بسم الله الرحمن الرحیم

هوش مصنوعی

فصل ۴

جستجو در محیط‌های پیچیده

Search in Complex Environments

کاظم فولادی قلعه
دانشکده مهندسی، دانشکدگان فارابی
دانشگاه تهران

http://courses.fouladi.ir/ai

هوش مصنوعی

فصل ۴

جستجو در محیط‌های پیچیده (۱)

Search in Complex Environments (1)

کاظم فولادی قلعه
دانشکده مهندسی، دانشکدگان فارابی
دانشگاه تهران

http://courses.fouladi.ir/ai

هوش مصنوعی

جستجو در محیطهای پیچیده (۱)

۱

الگوریتم‌های جستجوی محلی و مسائل بهینه‌سازی

جستجو به دنبال یک حالت هدف بدون «اهمیت داشتن مسیر»

در بسیاری از مسائل بهینه‌سازی، **مسیر** رسیدن به هدف، بی‌اهمیت است؛
بلکه، خود **حالت هدف** راه‌حل مسئله است.

فضای حالت = مجموعه‌ی پیکربندی‌های «کامل»

مثال

- یافتن پیکربندی تور بهینه در مسئله‌ی فروشنده‌ی دوره‌گرد (TSP)
- یافتن پیکربندی ارضاکننده‌ی قیدها؛ مانند مسئله‌ی n-وزیر

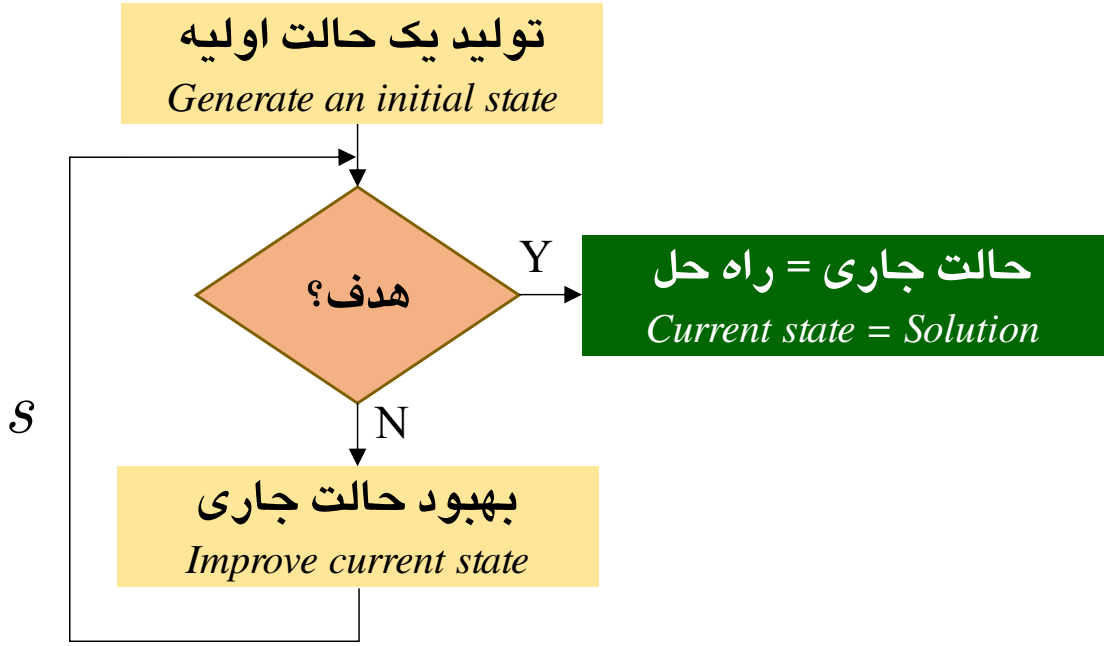
الگوریتم‌های بهبود تکراری (جستجوی محلی)

برای یافتن حالت هدف بدون «اهمیت داشتن مسیر»

ITERATIVE IMPROVEMENT ALGORITHMS (LOCAL SEARCH)

قاعده‌ی عمومی الگوریتم‌های بهبود تکراری

تولید یک حالت اولیه‌ی ساده، سپس بهبود گام به گام آن، تا رسیدن به حالت هدف



$$s[n+1] = F\{s[n]\} \quad , n = 0,1,2,\ldots$$

الگوریتم‌های بهبود تکراری (جستجوی محلی)

ویژگی‌ها

قاعده‌ی عمومی الگوریتم‌های بهبود تکراری

تولید یک حالت اولیه‌ی ساده، سپس بهبود گام به گام آن، تا رسیدن به حالت هدف

مزایای اصلی الگوریتم‌های بهبود تکراری

حافظه‌ی مصرفی پایین (معمولاً مقدار ثابت)	یافتن راه‌حل‌های مناسب در فضاهای حالت بزرگ یا نامتناهی (پیوسته)

مناسب برای حل مسائل بهینه‌سازی خالص

یافتن بهترین حالت با توجه به یک تابع هدف (objective function)

مسئله‌ی بهینه‌سازی

بهینه‌سازی
Optimization

ماکزیمم‌سازی (بیشینه‌سازی) *Maximization*	**می‌نیمم‌سازی** (کمینه‌سازی) *Minimization*
$\max\ f(x)$	$\min\ f(x)$

تابع هدف
Objective function

$$x^* = \arg\max f(x) \qquad x^* = \arg\min f(x)$$

$$\boxed{\min\ f(x) \Leftrightarrow \max\ -\ f(x)}$$

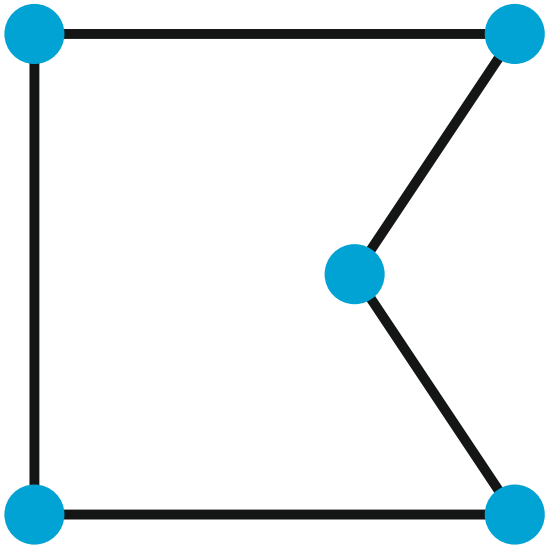
مثال: مسئله‌ی فروشنده‌ی دوره‌گرد

تعریف مسئله

TRAVELLING SALESPERSON PROBLEM (TSP)

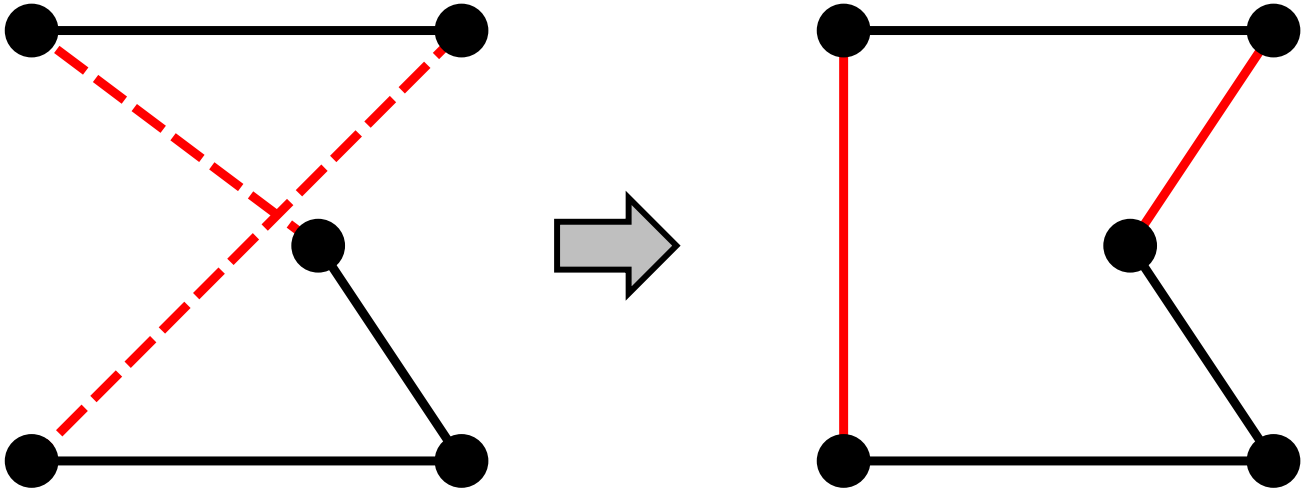
یافتن توری که هر شهر را دقیقاً یک مرتبه بپیماید.

دور بهینه (با کمترین هزینه)



مثال: راه حل مسئله‌ی فروشنده‌ی دوره‌گرد با جستجوی محلی

شروع با یک تور کامل
انجام تعویض‌های دو به دو
تا رسیدن به تور بهینه



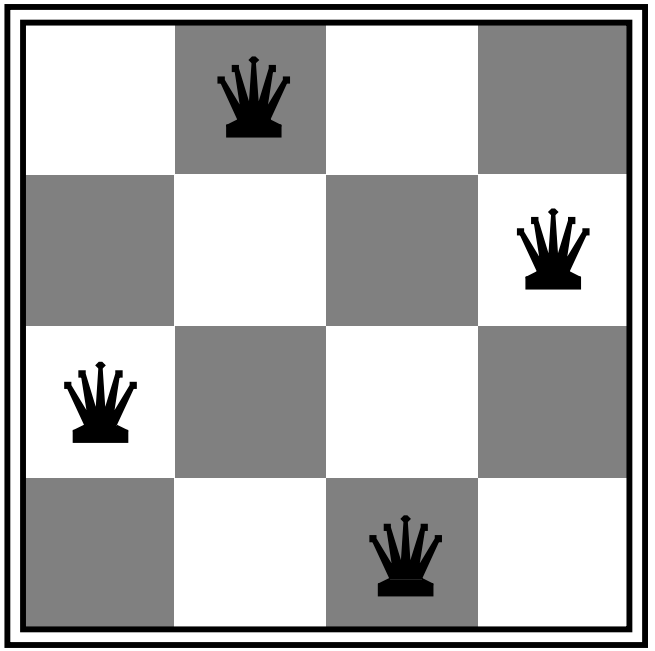
مثال: مسئله‌ی n–وزیر

تعریف مسئله

n-QUEENS

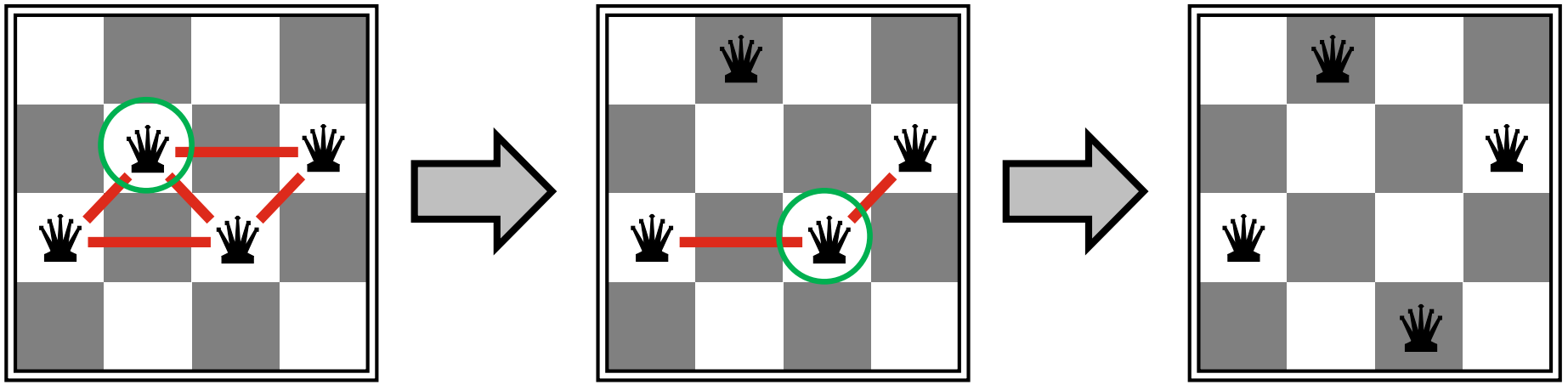
قرار دادن n مهره‌ی وزیر روی صفحه شطرنج $n \times n$ بدون اینکه یکدیگر را تهدید کنند.

نباید **هم‌سطر**، **هم‌ستون** یا **هم‌قطر** باشند.



مثال: راه حل مسئله‌ی n–وزیر با جستجوی محلی

شروع با یک ترتیب انتساب وزیرها (مثلاً هر وزیر در یک ستون)
جابجا کردن یک وزیر برای کاهش تعداد تداخل‌ها
تا رسیدن به حالت بدون تداخل



با این روش، مسئله‌ی n–وزیر برای n بسیار بزرگ (در حدود یک میلیون) تقریباً به طور آنی حل می‌شود.

هوش مصنوعی

جستجو در محیطهای پیچیده (۱)

۲

تپه‌نوردی

تپه‌نوردی (افزایش/کاهش گرادیانی)

پایه‌ای‌ترین تکنیک جستجوی محلی

HILL-CLIMBING (GRADIENT ASCENT/DESCENT)

مشابه بالا رفتن از اورست در مِه غلیظ با داشتن فراموشی

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

  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
```

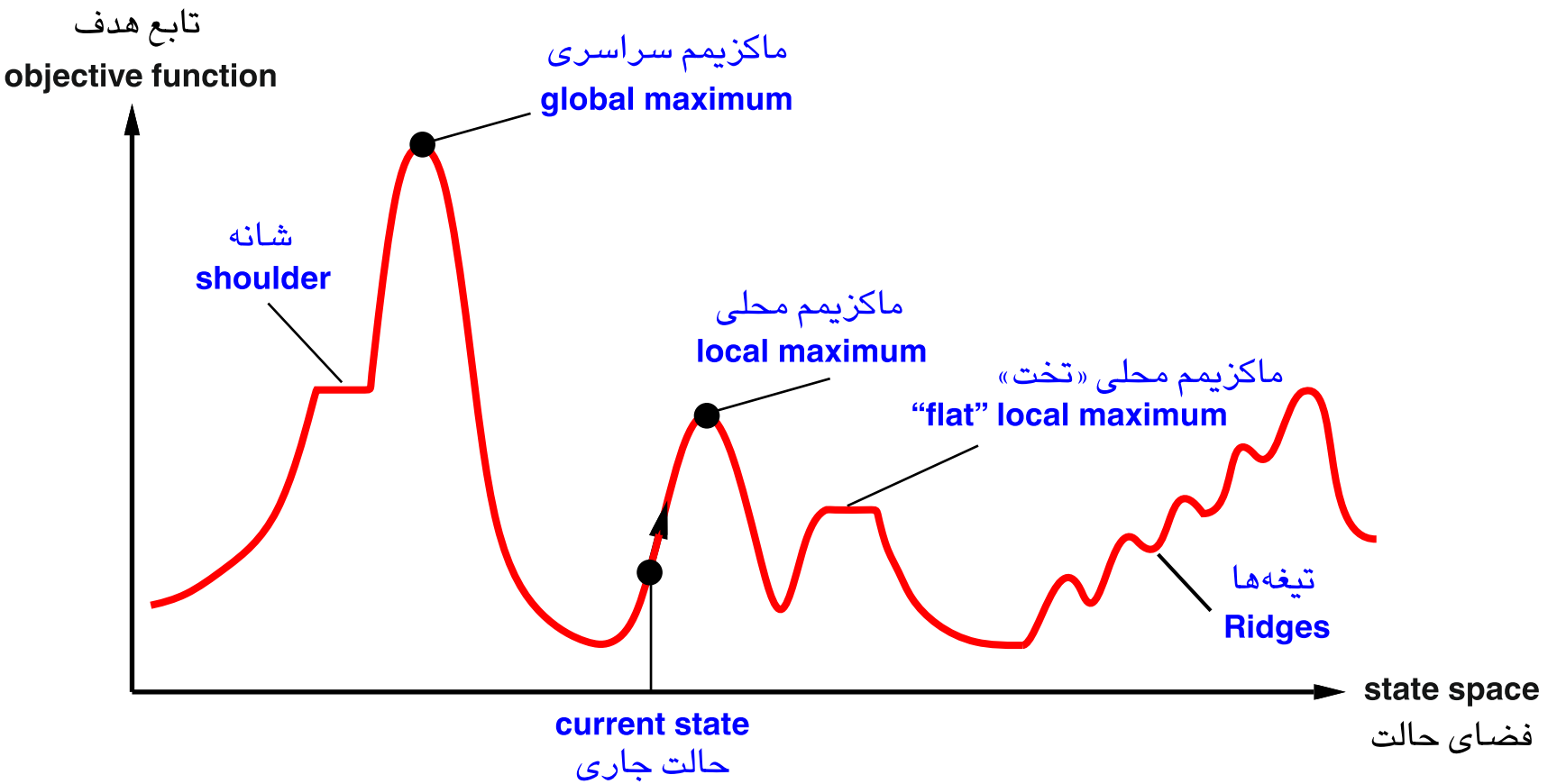
قاعده‌ی تپه‌نوردی: حرکت در مسیر افزایش تابع ارزیابی تا جای ممکن

نام دیگر: الگوریتم جستجوی محلی حریصانه

تپه‌نوردی معمولاً خیلی سریع به سوی یک راه‌حل پیشرفت می‌کند،
زیرا معمولاً به طور کاملاً ساده‌ای یک حالت بد را بهبود می‌دهد.

تپه‌نوردی

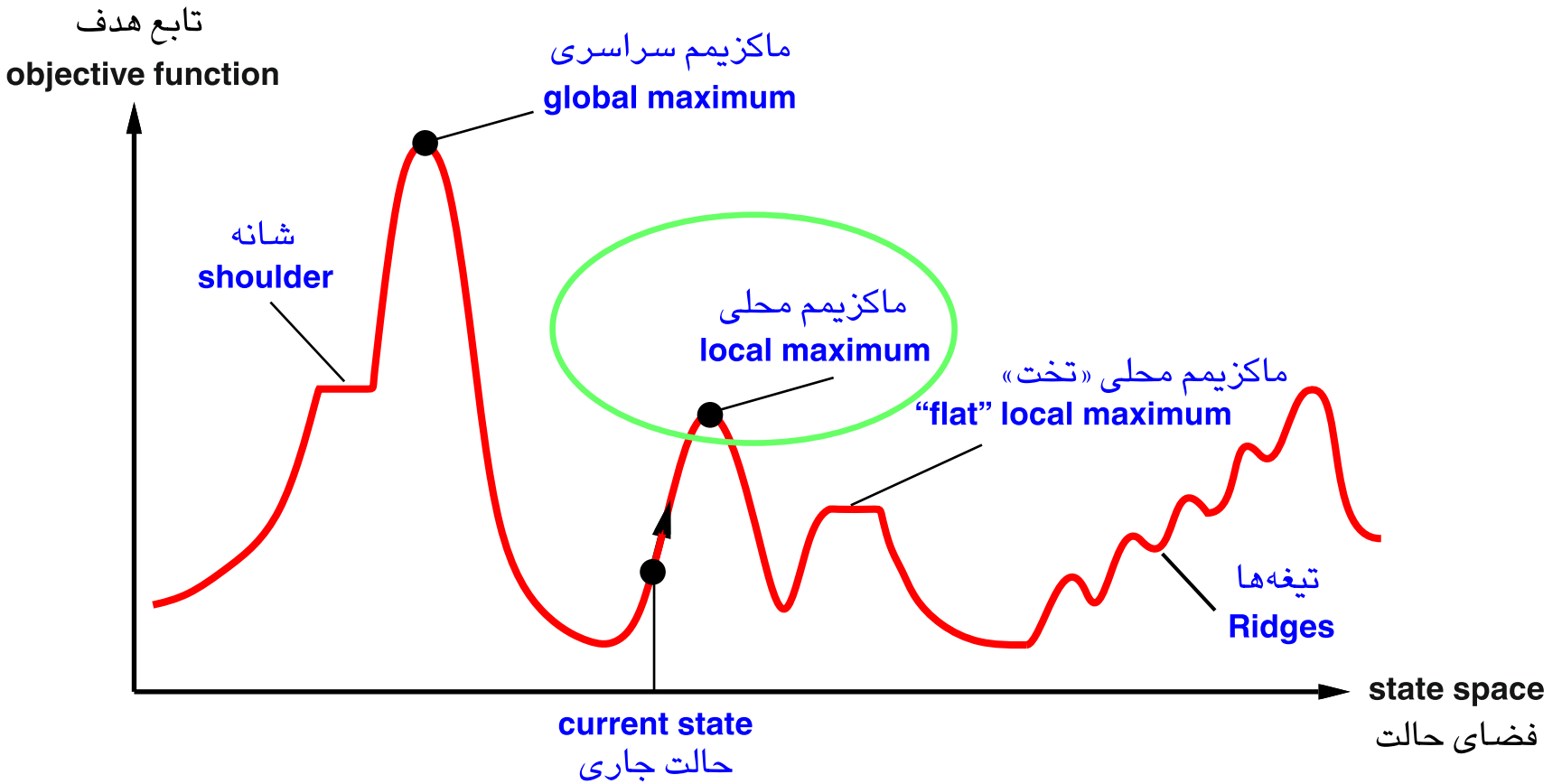
نمونه‌ی چشم‌انداز فضای حالت



تپه‌نوردی

مشکل: گیرافتادن در ماکزیمم محلی

ماکزیمم محلی، قله‌ای است که بلندتر از هر یک از همسایگانش است، اما کوتاه‌تر از ماکزیمم سراسری است.

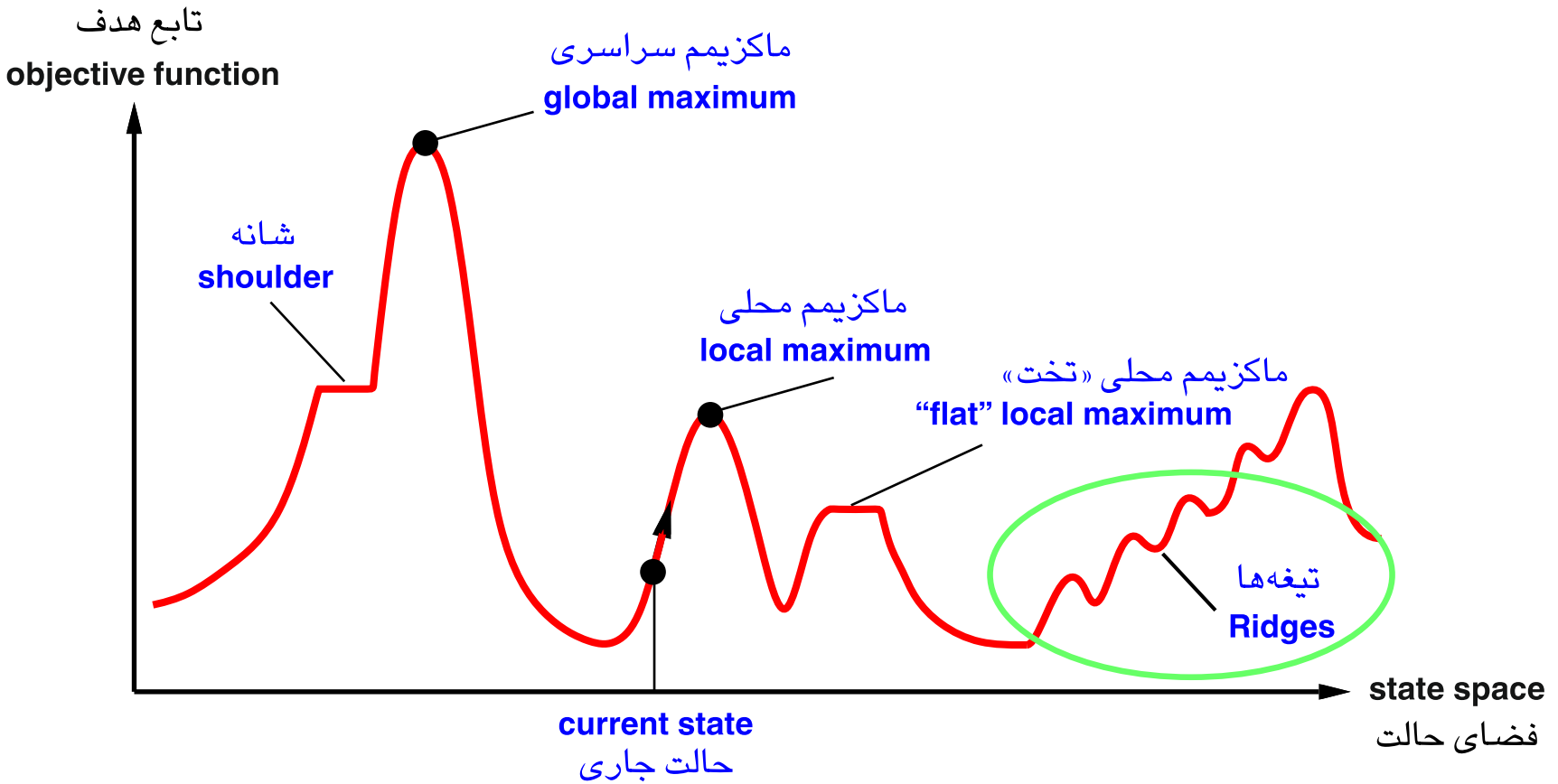


تپه‌نوردی

مشکل: گیرافتادن در تیغه‌ها

تیغه‌ها، حاصل یک دنباله از ماکزیمم‌های محلی هستند

که عبور از آن برای الگوریتم‌های حریصانه بسیار دشوار است.



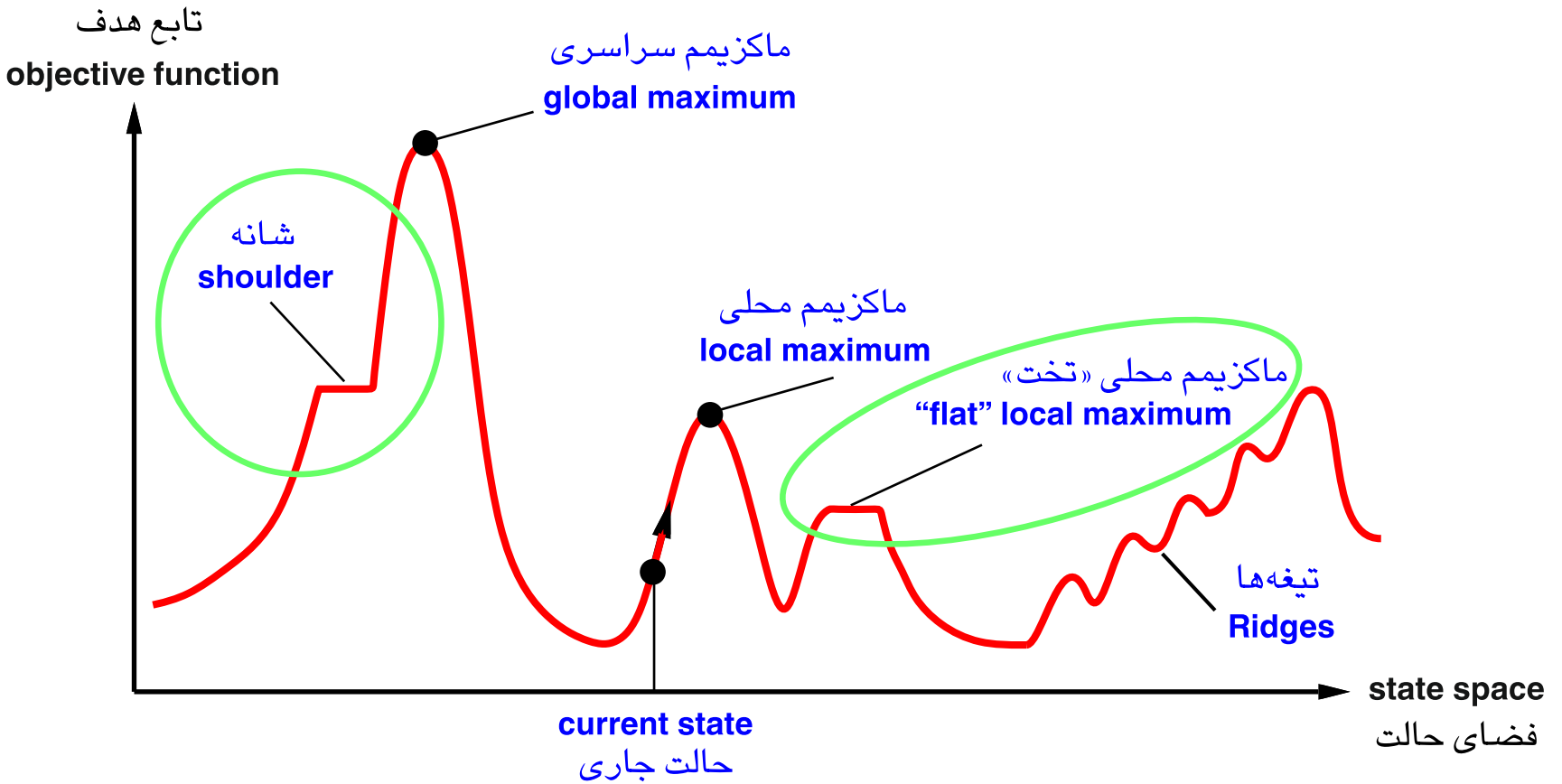
تپه‌نوردی

مشکل: گیرافتادن در فلات‌ها

PLATEAUX

فلات‌ها، ناحیه‌ای از فضای حالت هستند که در آن تابع ارزیابی ثابت است.

فلات می‌تواند یک **ماکزیمم محلی تخت** باشد (که هیچ مسیر رو به بالایی ندارد) یا یک **شانه** باشد (که بتوان از آن بالاتر هم رفت).



تپه‌نوردی

مشکلات

- گیر افتادن در ماکزیمم محلی
- گیر افتادن در تیغه‌ها
- گیر افتادن در فلات‌ها

تپه‌نوردی

مشکل تیغه‌ها در توابع چندمتغیره

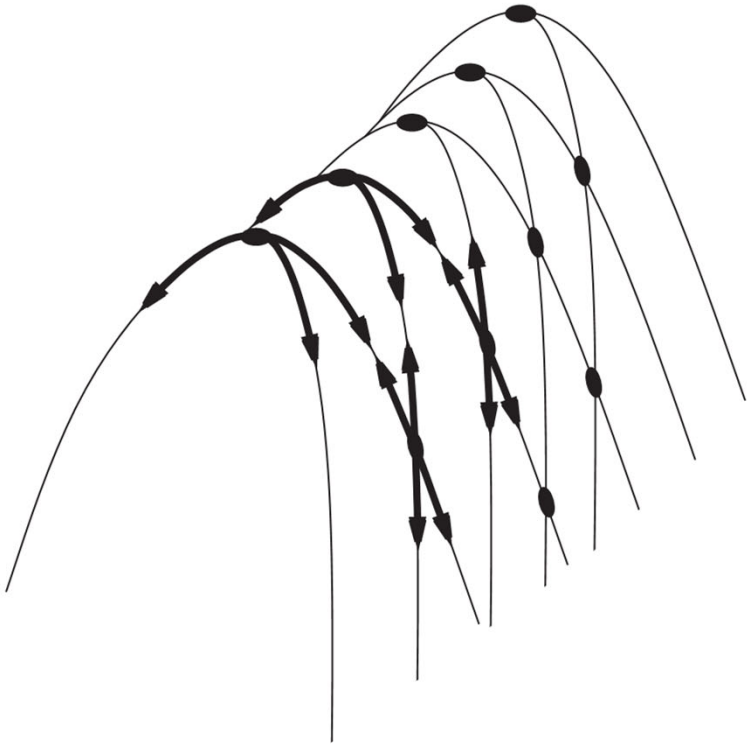


Figure 4.4 Illustration of why ridges cause difficulties for hill climbing. The grid of states (dark circles) is superimposed on a ridge rising from left to right, creating a sequence of local maxima that are not directly connected to each other. From each local maximum, all the available actions point downhill.

گزینه‌هایی برای حل مشکلات تپه‌نوردی

شکل‌های دیگر تپه‌نوردی

تپه‌نوردی اتفاقی
Stochastic hill-climbing

انتخاب تصادفی از میان گزینه‌های حرکت به سمت بالا

معمولا کندتر از «تندترین فراز» همگرا می‌شود،
اما در بعضی فضاهای حالت راه‌حل‌های بهتری می‌یابد.

تپه‌نوردی نخستین–گزینه
First-choice hill-climbing

نوعی پیاده‌سازی **تپه‌نوردی اتفاقی**:
مابعدهای یک حالت به صورت تصادفی یکی پس دیگری، آن‌قدر تولید می‌شوند تا حالتی تولید شود که بهتر از حالت جاری باشد.

مناسب برای زمانی که یک حالت تعداد زیادی مابعد دارد.

تپه‌نوردی با شروع مجدد تصادفی
Random-restart hill-climbing

یک مجموعه جستجوی تپه‌نوردی با حالت‌های تصادفی شروع مختلف اجرا می‌کند و با پیدا شدن هدف توقف می‌کند.

بر مشکل ماکزیمم‌های محلی غلبه می‌کند؛ با احتمال یک **کامل** است.
حرکت‌های تصادفی یک‌سویه: ☺ فرار از شانه‌ها ☹ افتادن در حلقه در ماکزیمم‌های تخت

تپه‌نوردی

افزایش/کاهش گرادیانی

تپه‌نوردی *Hill-climbing*	
ماکزیمم‌سازی (بیشینه‌سازی) *Maximization*	**می‌نیمم‌سازی** (کمینه‌سازی) *Minimization*
افزایش گرادیانی *Gradient Ascent*	**کاهش گرادیانی** *Gradient Descent*
تندترین فراز *Steepest Ascent*	**تندترین شیب** *Steepest Descent*
حرکت در جهت تندترین شیب/گرادیان صعودی تابع هدف	حرکت در جهت تندترین شیب/گرادیان نزولی تابع هدف

* **گرادیان** برای هر تابع چندمتغیره، برداری است که جهت بیشترین تغییرات تابع در هر نقطه را نشان می‌دهد.

هوش مصنوعی

جستجو در محیطهای پیچیده (۱)

۳

تافت شبیه‌سازی شده

تافت شبیه‌سازی‌شده

SIMULATED ANNEALING

فرآیند تافت در متالورژی:

فرآیند مورد استفاده برای تاب دادن یا سخت کردن فلزات و شیشه‌ها
با گرما دادن به آنها تا یک دمای بالا و سپس خنک کردن تدریجی آنها
⇐ به ماده امکان داده می‌شود تا در یک حالت کریستالی با انرژی پایین شکل بگیرد.

آنالوژی	
فلز *Metal*	**مسئله** *Problem*
حالت انرژی *Energy State*	**تابع هزینه (تابع ارزیابی)** *Cost function (Evaluation function)*
دما *Temperature*	**پارامتر کنترلی** *Control parameter*
حالت کریستالی با انرژی پایین *Low-energy crystalline state*	**راه‌حل بهینه‌ی مسئله** *Optimal solution of problem*

نمونه‌ای از الگوریتم‌های الهام‌گرفته شده از طبیعت
Nature-inspired

تافت شبیه‌سازی‌شده

مثال

SIMULATED ANNEALING

فرآیند تولید بستنی:

هم زدن مواد بستنی (گرما دادن به آنها)

قرار دادن در فریزر (خنک کردن تدریجی)

سپس هم زدن مجدد و قرار دادن مجدد در فریزر برای یک مدت بیشتر

تکرار عملیات فوق: هر بار میزان هم زدن کم می‌شود.

⇐ به مواد بستنی امکان داده می‌شود تا در یک حالت انرژی پایین ببندد.

تافت شبیه‌سازی‌شده

مثال

SIMULATED ANNEALING

توپ در یک ظرف موج‌دار:

می‌خواهیم یک توپ را در یک ظرف پر از شیار و پستی–بلندی به عمیق‌ترین نقطه برسانیم.

تکان دادن ظرف (گرما دادن به آن)

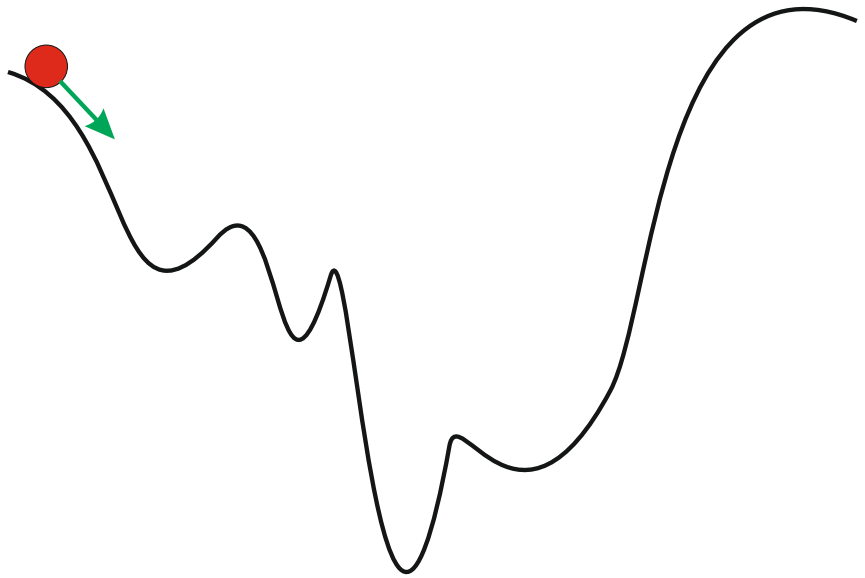
اجازه داده به توپ برای حرکت (خنک کردن تدریجی)

سپس تکان دادن مجدد و اجازه دادن مجدد به توپ برای یک مدت بیشتر

تکرار عملیات فوق: هر بار مدت تکان دادن و شدت آن کمتر می‌شود.

⇐ به توپ امکان داده می‌شود تا به یک حالت انرژی پایین (کمترین ارتفاع) برسد.

تکان دادن ظرف: ابتدا زیاد و به مرور تعداد و اندازه‌ی تکان‌ها کم می‌شود.



تافت شبیه‌سازی‌شده

SIMULATED ANNEALING

فرار از ماکزیمم محلی، با اجازه دادن به مقداری حرکت «بد»
اما به طور تدریجی، اندازه و تعداد حرکت‌های بد کاهش می‌یابد.

function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state
  inputs: *problem*, a problem
    schedule, a mapping from time to "temperature"

  current ← MAKE-NODE(*problem*.INITIAL-STATE)
  for $t = 1$ **to** ∞ **do**
    $T \leftarrow schedule(t)$
    if $T = 0$ **then return** *current*
    next ← a randomly selected successor of *current*
    $\Delta E \leftarrow next.\text{VALUE} - current.\text{VALUE}$
    if $\Delta E > 0$ **then** *current* ← *next*
    else *current* ← *next* only with probability $e^{\Delta E/T}$

تافت شبیه‌سازی‌شده

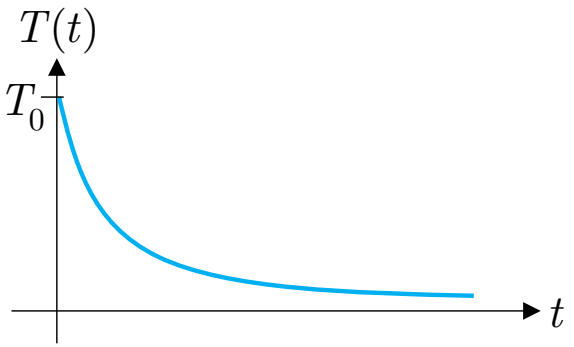
«دما» به عنوان متغیر کنترلی

با متغیر دما میزان اجازه دادن به حرکت‌های بد کنترل می‌شود:

هر چه دما کمتر باشد، میزان حرکت‌های بد کاهش می‌یابد.

با گذر زمان، باید احتمال حرکت‌های بد کاهش یابد.

دما را به عنوان تابعی نزولی از زمان در نظر می‌گیریم.



ثابت می‌شود که اگر دما به اندازه‌ی کافی آهسته کاهش یابد، الگوریتم با احتمال یک به بهینه‌ی سراسری می‌رسد.

تافت شبیه‌سازی‌شده

انواع گوناگون

تافت شبیه‌سازی‌شده استاندارد (تافت بولتزمن)
Standard/Boltzmann Annealing (SA)

استفاده از توزیع احتمال بولتزمن $p(x) = \alpha e^{E(x)/kT}$

شرط آماری یافتن بهینه‌ی سراسری:
- دما T سریع‌تر از $T(k) = T_0/\ln k$ کاهش نیابد.
- دمای اولیه T_0 به اندازه‌ی کافی بزرگ باشد.

☺ تضمین همگرایی احتمالاتی به راه‌حل بهینه ☹ سرعت کند در عمل

اطفای شبیه‌سازی‌شده
Simulated Quenching (SQ)

استفاده از تابع زمان‌بندی دمای نمایی

$$T_{n+1} = T_n c \quad ,0 < c < 1 \Rightarrow T_n = T_0 \exp[(c-1)n]$$

☹ عدم تضمین همگرایی به راه‌حل بهینه ☺ نتایج عملی خوب

تافت سریع
Fast Annealing (FA)

استفاده از زمان‌بندی دمای به طور نمایی سریع‌تر از SA استاندارد

$$T_n = T_0 \ / \ n$$

☺ تضمین همگرایی احتمالاتی به راه‌حل بهینه ☺ سرعت بیشتر در عمل

تافت شبیه‌سازی‌شده وفقی
Adaptive Simulated Annealing (ASA)

بازتافت شبیه‌سازی‌شده‌ی بسیار سریع (Very Fast Simulated Re-annealing)
- قابل استفاده برای فضاهای پارامتر چندبعدی یا دامنه‌های مختلف برای پارامترها
- استفاده از توابع زمان‌بندی دمای مختلف برای هر پارامتر
- قادر به اجرای عمل اطفا + بازتافت

☺ تضمین همگرایی احتمالاتی به راه‌حل بهینه

هوش مصنوعی

فصل ۴

جستجو در محیط‌های پیچیده (۲)

Search in Complex Environments (2)

کاظم فولادی قلعه
دانشکده مهندسی، دانشکدگان فارابی
دانشگاه تهران

http://courses.fouladi.ir/ai

هوش مصنوعی

جستجو در محیطهای پیچیده (۲)

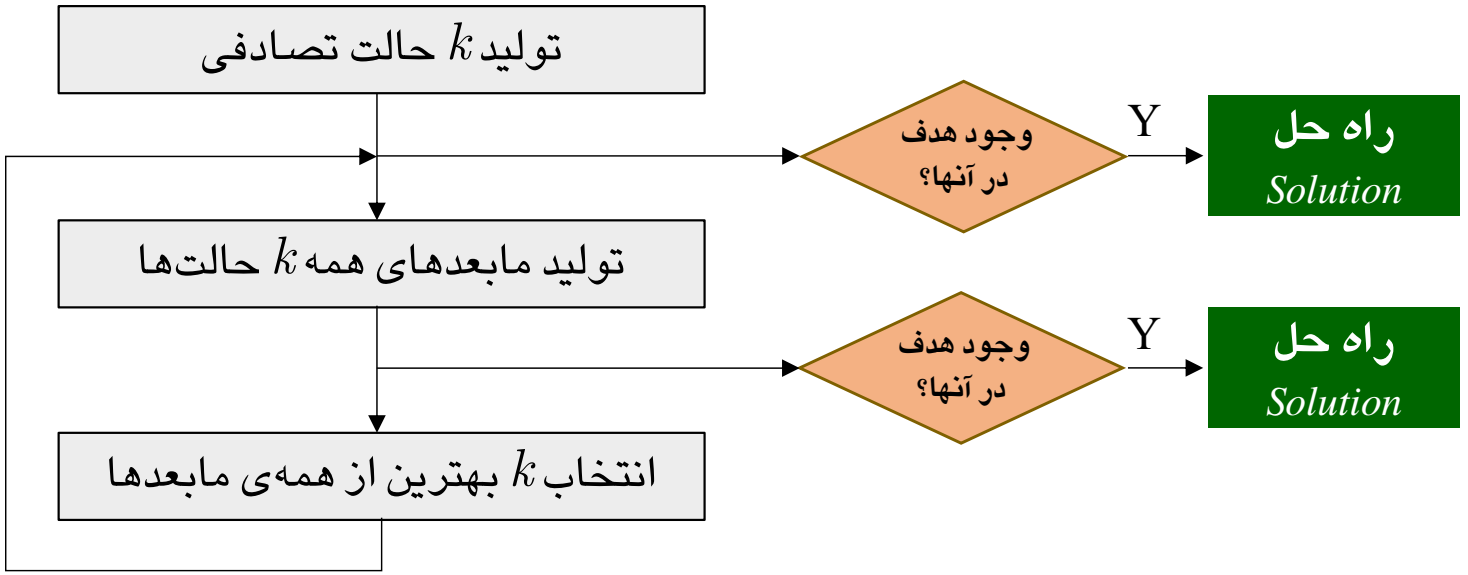
۱

جستجوی پرتوی محلی

جستجوی پرتوی محلی

LOCAL BEAM-SEARCH

ایده: نگهداری k حالت به‌جای ۱ حالت + انتخاب k مابعد بهتر هر حالت



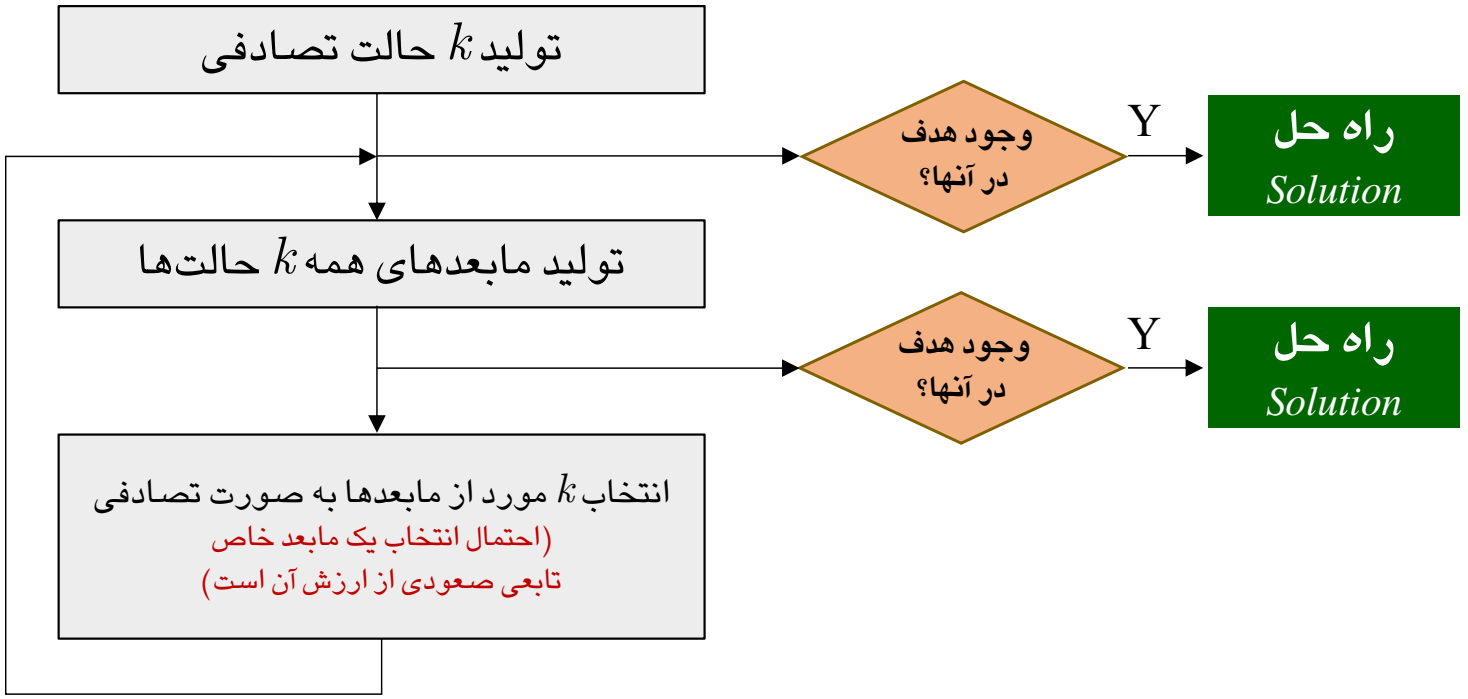
این الگوریتم با اجرای k جستجوی موازی با شروع تصادفی متفاوت است:
این جستجوها مستقل از هم نیستند و اطلاعات سودمند بین آنها رد و بدل می‌شود:
جستجوهایی که حالت‌های خوبی پیدا می‌کنند، دیگر جستجوها را تشویق می‌کنند تا به آنها بپیوندند.

مشکل: اغلب همه‌ی k حالت به یک تپه‌ی محلی منجر می‌شوند!: نبود تنوع در میان حالت‌ها

جستجوی پرتوی محلی اتفاقی

STOCHASTIC LOCAL BEAM-SEARCH

ایده: نگهداری k حالت به‌جای ۱ حالت + انتخاب تصادفی k مابعد از حالت‌ها



وجود شباهت با فرآیند انتخاب طبیعی (Natural selection):

مابعدها: **فرزندان** (offspring) حالت: **موجود زنده** (organism) ارزش: **میزان برازش** (fitness)

هوش مصنوعی

جستجو در محیطهای پیچیده (۳)

۲

الگوریتم‌های تطوری (تکاملی)

الگوریتم‌های تطوری (تکاملی)

EVOLUTIONARY ALGORITHMS

الگوریتم‌های تطوری
Evolutionary Algorithms

به‌عنوان شکل‌های تغییریافته از جستجوی پرتوی اتفاقی
که به‌صورت صریح از استعاره‌ی **انتخاب طبیعی** در زیست‌شناسی انگیزه یافته است.

یک **جمعیت** از افراد (حالت‌ها) وجود دارد،
برازنده‌ترین افراد (بالاترین مقادیر) فرزندان (حالت‌های مابعد) را تولید می‌کنند؛
که جمعیت نسل بعدی را تکثیر می‌کنند (در فرآیندی که **بازترکیب** نامیده می‌شود).

شکل‌های بی‌پایانی از الگوریتم‌های تطوری وجود دارد؛ بر اساس
* اندازه‌ی جمعیت * بازنمایی هر فرد * روال بازترکیب * نرخ جهش * آرایش نسل جدید

الگوریتم‌های تطوری

EVOLUTIONARY ALGORITHMS

الگوریتم‌های تطوری
Evolutionary Algorithms

انواع استراتژی‌های تطوری بر اساس بازنمایی فرد		
	الگوریتم‌های ژنتیک *Genetic Algorithm*	هر فرد یک رشته روی یک الفبای متناهی است (معمولاً یک رشته‌ی بیتی) [دقیقاً مانند DNA که رشته‌ای روی الفبای ACGT است]
	استراتژی‌های تطوری *Evolution Strategies*	هر فرد یک دنباله از اعداد حقیقی است.
	برنامه‌نویسی ژنتیک *Genetic Programming*	هر فرد یک برنامه‌ی کامپیوتری است.

الگوریتم‌های ژنتیک

GENETIC ALGORITHMS (GA)

معرفی رسمی توسط John Holland (1970s)

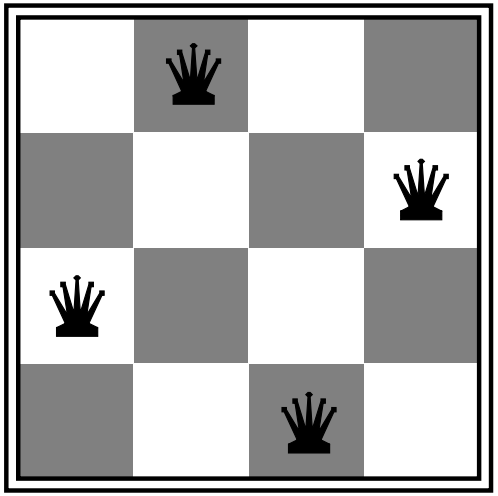
مشخصه‌های الگوریتم‌های ژنتیک	
✓ روش جستجوی اتفاقی	✗ روش جستجوی قطعی
✓ کار بر روی یک جمعیت از راه‌حل‌ها	✗ کار بر روی یک راه‌حل
✓ روش جستجوی آگاهانه	✗ روش جستجوی ناآگاهانه
✓ تفکیک الگوریتم از بازنمایی	✗ یکپارچگی الگوریتم با بازنمایی

الگوریتم‌های ژنتیک

کروموزوم (ژنوم)

کروموزوم *Chromosome*	یک ساختمان‌داده برای کدگذاری راه‌حل‌های مسئله (حالت‌ها) برای ذخیره‌سازی در کامپیوتر (معمولاً **رشته/بردار**)
ژنوم *Genome*	
فرد *Individual*	

مثال



$$(3, 1, 4, 2)$$

الگوریتم‌های ژنتیک

جمعیت

جمعیت *Population*	مجموعه‌ای از افراد (کروموزوم‌ها)

الگوریتم‌های ژنتیک

انتخاب

انتخاب *Selection*	انتخاب افراد (کروموزوم‌ها) برای جفت شدن

ضوابط متنوعی برای **گزینش بهترین افراد برای جفت شدن** وجود دارد.

الگوریتم‌های ژنتیک

بازترکیب (مانند تقاطع)

بازترکیب *Recombination*	ترکیب مجدد والدها برای تولید فرزندان
حالت خاص ↓	
تقاطع *Crossover*	ترکیب ضربدری دو کروموزوم والد برای تولید دو یا چند فرزند

تقاطع می‌تواند به صورت (**جنسی** یا **غیرجنسی**) / و حتی **تک‌فرزندی** تعریف شود.

الگوریتم‌های ژنتیک

جهش

جهش *Mutation*	تغییری تصادفی در یک کروموزوم

- جهش مقدار مشخصی **تصادفی بودن** به جستجو اضافه می‌کند.
- جهش به جستجو کمک می‌کند تا راه‌حل‌هایی که *تقاطع* به تنهایی به آنها برخورد نمی‌کند پیدا شوند.

الگوریتم‌های ژنتیک

تابع برازش (تابع هدف)

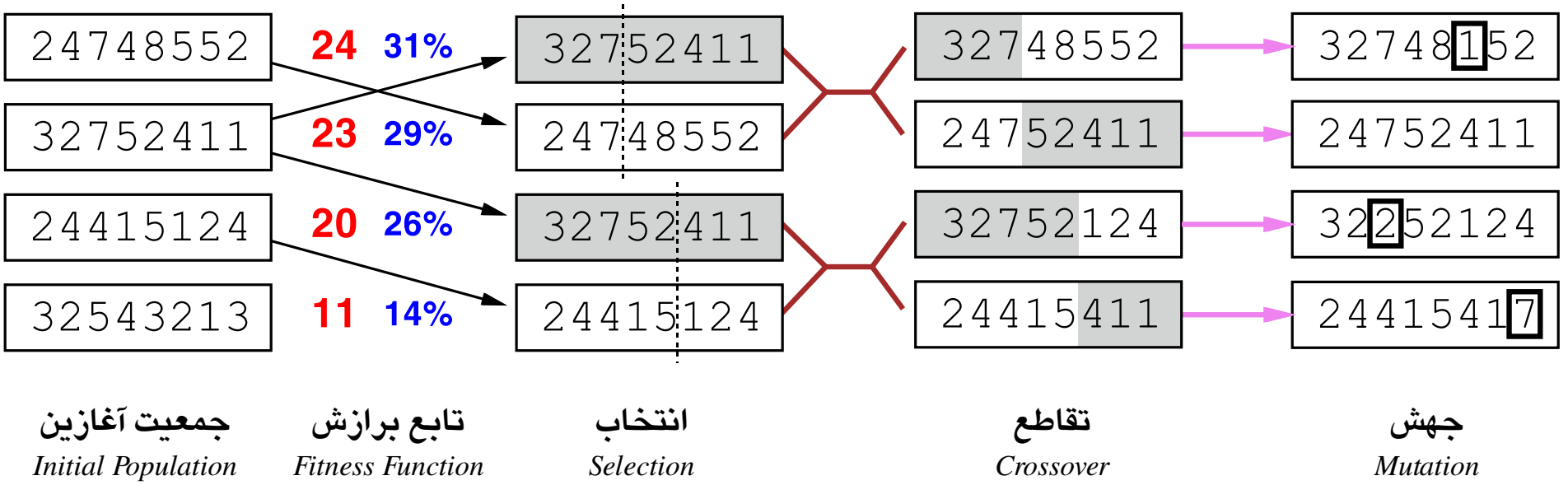
تابع برازش *Fitness Function*	تابعی که میزان خوب بودن یک فرد (کروموزوم) را نشان می‌دهد.
تابع هدف *Objective Function*	

هدف، ماکزیمم‌سازی تابع برازش است.

الگوریتم‌های ژنتیک

مثال

الگوریتم ژنتیک = جستجوی پرتوی محلی اتفاقی + تولید مابعدها از روی **جفت‌هایی** از حالت‌ها



نمونه‌ی الگوریتم ژنتیک، ارائه شده برای بازنمایی حالت‌های مسئله‌ی ۸-وزیر در قالب رشته‌ی عددی
هر عدد برای یک ستون صفحه‌ی شطرنج و عدد مربوطه شماره‌ی سطر است.

الگوریتم ژنتیک

شبه کد

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 ← empty set
    for $i = 1$ **to** SIZE(*population*) **do**
      $x \leftarrow$ RANDOM-SELECTION(*population*, FITNESS-FN)
      $y \leftarrow$ RANDOM-SELECTION(*population*, FITNESS-FN)
      child ← REPRODUCE(x, y)
      if (small random probability) **then** *child* ← MUTATE(*child*)
      add *child* to *new_population*
    population ← *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

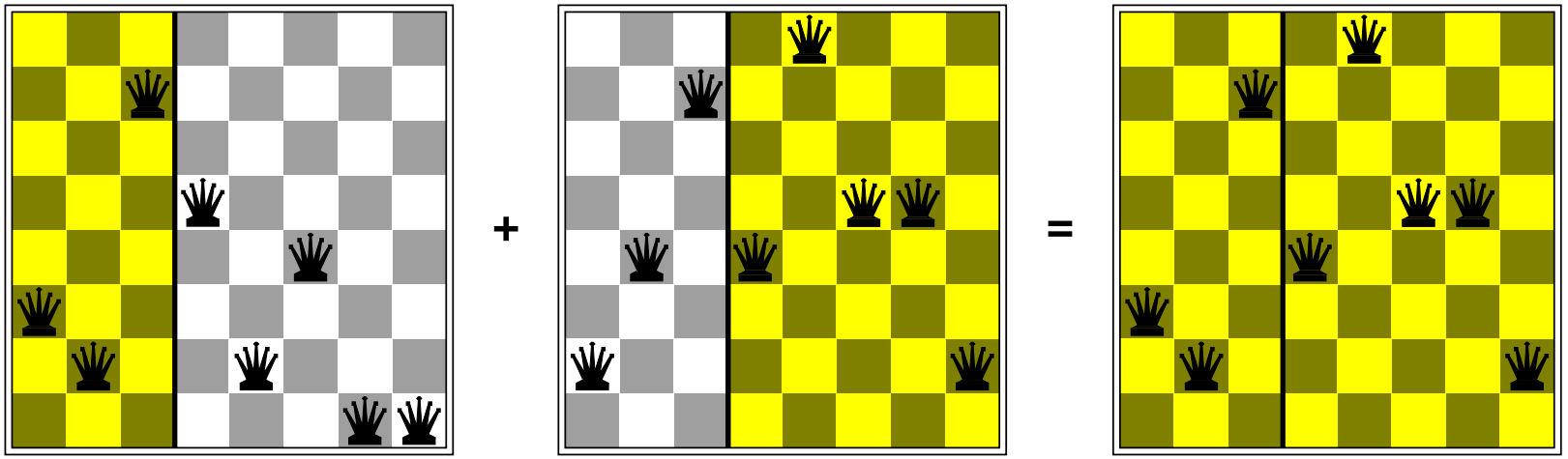
  $n \leftarrow$ LENGTH(x); $c \leftarrow$ random number from 1 to n
  return APPEND(SUBSTRING($x, 1, c$), SUBSTRING($y, c + 1, n$))

الگوریتم ژنتیک

بازنمایی حالت‌ها

حالت‌ها (کروموزوم‌ها) در الگوریتم ژنتیک به صورت **رشته‌ها** کدگذاری می‌شوند.

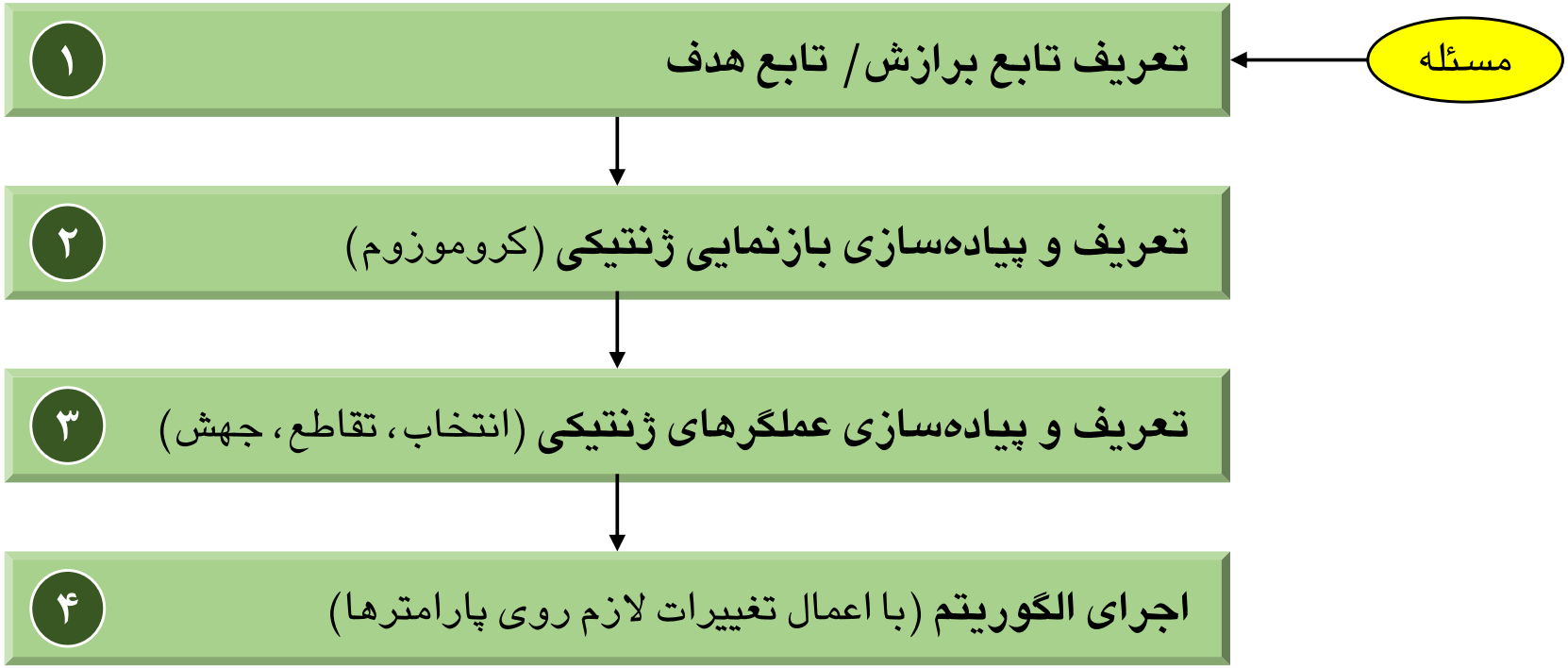
تقاطع (crossover) در صورتی کمک می‌کند که **زیررشته‌ها** اجزای معناداری باشند.



مثال: در مسئله‌ی ۸-وزیر، رشته‌ی عددی دارای زیررشته‌های معنادار است.

الگوریتم ژنتیک

مراحل استفاده از الگوریتم ژنتیک برای حل مسئله



شکل‌های متنوعی از GA وجود دارد، اما اگر تابع برازش، کروموزوم‌ها و عملگرهای ژنتیکی به خوبی تعریف شده باشند، تغییر شکل الگوریتم معمولاً بهبودهای اندکی ایجاد می‌کند.

الگوریتم‌های ژنتیک

مزایا و معایب

مزایا و معایب الگوریتم‌های ژنتیک	
مزایا	معایب
سادگی بالا	سرعت پایین
عملکرد خوب روی انواع مختلفی از مسائل	محاسبات سنگین
عملکرد خوب روی فضاهای هیبرید (پیوسته/گسسته)	عدم وفق‌یابی مناسب با موقعیت‌های جدید
شک کمتر به گیر افتادن در بهینه‌های محلی	

هوش مصنوعی

جستجو در محیطهای پیچیده (۲)

۳

جستجوی محلی در فضاهای پیوسته

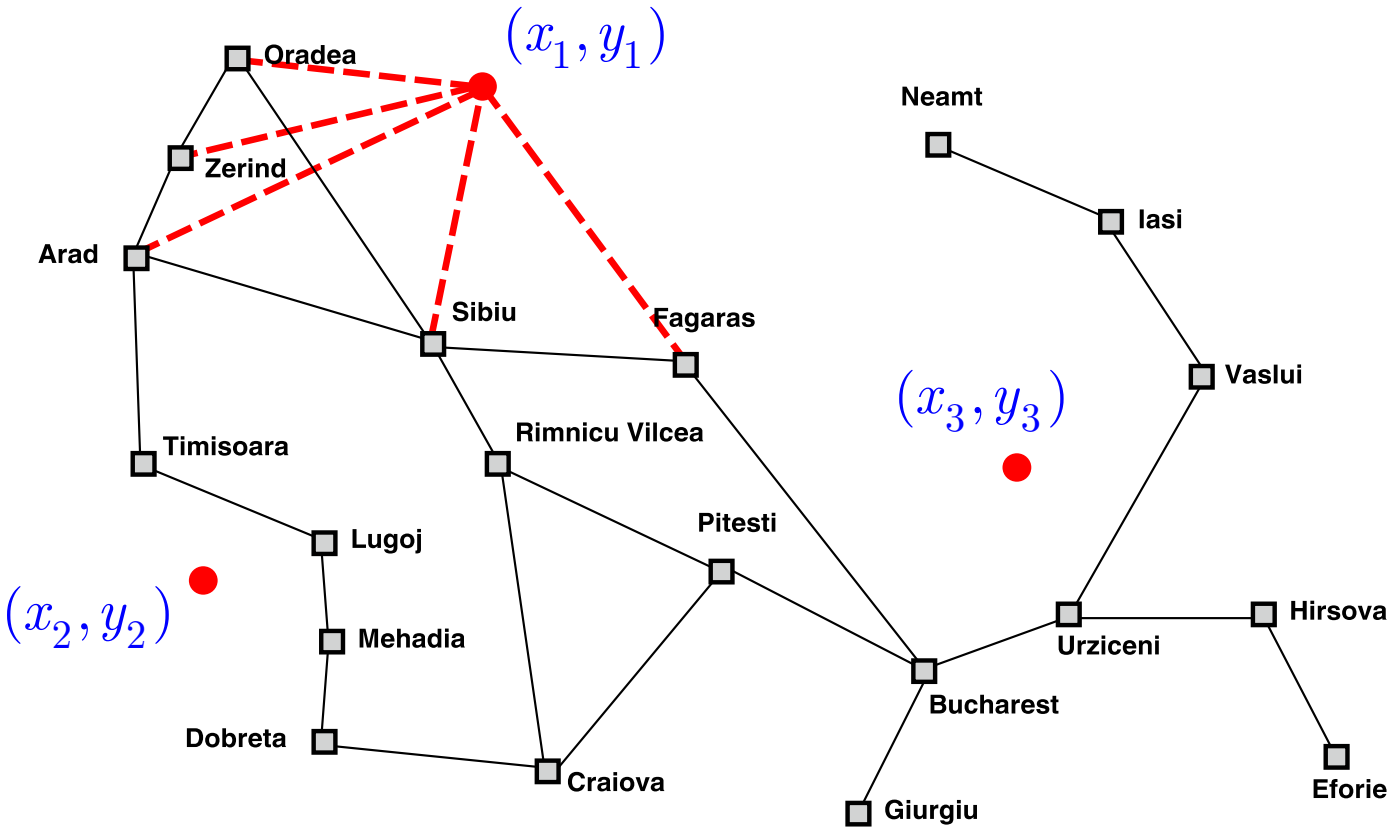
فضاهای حالت پیوسته

مثال

می‌خواهیم برای سه فرودگاه در رومانی مکان‌یابی کنیم.

$(x_1, y_1), (x_2, y_2), (x_3, y_3)$ فضای حالت ۶ بعدی

تابع هدف = مجموع مربعات فواصل از هر شهر تا نزدیک‌ترین فرودگاه $= f(x_1, y_1, x_2, y_2, x_3, y_3)$



روش‌های جستجو در فضای پیوسته

روش تپه‌نوردی (افزایش/کاهش گرادیانی)

تپه‌نوردی *Hill-climbing*	
ماکزیمم‌سازی (بیشینه‌سازی) *Maximization*	**می‌نیمم‌سازی** (کمینه‌سازی) *Minimization*
افزایش گرادیانی *Gradient Ascent*	**کاهش گرادیانی** *Gradient Descent*
تندترین فراز *Steepest Ascent*	**تندترین شیب** *Steepest Descent*
حرکت در جهت تندترین شیب/گرادیان صعودی تابع هدف	حرکت در جهت تندترین شیب/گرادیان نزولی تابع هدف

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

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

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right)$$

$$\mathbf{x} = (x_1, x_2, \ldots, x_n)$$

گرادیان برای هر تابع چندمتغیره، برداری است که جهت بیشترین تغییرات تابع در هر نقطه را نشان می‌دهد.

روش‌های جستجو در فضای پیوسته

روش نیوتن–رافسون

NEWTON–RAPHSON METHOD

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

ماتریس هسی (Hessian)

ماتریس مشتقات دوم

$$\mathbf{H}_f(\mathbf{x}) = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right]_{n\times n}$$

$$\mathbf{x} = (x_1, x_2, \ldots, x_n)$$

برای حل:

$$\nabla f(\mathbf{x}) = 0$$

گسسته‌سازی فضای حالت

DISCRETIZATION

روش‌های گسسته‌سازی
برای تبدیل فضای پیوسته به فضای گسسته

برای مثال:
گرادیان‌های تجربی:
در نظر گرفتن تغییرات در هر یک از مختصات به اندازه‌ی $\pm\delta$

امکان استفاده از الگوریتم‌های جستجوی فضای حالت گسسته پس از گسسته‌سازی

هوش مصنوعی

جستجو در محیط‌های پیچیده (۲)

۴

جستجو با کنش‌های غیرقطعی

هوش مصنوعی

جستجو در محیطهای پیچیده (۲)

۵

جستجو با مشاهدات جزئی

هوش مصنوعی

جستجو در محیط‌های پیچیده (۲)

۶

عامل‌های جستجوی برخط و محیط‌های ناشناخته

روش‌های جستجو

برون‌خط/ برخط

روش‌های جستجو	
برون‌خط *Offline*	**برخط** *Online*
راه‌حل پیش از اجرا مشخص می‌شود.	محاسبات و کنش‌ها یک درمیان انجام می‌شوند.

نخست انجام یک کنش،
سپس مشاهده‌ی محیط و
محاسبه‌ی کنش بعدی

ضروری برای محیط‌های پویا و نیمه‌پویا
و
محیط‌های ناشناخته
(مسائل اکتشافی)

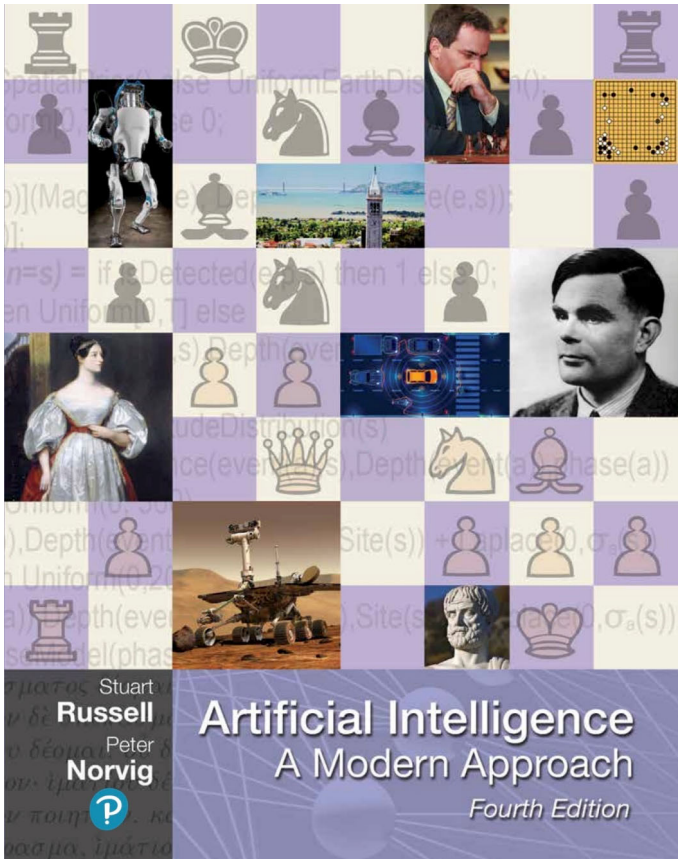
هوش مصنوعی

جستجو در محیطهای پیچیده (۲)

۷

منابع

منبع اصلی



Stuart Russell and Peter Norvig,
Artificial Intelligence: A Modern Approach,
4th Edition, Prentice Hall, 2020.

Chapter 4

CHAPTER 4

SEARCH IN COMPLEX ENVIRONMENTS

In which we relax the simplifying assumptions of the previous chapter, to get closer to the real world.

Chapter 3 addressed problems in fully observable, deterministic, static, known environments where the solution is a sequence of actions. In this chapter, we relax those constraints. We begin with the problem of finding a good state without worrying about the path to get there, covering both discrete (Section 4.1) and continuous (Section 4.2) states. Then we relax the assumptions of determinism (Section 4.3) and observability (Section 4.4). In a nondeterministic world, the agent will need a conditional plan and carry out different actions depending on what it observes—for example, stopping if the light is red and going if it is green. With partial observability, the agent will also need to keep track of the possible states it might be in. Finally, Section 4.5 guides the agent through an unknown space that it must learn as it goes, using **online search**.

4.1 Local Search and Optimization Problems

In the search problems of Chapter 3 we wanted to find paths through the search space, such as a path from Arad to Bucharest. But sometimes we care only about the final state, not the path to get there. For example, in the 8-queens problem (Figure 4.3), we care only about finding a valid final configuration of 8 queens (because if you know the configuration, it is trivial to reconstruct the steps that created it). This is also true for many important applications such as integrated-circuit design, factory floor layout, job shop scheduling, automatic programming, telecommunications network optimization, crop planning, and portfolio management.

Local search

Local search algorithms operate by searching from a start state to neighboring states, without keeping track of the paths, nor the set of states that have been reached. That means they are not systematic—they might never explore a portion of the search space where a solution actually resides. However, they have two key advantages: (1) they use very little memory; and (2) they can often find reasonable solutions in large or infinite state spaces for which systematic algorithms are unsuitable.

Optimization problem
Objective function

Local search algorithms can also solve **optimization problems**, in which the aim is to find the best state according to an **objective function**.

State-space landscape

To understand local search, consider the states of a problem laid out in a **state-space landscape**, as shown in Figure 4.1. Each point (state) in the landscape has an "elevation," defined by the value of the objective function. If elevation corresponds to an objective function,