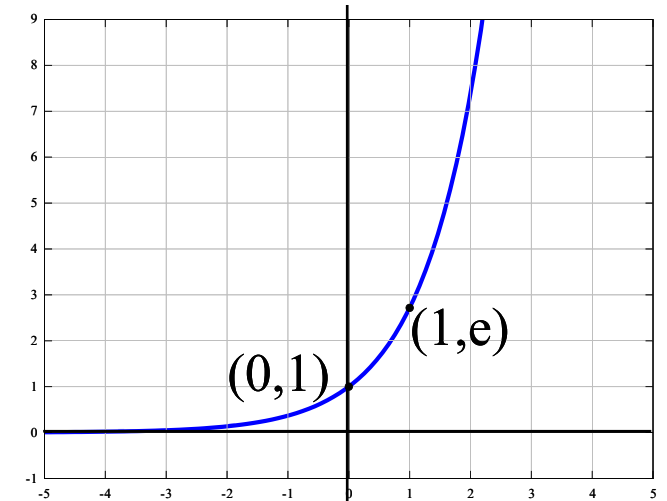
if $T = 0$ **then return** *current*

next \leftarrow a randomly selected successor of *current*

$\Delta E \leftarrow \text{next.VALUE} - \text{current.VALUE}$

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

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



شبیه‌سازی ذوب فلزات ...

- در مقادیر بالاتر T احتمال انجام حرکات بد (رفتن به سمت پایین) بیشتر است و با کاهش T این احتمال کاهش می‌یابد.
- اگر همواره T برابر با صفر باشد مانند تپه‌نوردی اولین انتخاب عمل می‌کند.
- اگر همواره T برابر با یک مقدار خیلی بزرگ باشد مانند قدم‌زن تصادفی 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.

کدام یک از گزینه‌های زیر در مورد الگوریتم تپهنوردی و simulated annealing غلط است؟
(فناوری اطلاعات دولتی ۹۳)

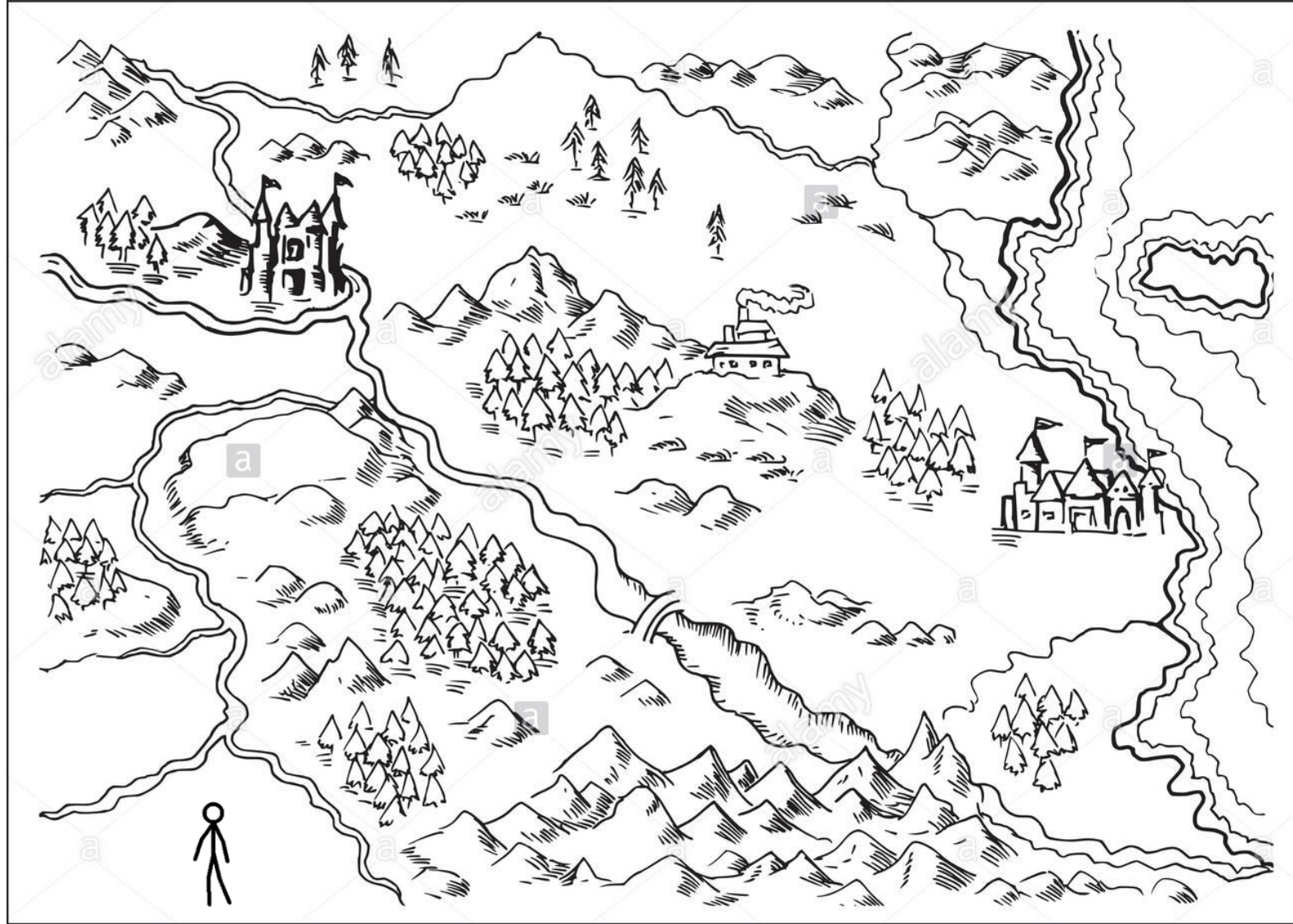
۱) الگوریتم تپهنوردی نزدیک‌ترین ماکزیمم را پیدا می‌کند.

۲) وقتی حرارت خیلی کم شود الگوریتم simulated annealing تبدیل به الگوریتم تپهنوردی می‌شود.

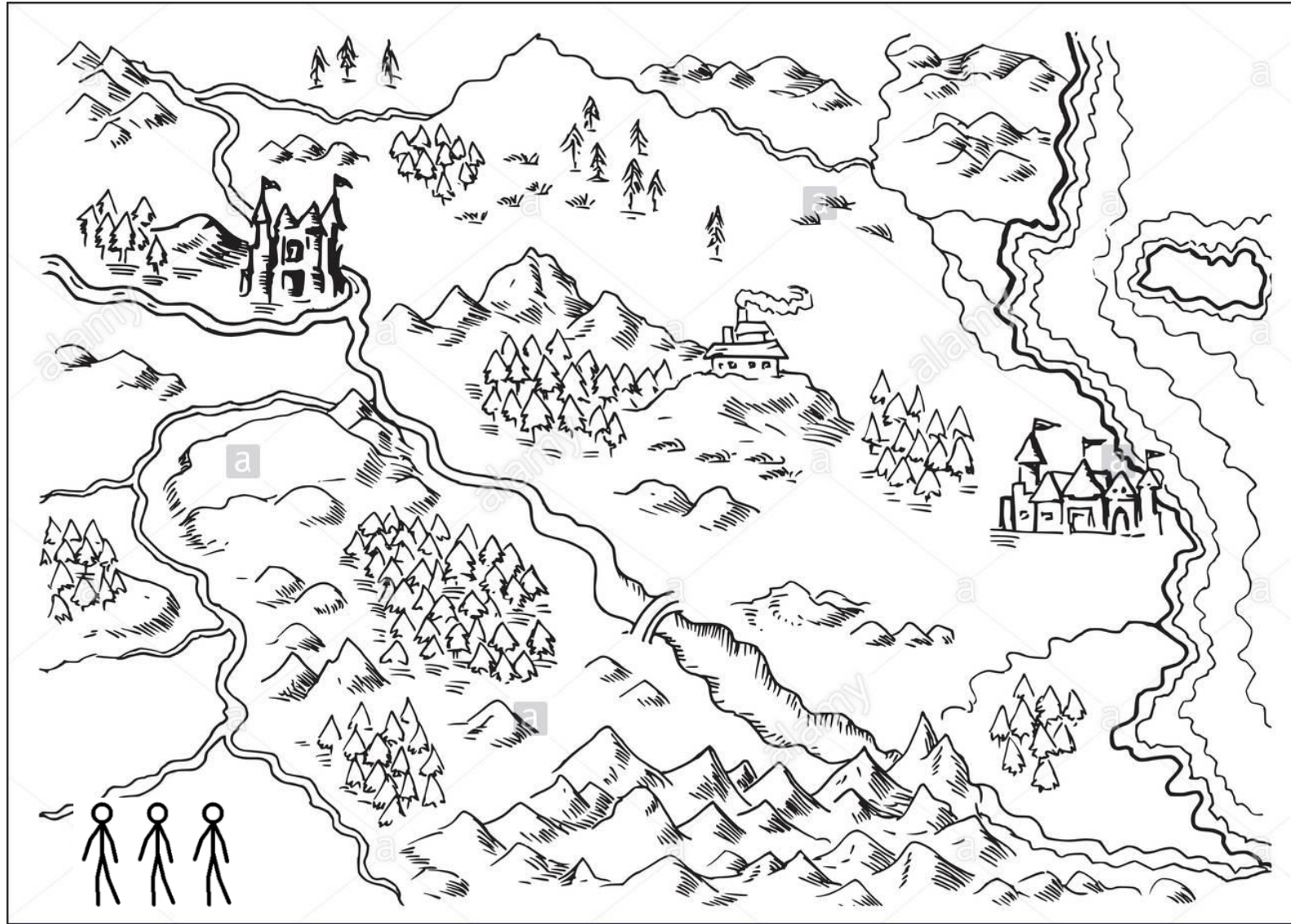
۳) اگر حرارت خیلی زیاد باشد و در طول الگوریتم کم نشود الگوریتم simulated annealing تبدیل به الگوریتم تصادفی می‌شود.

۴) ✓ الگوریتم simulated annealing مستقل از این که حرارت چه مقداری داشته باشد می‌تواند از مینیمم محلی فرار کند.

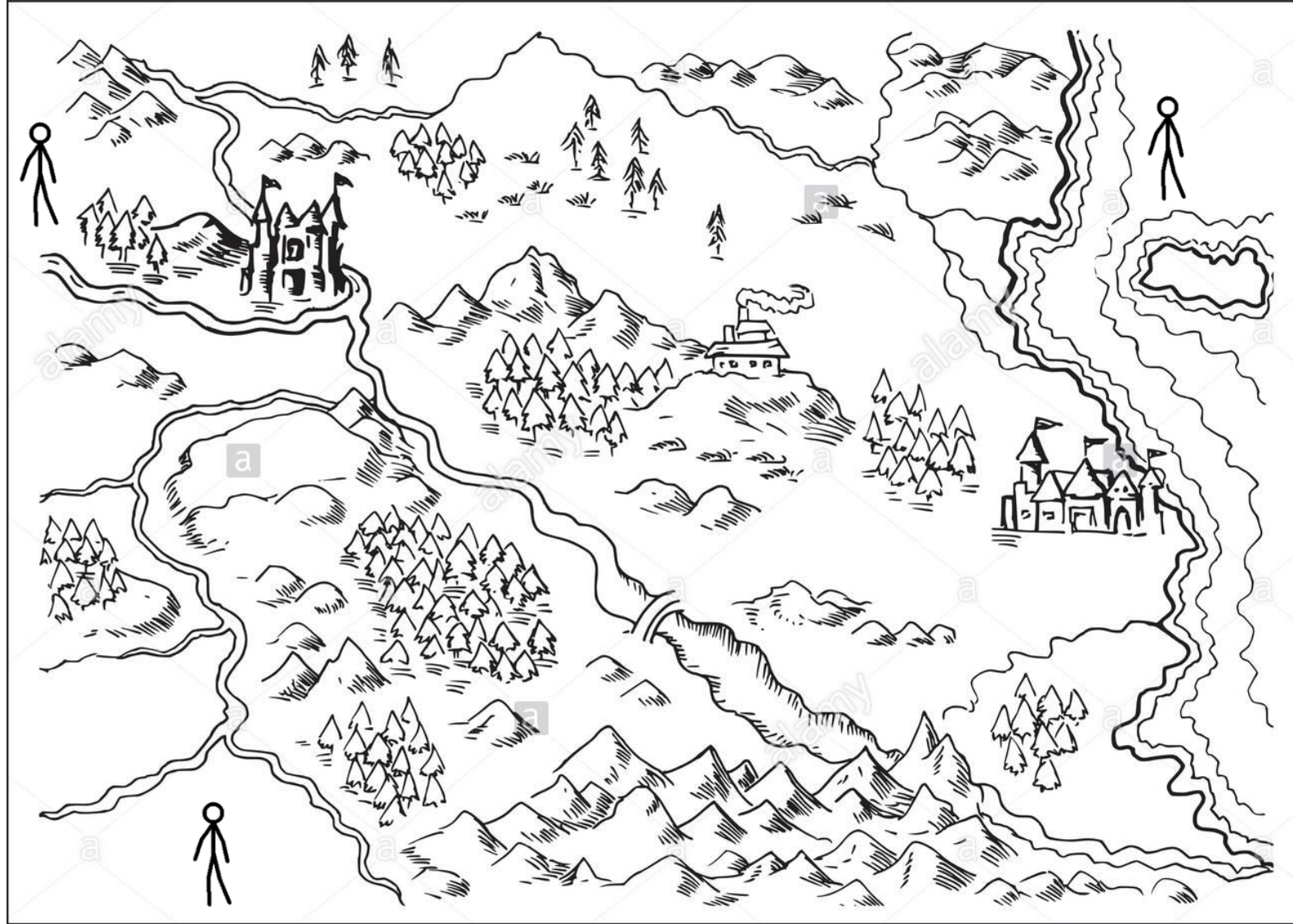
جستجوی پرتو محلی (Local Beam Search)



جستجوی پرتو محلی (Local Beam Search)



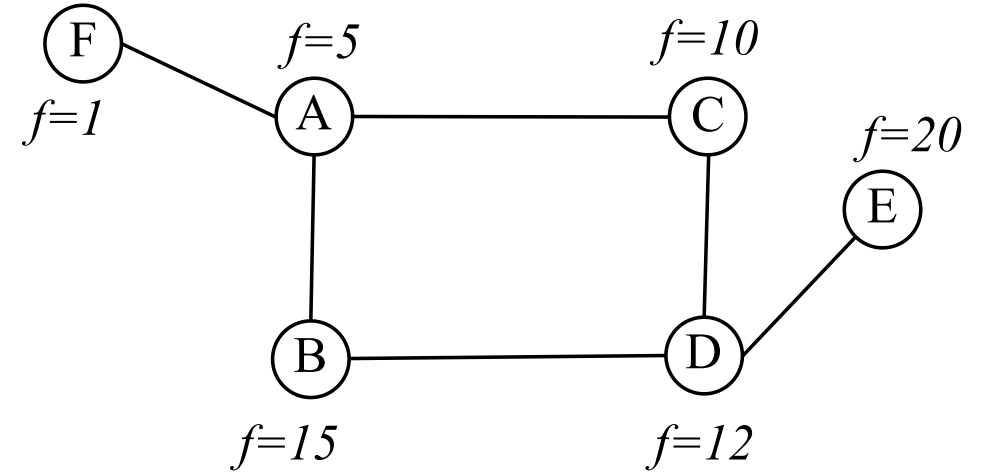
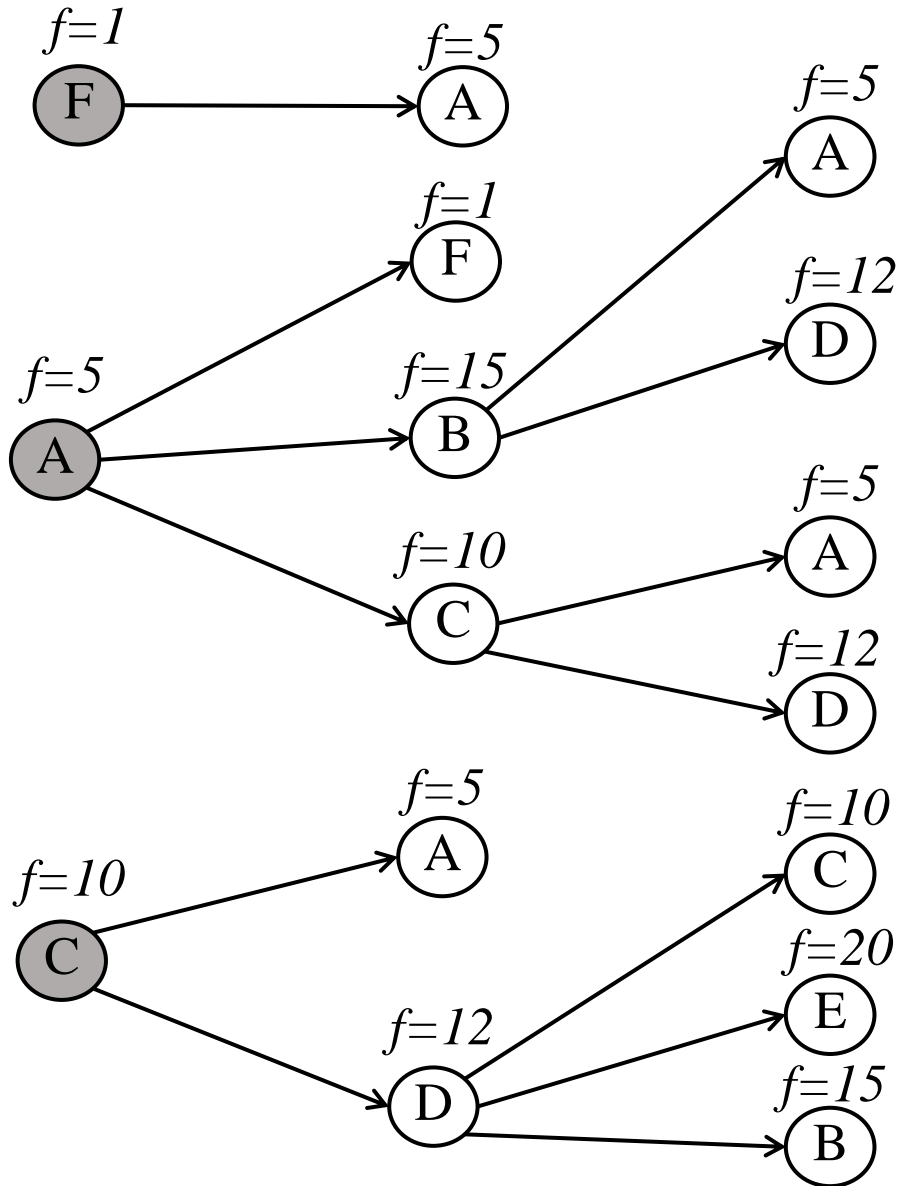
جستجوی پرتو محلی (Local Beam Search)



جستجوی پرتو محلی

- به جای نگه‌داری تنها یک گره در حافظه، چندین گره را در حافظه نگه‌داری می‌کند.
- شروع با k حالت که به طور تصادفی ایجاد شده‌اند.
- در هر تکرار، تمام فرزندان برای هر k حالت تولید می‌شوند.
- اگر یکی از آن‌ها حالت هدف بود جستجو متوقف می‌شود و در غیر این صورت از میان لیست کامل فرزندان، k تا از بهترین‌ها انتخاب می‌شوند و مرحله بالا تکرار می‌شود.
- تفاوت با جستجوی با شروع مجدد تصادفی
 - در جستجوی با شروع مجدد تصادفی، هر فرآیند جستجو به‌طور مستقل از بقیه اجرا می‌شود.
 - در جستجوی پرتو محلی، اطلاعات سودمند در بین k جستجوی موازی رد و بدل می‌گردد.
 - اگر همسایه‌های یک نقطه از همسایه‌های نقاط دیگر بهتر باشد الگوریتم بر روی همسایه‌های آن نقطه متمرکز می‌شود.

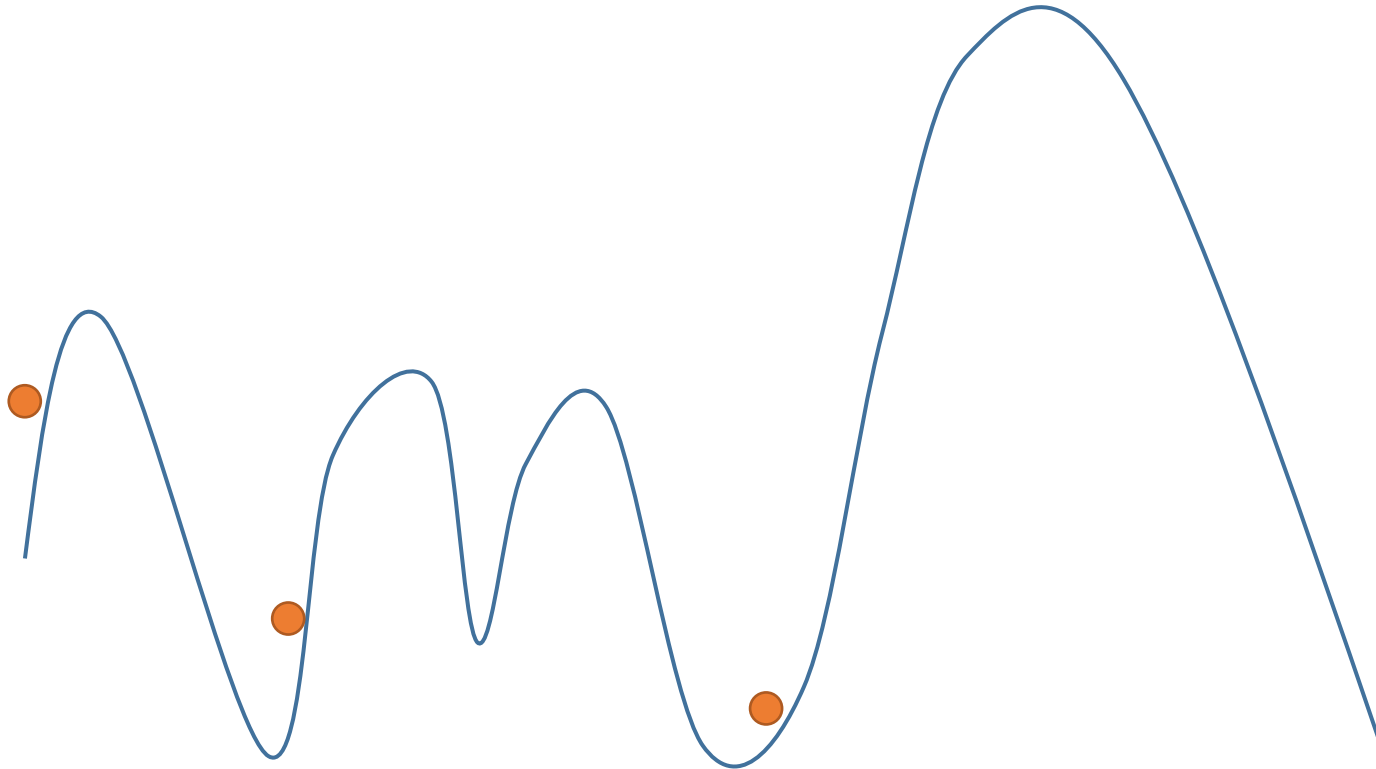
جستجوی پرتو محلی – مثال



جستجوی پرتو محلی ...

- مشکل جستجوی پرتو محلی

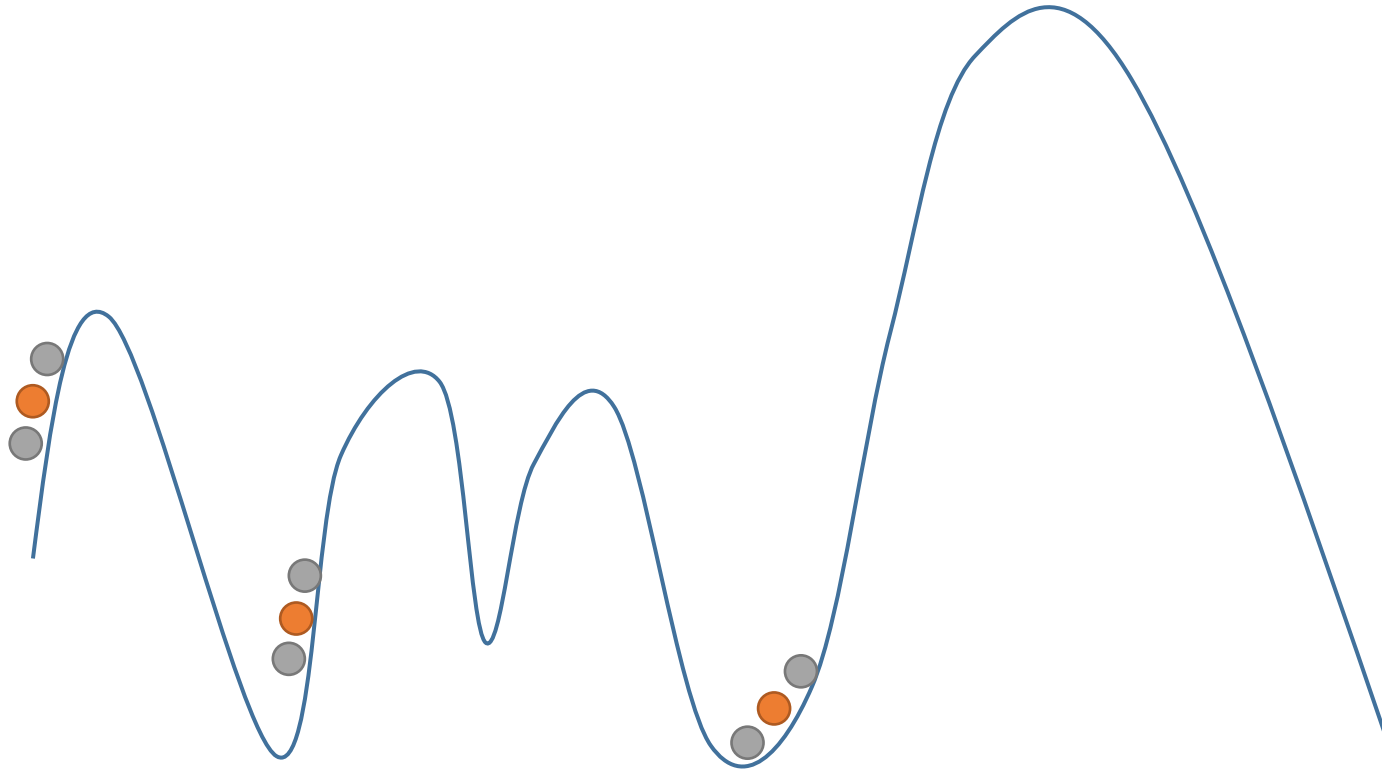
- این روش ممکن است از نبود تنوع بین k حالت رنج ببرد. به سرعت تمامی آن‌ها در محدوده کوچکی از فضای حالت جمع شوند.



جستجوی پرتو محلی ...

- مشکل جستجوی پرتو محلی

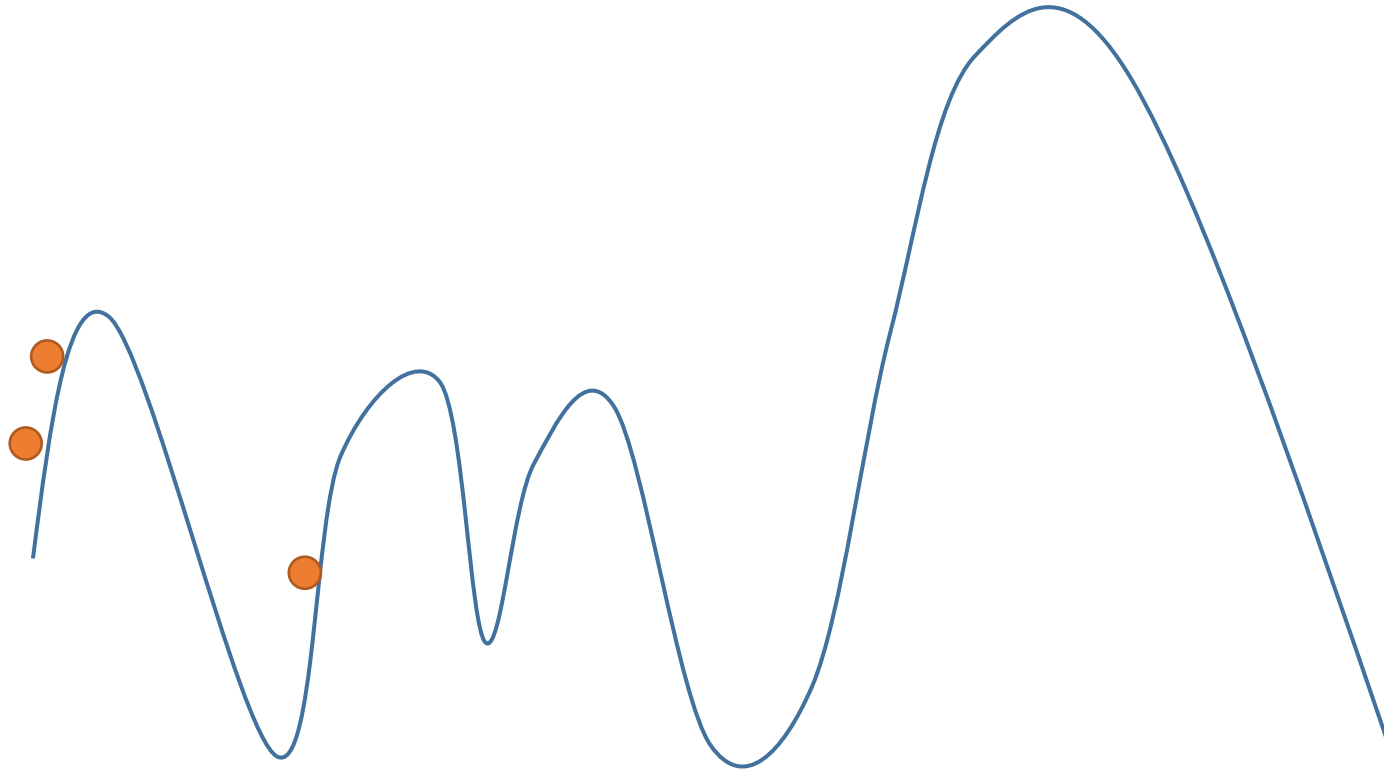
- این روش ممکن است از نبود تنوع بین k حالت رنج ببرد. به سرعت تمامی آن‌ها در محدوده کوچکی از فضای حالت جمع شوند.



جستجوی پرتو محلی ...

- مشکل جستجوی پرتو محلی

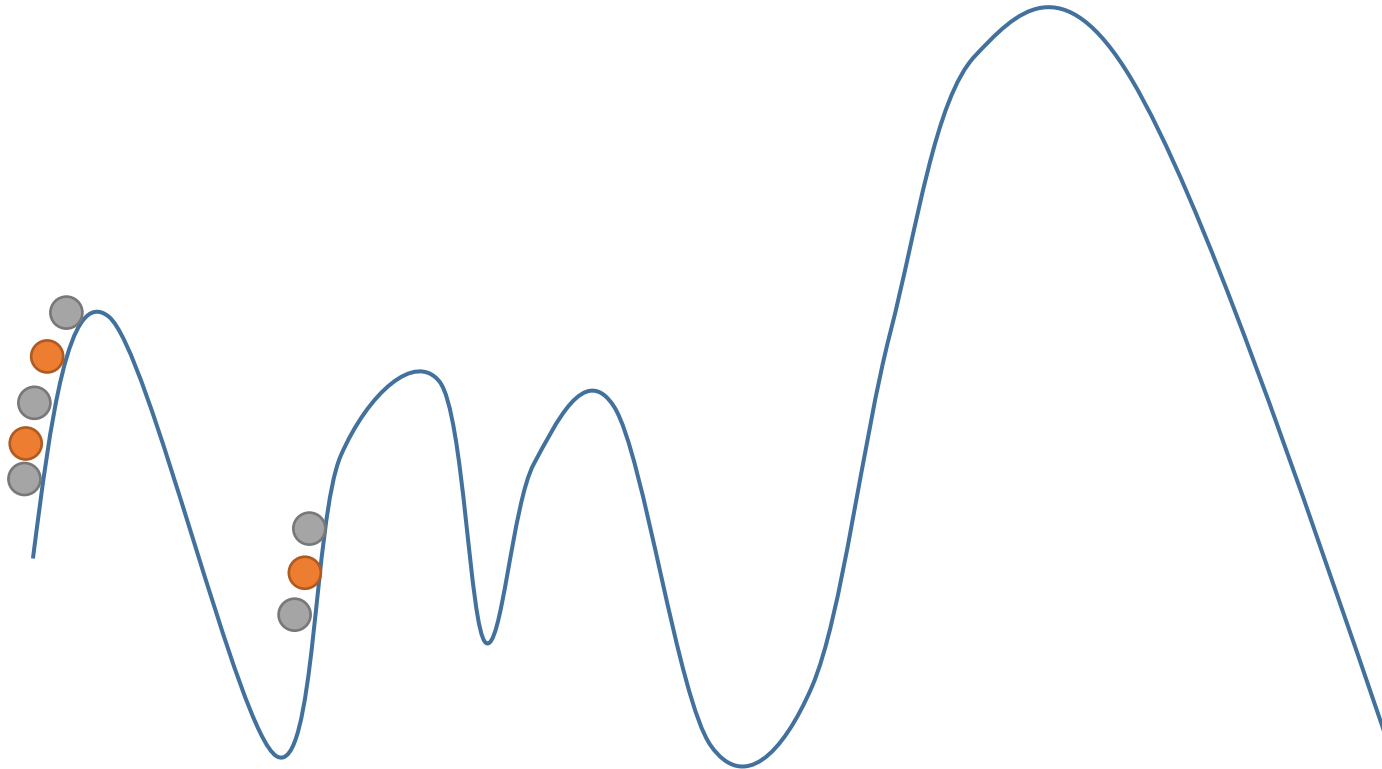
- این روش ممکن است از نبود تنوع بین k حالت رنج ببرد. به سرعت تمامی آن‌ها در محدوده کوچکی از فضای حالت جمع شوند.



جستجوی پرتو محلی ...

- مشکل جستجوی پرتو محلی

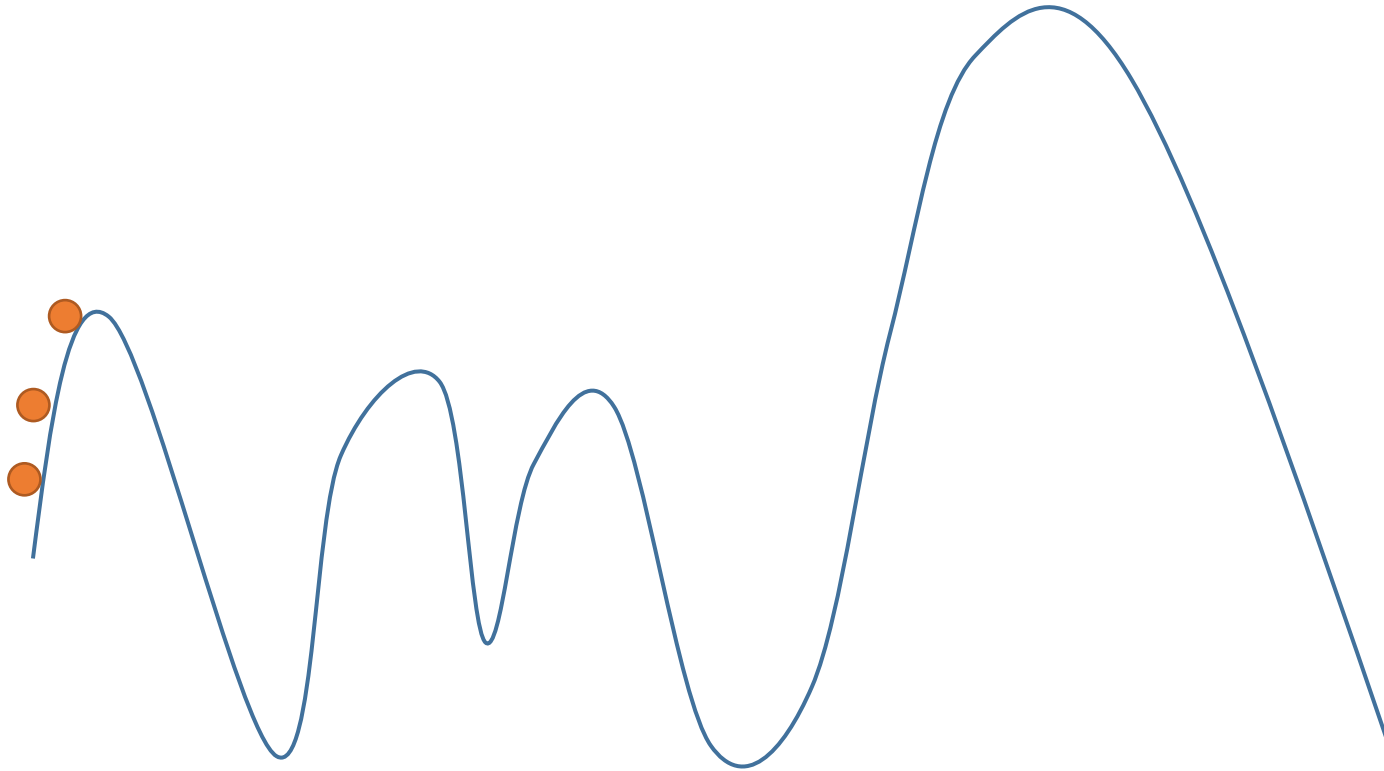
- این روش ممکن است از نبود تنوع بین k حالت رنج ببرد. به سرعت تمامی آن‌ها در محدوده کوچکی از فضای حالت جمع شوند.



جستجوی پرتو محلی ...

- مشکل جستجوی پرتو 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 پرتو محلی ...

- جستجوی پرتو 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.
- اگر $k=1$ و $k=\infty$ باشد، جستجوی پرتو محلی به ترتیب به کدام جستجوها تبدیل می‌شوند؟

الگوریتم ژنتیک (Genetic Algorithm)

- گونه‌ای از جستجوی پرتو اتفاقی
- به جای تولید حالت‌های پسین با تغییر یکی از حالات، پسین‌ها از ترکیب دو حالت والد به دست می‌آیند.
- براساس ایده‌ی تکامل طبیعی و نظریه‌ی داروین
- 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.
- افراد باقی‌مانده از هر نسل یک جامعه، تولید مثل کرده و فرزندی تولید می‌کنند که نسل بعد را تشکیل می‌دهند.
- هر فرزند هر یک از ژن‌ها یا خصوصیات خود را از یکی 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.

الگوریتم ژنتیک ...

• چارچوب کلی یک الگوریتم ژنتیک

(۱) تعیین روش بازنمایی و تابع برازش (*fitness*)

(۲) ایجاد جمعیت اولیه

(۳) تکرار تا برقراری شرط خاتمه

(۱-۳) ارزیابی جمعیت

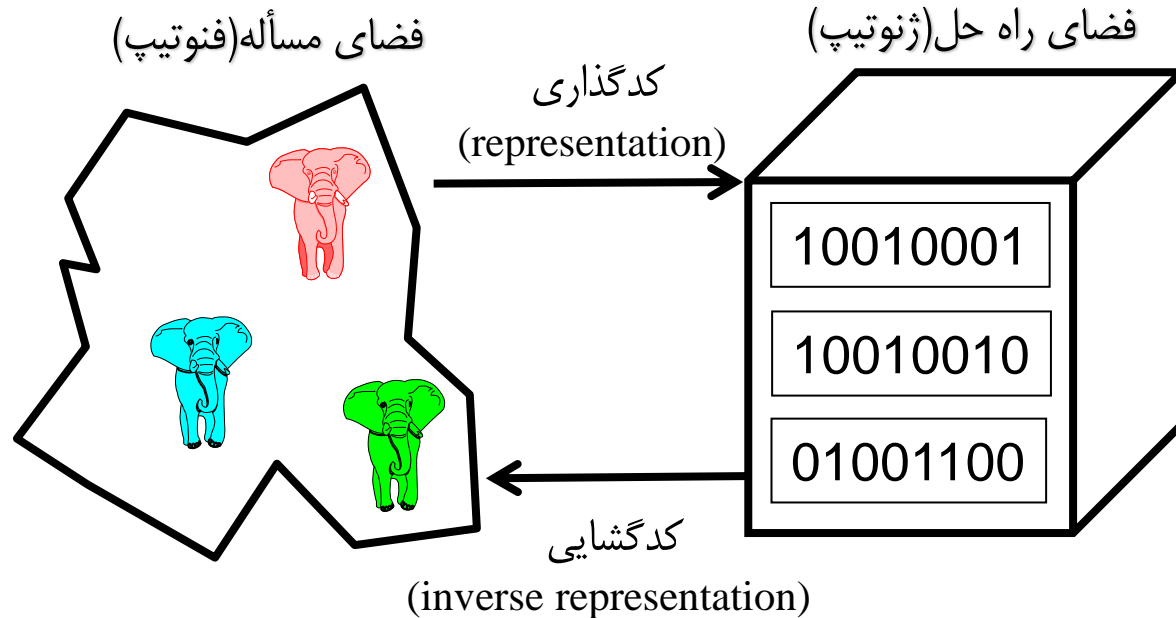
(۲-۳) انتخاب والدین

(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 گسسته: 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 مدل کننده هر وضعیت
- ژن: هر یک از فیلدهای یک کروموزوم



- برای آن که الگوریتم امکان یافتن جواب بهینه را داشته باشد روش بازنمایی باید تمام راه‌حل‌های 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، برازندگی آن بیشتر خواهد بود.

• مثال ۸- وزیر



2	4	7	4	8	5	5	2
---	---	---	---	---	---	---	---

- بازنمایی: آرایه‌ای یک بعدی از ۸ المان که المان i ام آن نشان‌دهنده‌ی سطر وزیر ستون i ام است.
- تابع برازش حالت n : تعداد زوج وزیرهایی که در وضعیت n همدیگر را گارد نمی‌دهند.
- در حالت نشان داده شده برابر با ۲۴ است.
- ماکزیمم برازندگی (برازندگی وضعیت هدف)؟

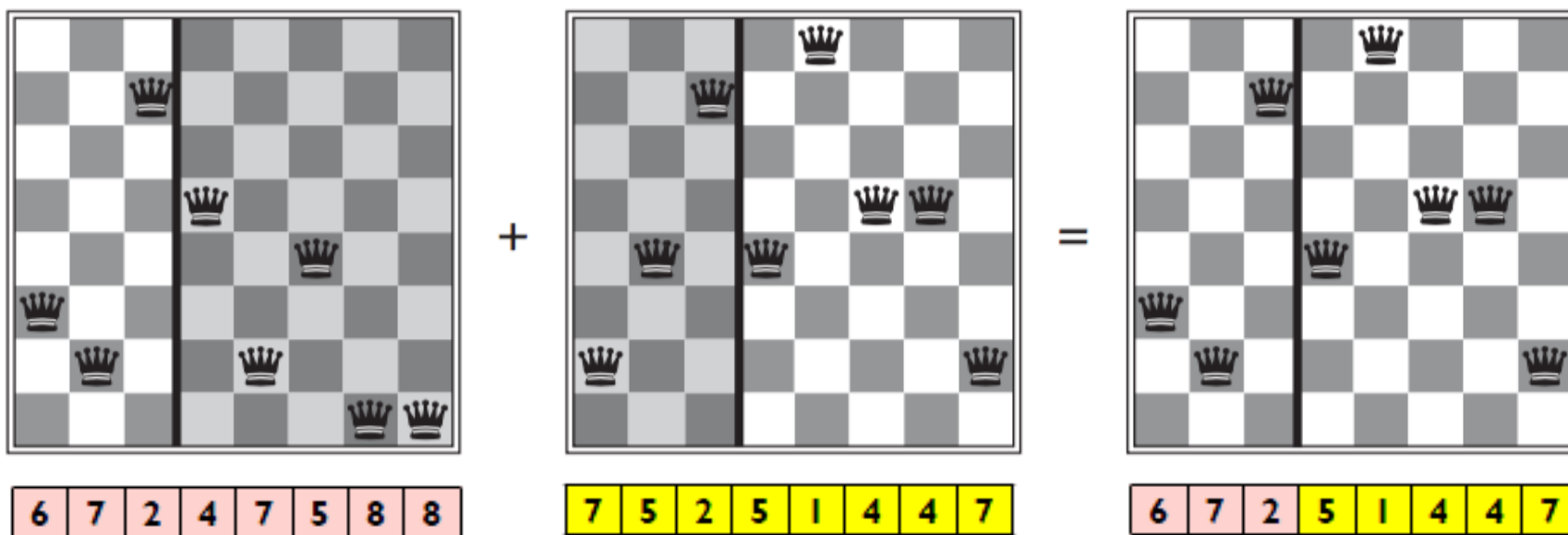
الگوریتم ژنتیک (از راهیان ارشد)

- 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)
 - ۱-۲ مقدار برازندگی هر یک از کروموزوم‌های نسل فعلی را محاسبه کن.
 - ۲-۲ به هریک از کروموزوم‌های نسل فعلی، با توجه به مقدار برازندگی‌اش، یک احتمال انتخاب نسبت بده، به طوری که کروموزوم با برازندگی بیشتر، احتمال انتخاب بیشتری داشته باشد.
 - ۳-۲ k 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 انتخاب‌شده را دوبه دو با هم ترکیب کن و از ترکیب هر دوتای آن‌ها، دو فرزند در نسل بعدی اضافه کن (عمل crossover یا ترکیب یا تقاطع).
 - ۵-۲ برخی ژن‌های برخی 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 متقاطع (Crossover)

- دو کروموزوم والد به روش‌های مختلفی می‌توانند با هم ترکیب شوند و دو فرزند تولید کنند.
- ترکیب یک نقطه‌ای

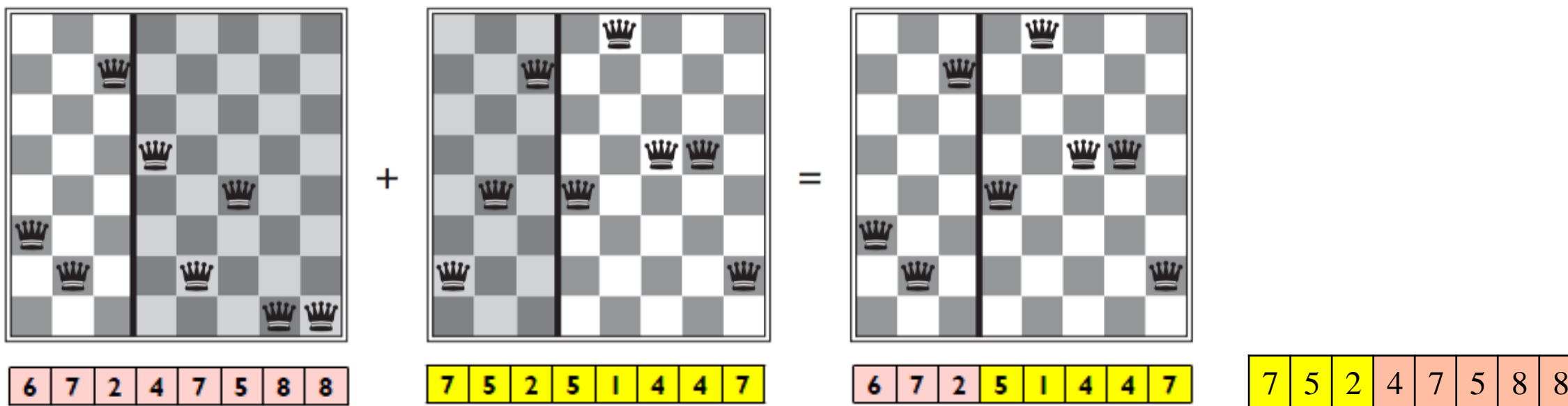
- انتخاب یک نقطه از موقعیت‌های داخل کروموزوم‌ها به‌طور تصادفی
- تولید فرزندان از برخورد کروموزوم‌های والد در نقطه‌ی پیوند



عمل ترکیب متقاطع (Crossover)

- دو کروموزوم والد به روش‌های مختلفی می‌توانند با هم ترکیب شوند و دو فرزند تولید کنند.
- ترکیب یک نقطه‌ای

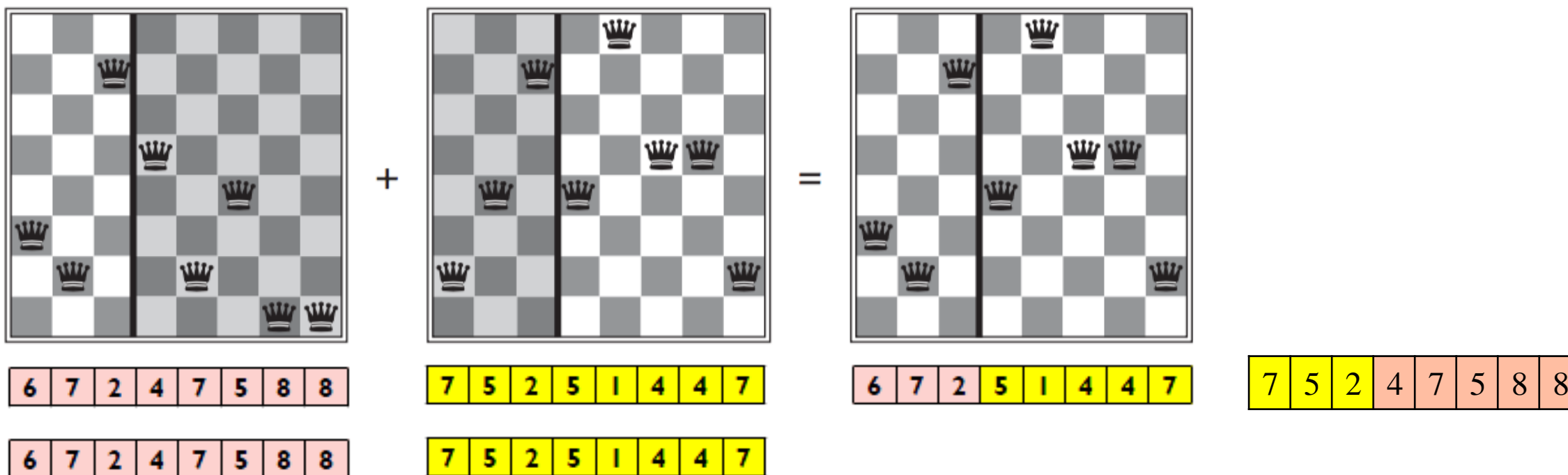
- انتخاب یک نقطه از موقعیت‌های داخل کروموزوم‌ها به‌طور تصادفی
- تولید فرزندان از برخورد کروموزوم‌های والد در نقطه‌ی پیوند



عمل ترکیب متقاطع (Crossover)

- دو کروموزوم والد به روش‌های مختلفی می‌توانند با هم ترکیب شوند و دو فرزند تولید کنند.
- ترکیب یک نقطه‌ای

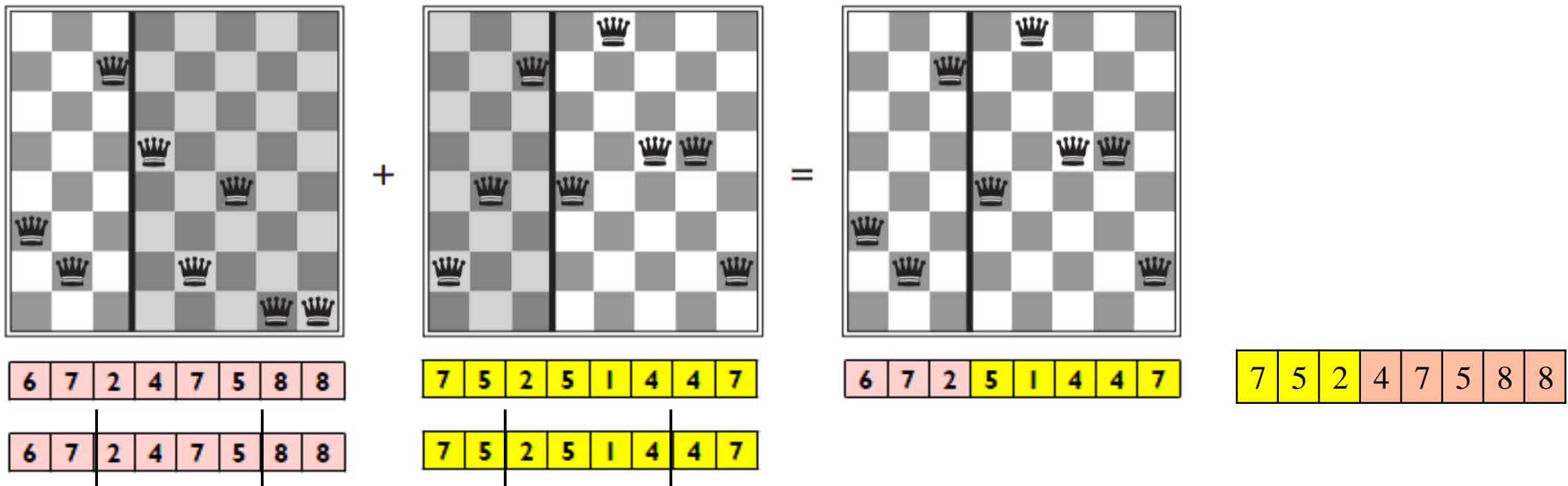
- انتخاب یک نقطه از موقعیت‌های داخل کروموزوم‌ها به‌طور تصادفی
- تولید فرزندان از برخورد کروموزوم‌های والد در نقطه‌ی پیوند



عمل ترکیب متقاطع (Crossover)

- دو کروموزوم والد به روش‌های مختلفی می‌توانند با هم ترکیب شوند و دو فرزند تولید کنند.
- ترکیب یک نقطه‌ای

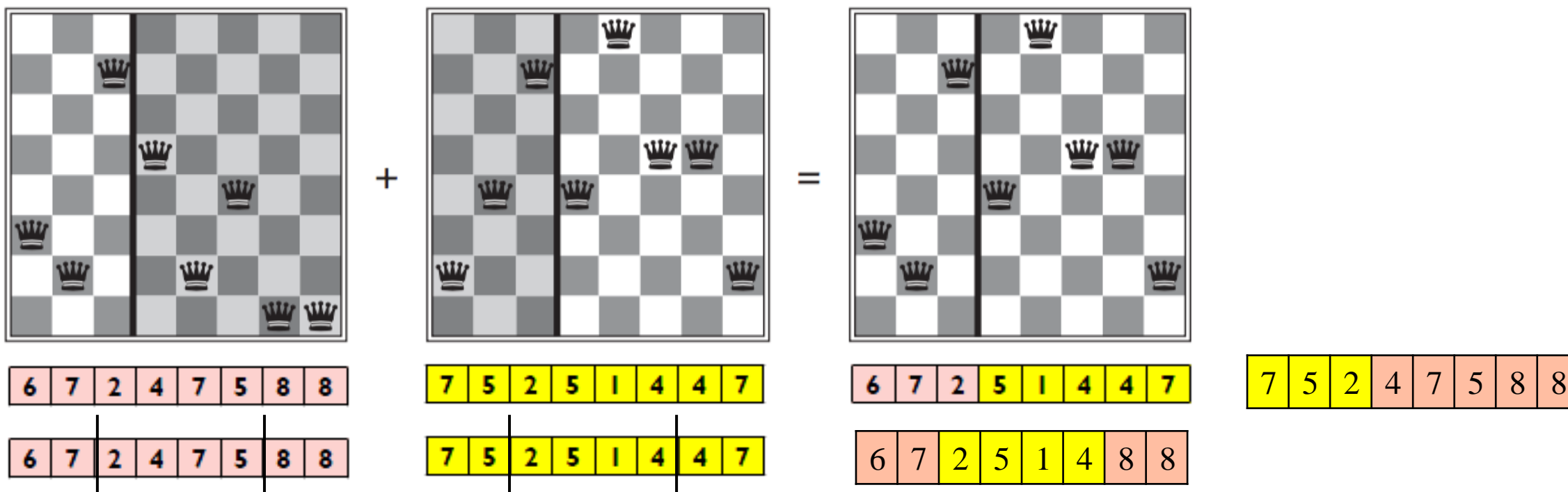
- انتخاب یک نقطه از موقعیت‌های داخل کروموزوم‌ها به‌طور تصادفی
- تولید فرزندان از برخورد کروموزوم‌های والد در نقطه‌ی پیوند



عمل ترکیب متقاطع (Crossover)

- دو کروموزوم والد به روش‌های مختلفی می‌توانند با هم ترکیب شوند و دو فرزند تولید کنند.
- ترکیب یک نقطه‌ای

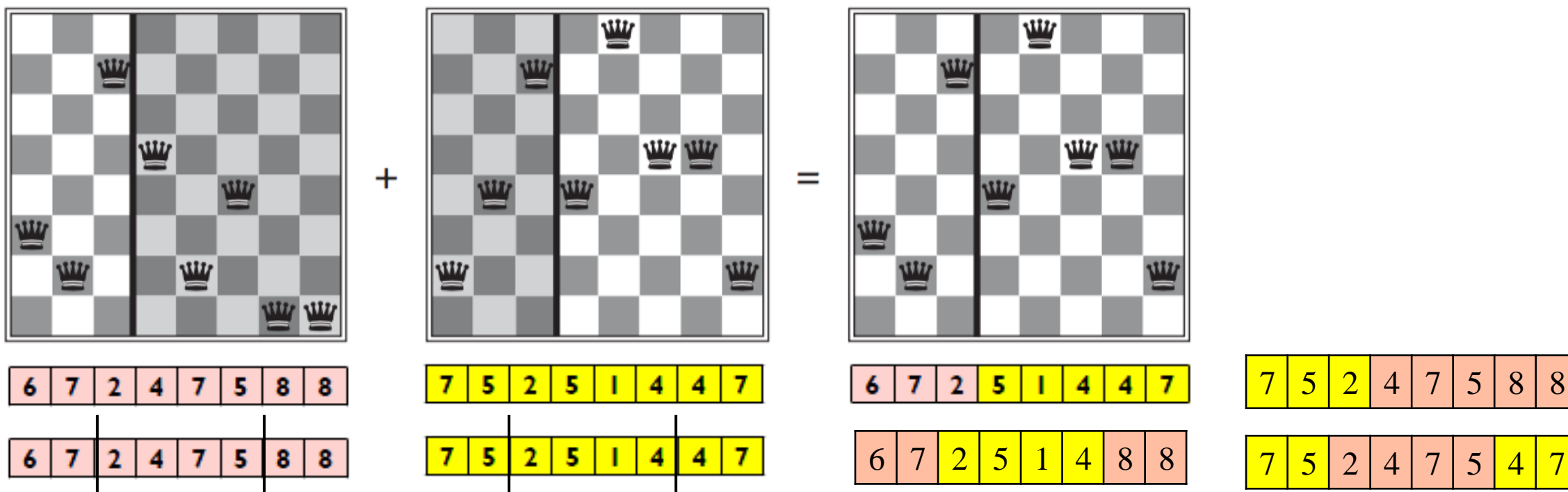
- انتخاب یک نقطه از موقعیت‌های داخل کروموزوم‌ها به‌طور تصادفی
- تولید فرزندان از برخورد کروموزوم‌های والد در نقطه‌ی پیوند



عمل ترکیب متقاطع (Crossover)

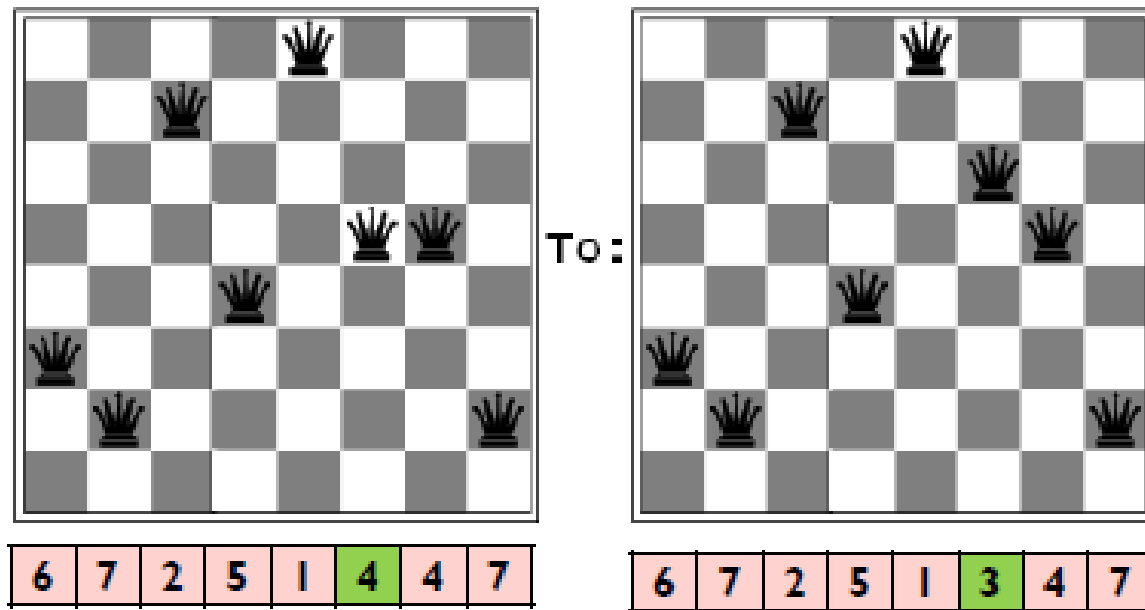
- دو کروموزوم والد به روش‌های مختلفی می‌توانند با هم ترکیب شوند و دو فرزند تولید کنند.
- ترکیب یک نقطه‌ای

- انتخاب یک نقطه از موقعیت‌های داخل کروموزوم‌ها به‌طور تصادفی
- تولید فرزندان از برخورد کروموزوم‌های والد در نقطه‌ی پیوند

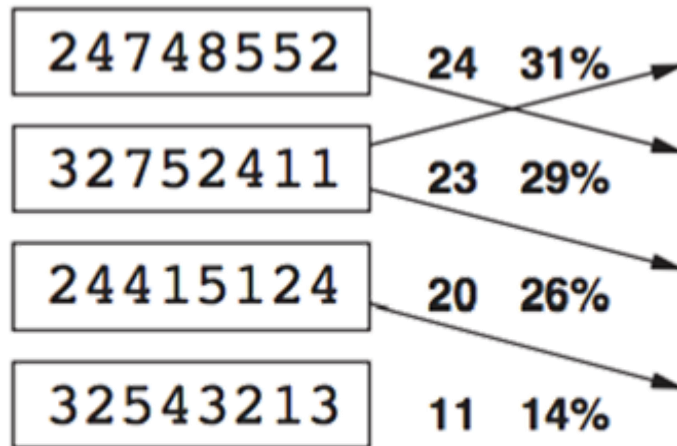


عمل جهش (Mutation)

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



تکراری از الگوریتم ژنتیک برای مسئله ۸ وزیر



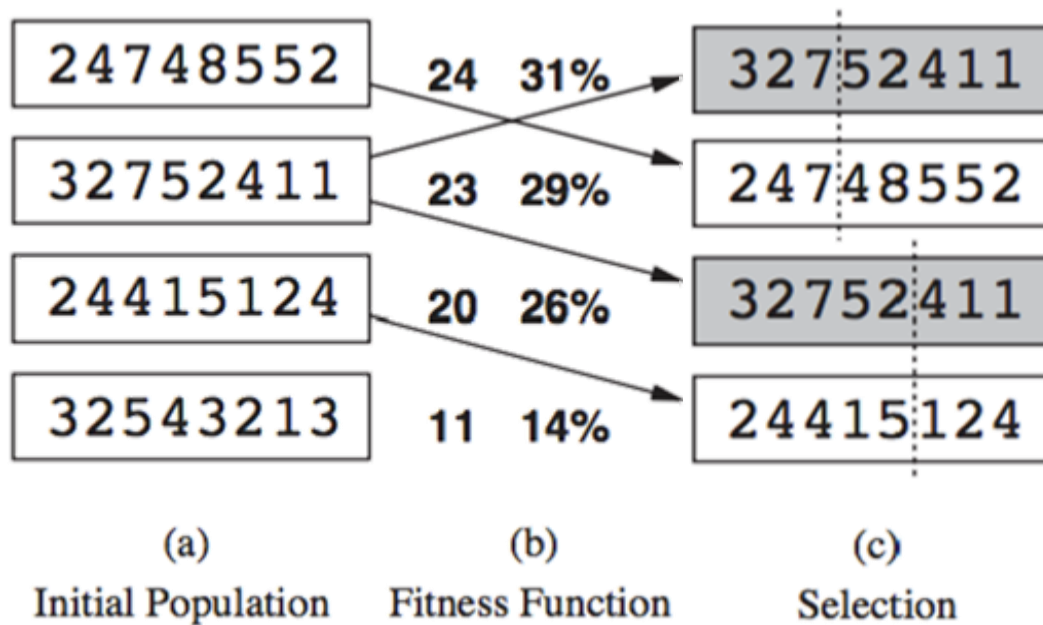
(a) Initial Population
(b) Fitness Function

• احتمال انتخاب کروموزوم i برابر است با $fitness(i) / \sum_{k=1}^n fitness(k)$

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

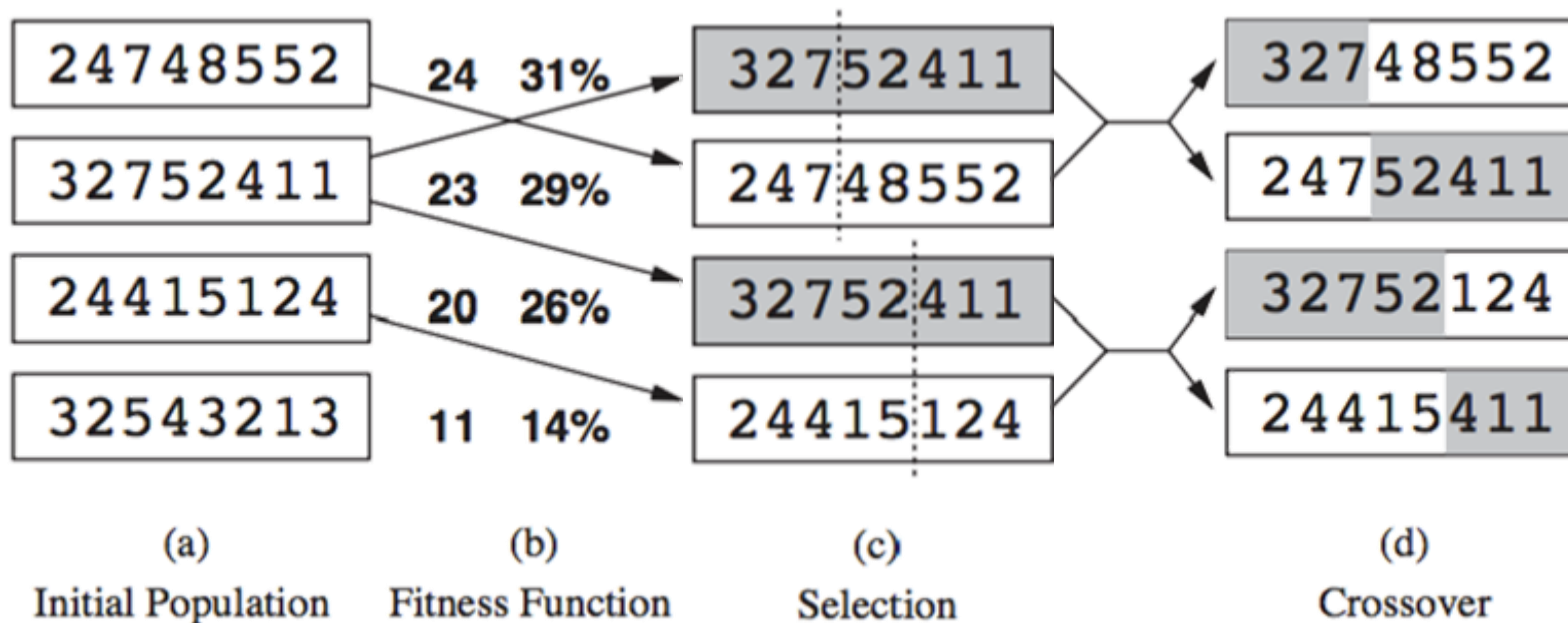
$$\%29 = 23 / (11 + 20 + 23 + 24) \bullet$$

تکراری از الگوریتم ژنتیک برای مسئله ۸ وزیر



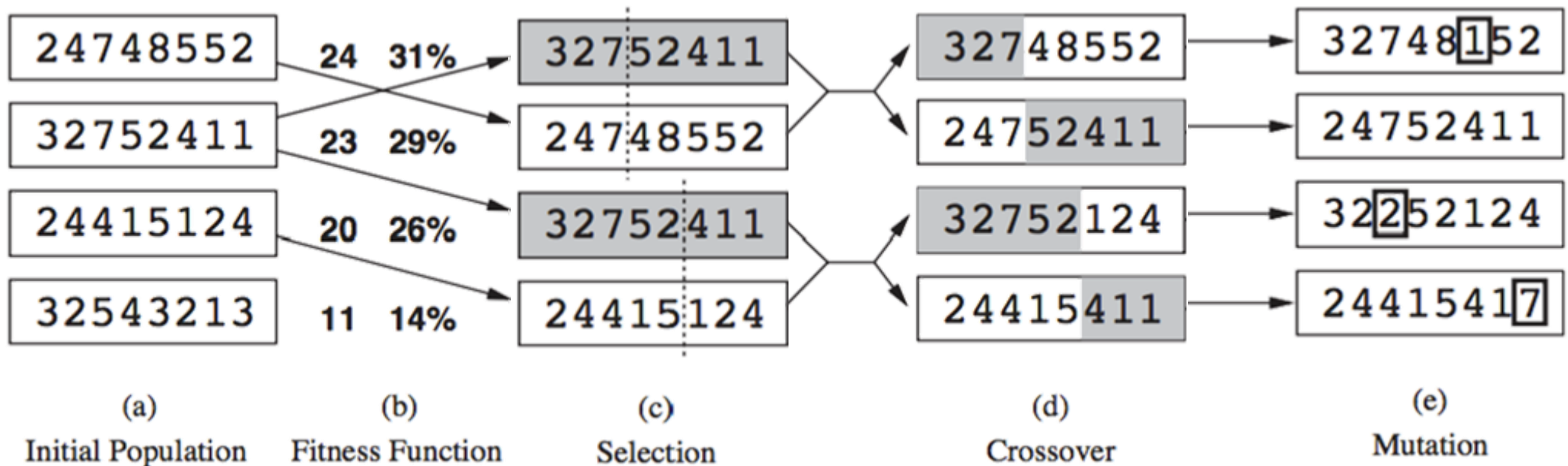
- احتمال انتخاب کروموزوم i برابر است با $fitness(i) / \sum_{k=1}^n fitness(k)$
- $\%31 = 24 / (11 + 20 + 23 + 24)$
- $\%29 = 23 / (11 + 20 + 23 + 24)$

تکراری از الگوریتم ژنتیک برای مسئله ۸ وزیر



- احتمال انتخاب کروموزوم i برابر است با $fitness(i) / \sum_{k=1}^n fitness(k)$
 - $\%31 = 24 / (11 + 20 + 23 + 24)$
 - $\%29 = 23 / (11 + 20 + 23 + 24)$

تکراری از الگوریتم ژنتیک برای مسئله ۸ وزیر



- احتمال انتخاب کروموزوم i برابر است با $fitness(i) / \sum_{k=1}^n fitness(k)$
- $\%31 = 24 / (11 + 20 + 23 + 24)$
- $\%29 = 23 / (11 + 20 + 23 + 24)$

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

- 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 ظهور می‌یابند.
- عملگر ترکیب به عنوان ویژگی مهم الگوریتم 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.

	A	B	C	D	E	F	G
1	7	5	6	3	5	6	9
2	6	5	2	1	5	5	8
3	4	4	3	4	4	5	5
4	3	4	3	3	3	1	4
5	4	2	2	1	1	1	2
6	2	3	3	2	1	3	5
7	4	4	3	3	1	5	8

بخشی از یک فضای حالت دو بعدی همراه با مقادیر تابع هدف در هر حالت در شکل زیر نشان داده شده است. هدف رسیدن به حالتی با بیشترین مقدار تابع هدف است. عامل حل مسئله می‌تواند از هر حالت با یک کنش به یکی از چهار حالت مجاور چپ، راست، بالا یا پایین برود (در صورتی که از محیط خارج نشود). خانه C5 مکان فعلی عامل را نشان می‌دهد. نحوه گذار حالات عامل با هر یک از الگوریتم‌های جستجوی زیر را نشان دهید.

الف. تپه‌نوردی با شروع مجدد تصادفی: فرض کنید به جز C5 دو شروع مجدد بعدی از حالات D2 و F5 صورت می‌گیرد. همچنین اجازه حرکت به خانه‌های مجاور با مقادیر تابع هدف یکسان با خانه فعلی وجود دارد.

ب. شبیه‌سازی ذوب فلزات: فرض کنید احتمالات مورد نیاز برای ۱۰ مرحله به ترتیب از چپ به راست به صورت زیر باشد.

$$P = 0, 1, 0, 0, 1, 0, 0, 0, 1, 0$$

ج. پرتوی موضعی با $k=3$: حالت‌های $C5$ ، $D2$ و $F5$ را به عنوان حالات اولیه در نظر بگیرید.

د. الگوریتم ژنتیک: با فرض آن که جمعیت اولیه تنها از یک عضو $C5$ تشکیل شده باشد روند الگوریتم ژنتیک به چه شکل خواهد بود؟

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 آن ها را بهبود دهد.	به طور سیستماتیک مسیرهای مختلف را امتحان می کند.	روش
پیکربندی کامل (Complete)	معمولا افزایشی (Incremental)	فضای حالت
معمولا خیلی کم (ثابت)	معمولا 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	زمان
جستجو و مسائل بهینه سازی	جستجو	حوزه