

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



هوش مصنوعی

فصل ۴

جستجو در محیط‌های پیچیده

Search in Complex Environments

کاظم فولادی قلعه
دانشکده مهندسی، پردیس 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، پردیس فارابی
دانشگاه تهران

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

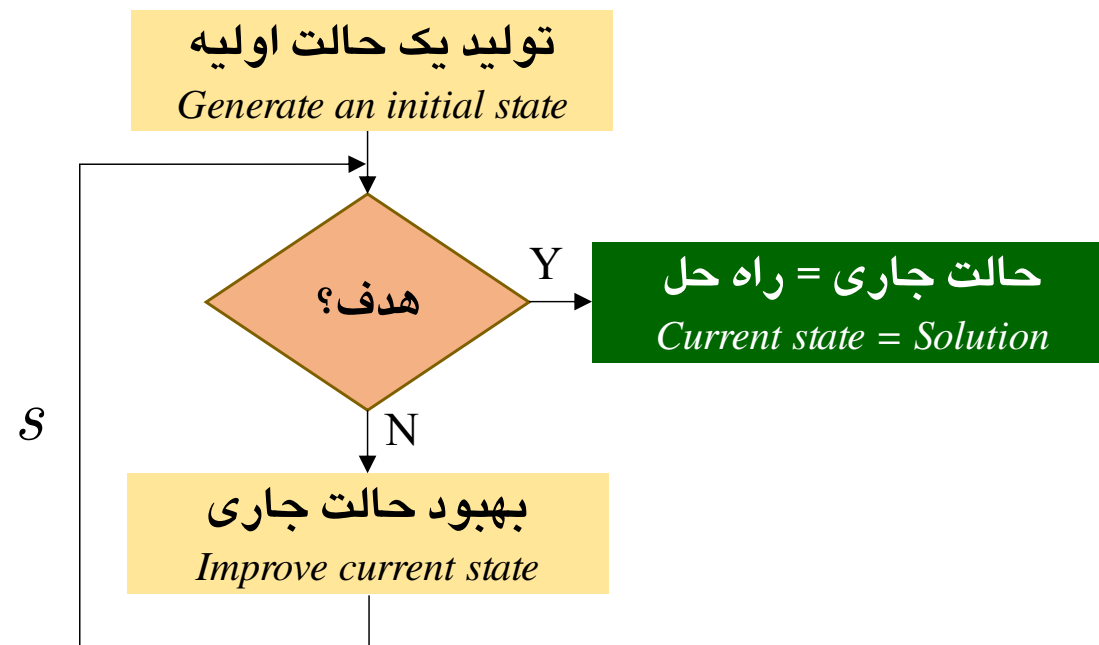
الگوریتم‌های بهبود تکراری (جستجوی محلی)

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



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

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

یافتن راه‌حل‌های مناسب در
فضاهای حالت **بزرگ** یا **نامتناهی** (پیوسته)

حافظه‌ی مصرفی پایین
(معمولاً مقدار **ثابت**)



مناسب برای حل **مسائل بهینه‌سازی خالص**
یافتن **بهترین حالت** با توجه به یک تابع هدف (objective function)

مسئله‌ی بهینه‌سازی

بهینه‌سازی
Optimization

می‌نیم‌سازی (کمینه‌سازی)
Minimization

ماکزیم‌سازی (بیشینه‌سازی)
Maximization

$$\min f(x)$$

تابع هدف
Objective function

$$\max f(x)$$

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

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

$$\min f(x) \Leftrightarrow \max -f(x)$$

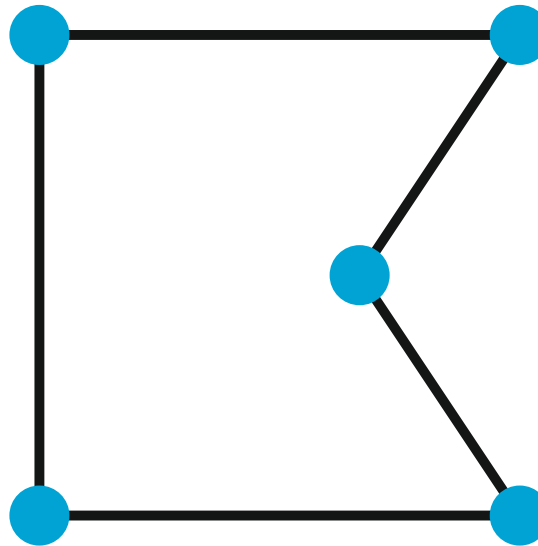
مثال: مسئله‌ی فروشنده‌ی دوره‌گرد

تعریف مسئله

TRAVELLING SALESPERSON PROBLEM (TSP)

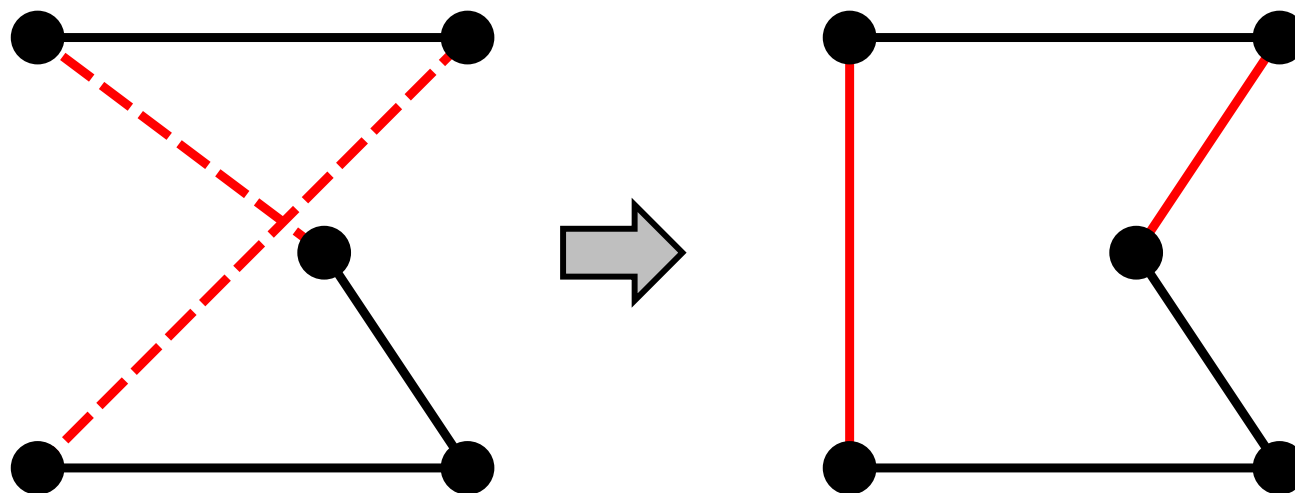
یافتن توری که هر شهر را دقیقاً یک مرتبه بپیماید.

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



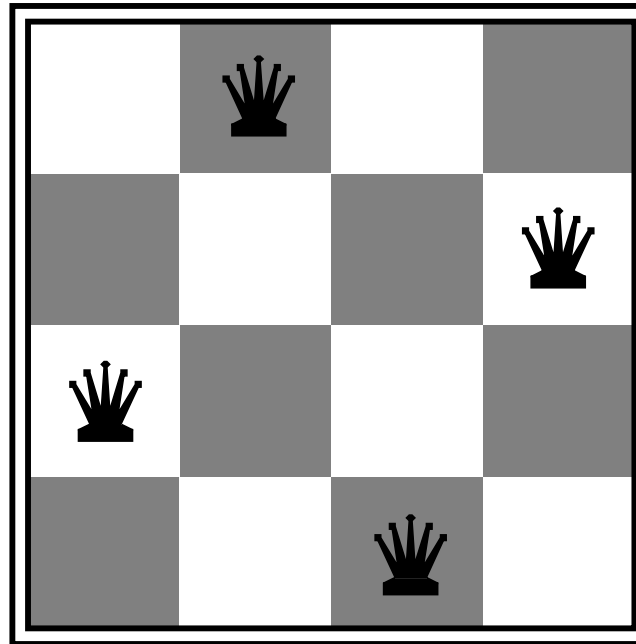
مثال: مسئله n -وزیر

تعریف مسئله

n -QUEENS

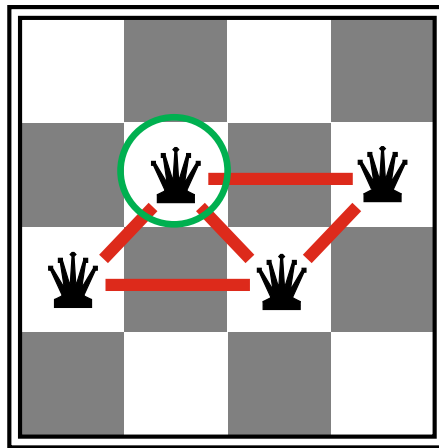
قرار دادن n مهره‌ی وزیر روی صفحه شطرنج $n \times n$ بدون اینکه یکدیگر را تهدید کنند.

نباید هم‌سطر، هم‌ستون یا هم‌قطر باشند.

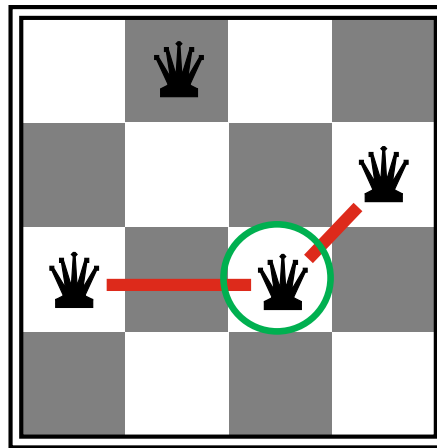
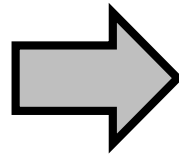


مثال: راه حل مسئله n -وزیر با جستجوی محلی

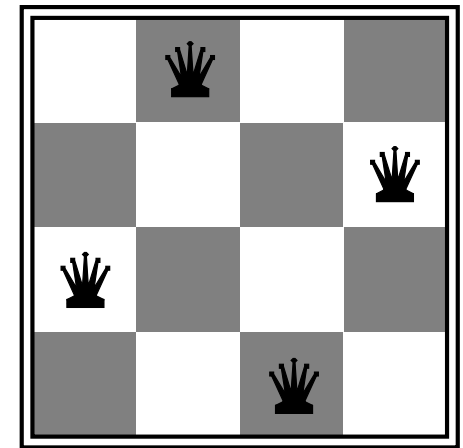
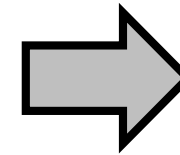
شروع با یک ترتیب انتساب وزیرها (مثلاً هر وزیر در یک ستون)
 جابجا کردن یک وزیر برای کاهش تعداد تداخلها
 تا رسیدن به حالت بدون تداخل



$h = 5$



$h = 2$



$h = 0$

با این روش، مسئله n -وزیر برای n بسیار بزرگ (در حدود یک میلیون) تقریباً به طور آنی حل می شود.

۲

تپه‌نوردی

تپه‌نوردی (افزایش / کاهش گرادیانی)

پایه‌ای‌ترین تکنیک جستجوی محلی

HILL-CLIMBING (GRADIENT ASCENT/DESCENT)

مشابه بالا رفتن از اورست در مه غلیظ با داشتن فراموشی

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*

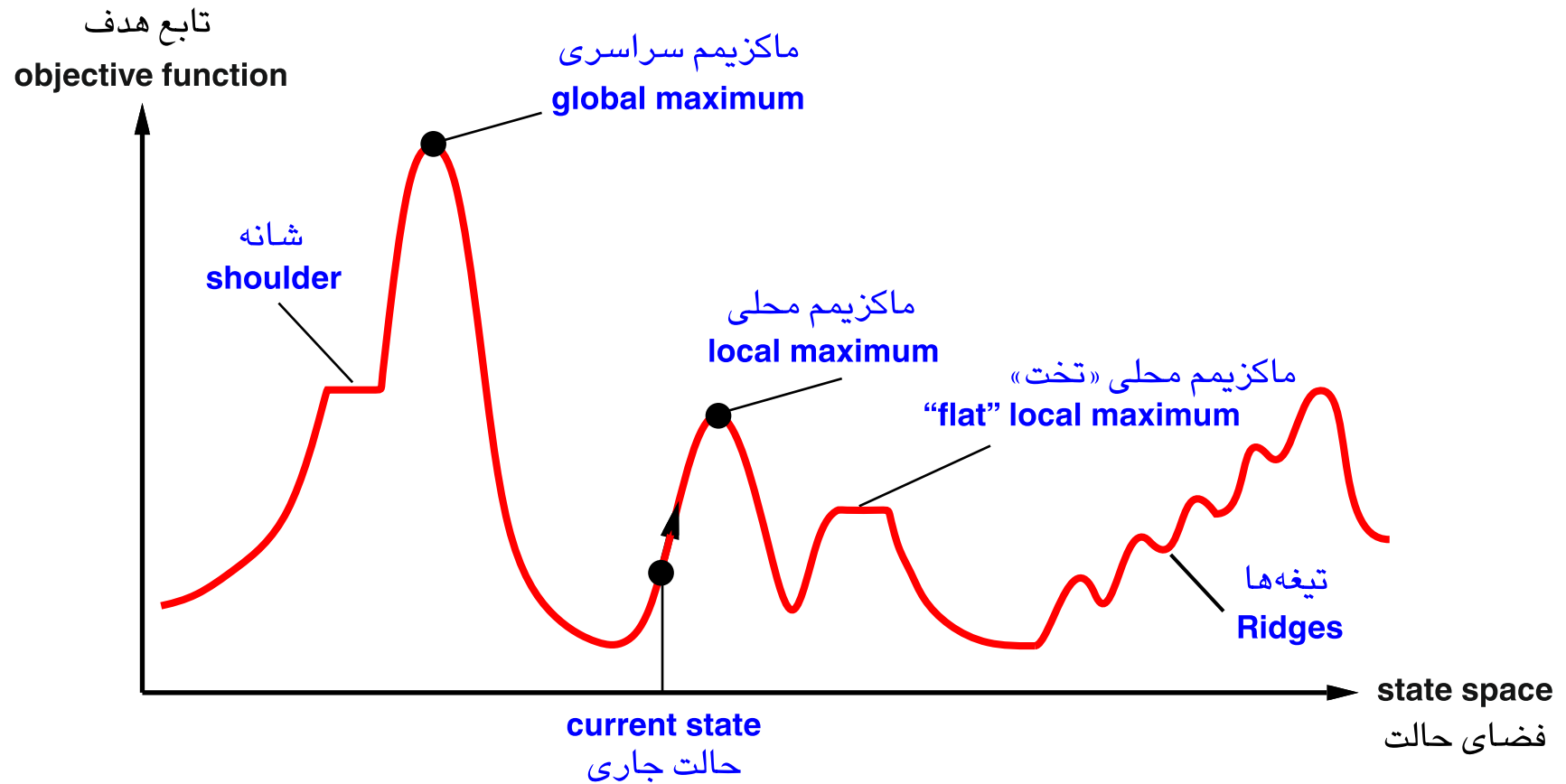
قاعده‌ی تپه‌نوردی: حرکت در مسیر افزایش تابع ارزیابی تا جای ممکن

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

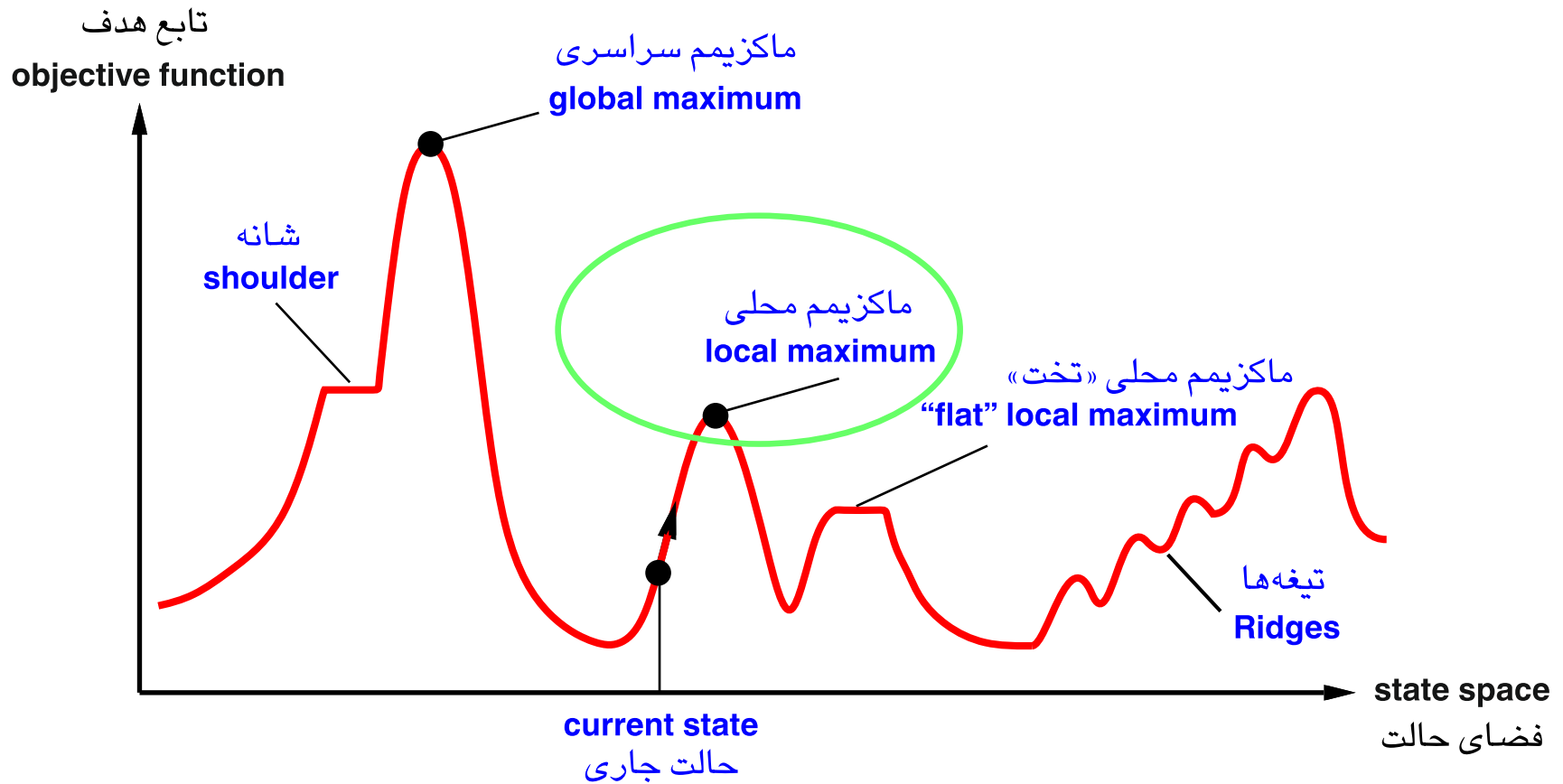
نمونه‌ی چشم‌انداز فضای حالت



تپه‌نوردی

مشکل: گیرافتادن در ماکزیمم محلی

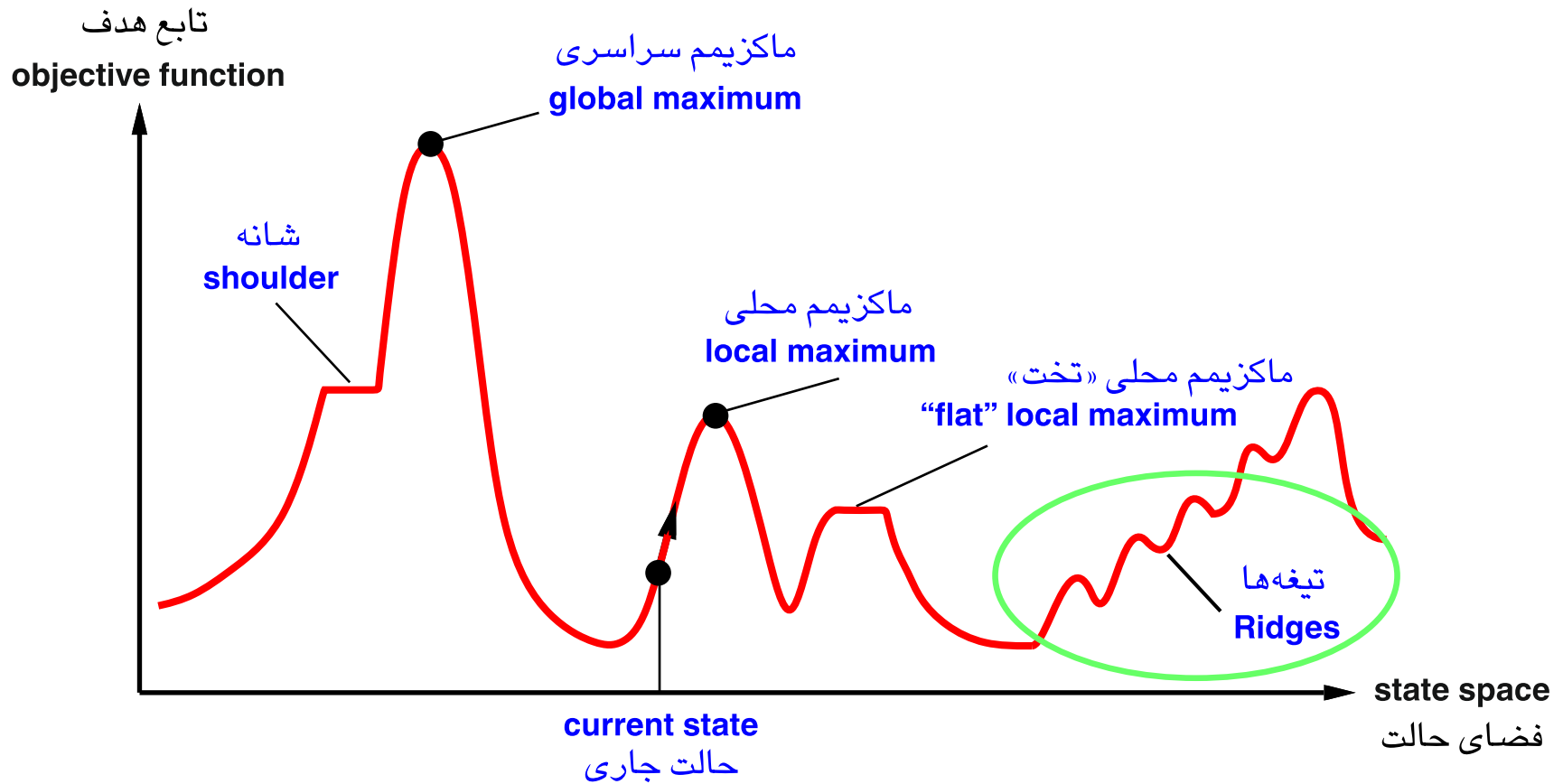
ماکزیمم محلی، قله‌ای است که بلندتر از هر یک از همسایگانش است، اما کوتاه‌تر از ماکزیمم سراسری است.



تپه‌نوردی

مشکل: گیرافتادن در تیغه‌ها

تیغه‌ها، حاصل یک دنباله از ماکزیمم‌های محلی هستند که عبور از آن برای الگوریتم‌های حریصانه بسیار دشوار است.

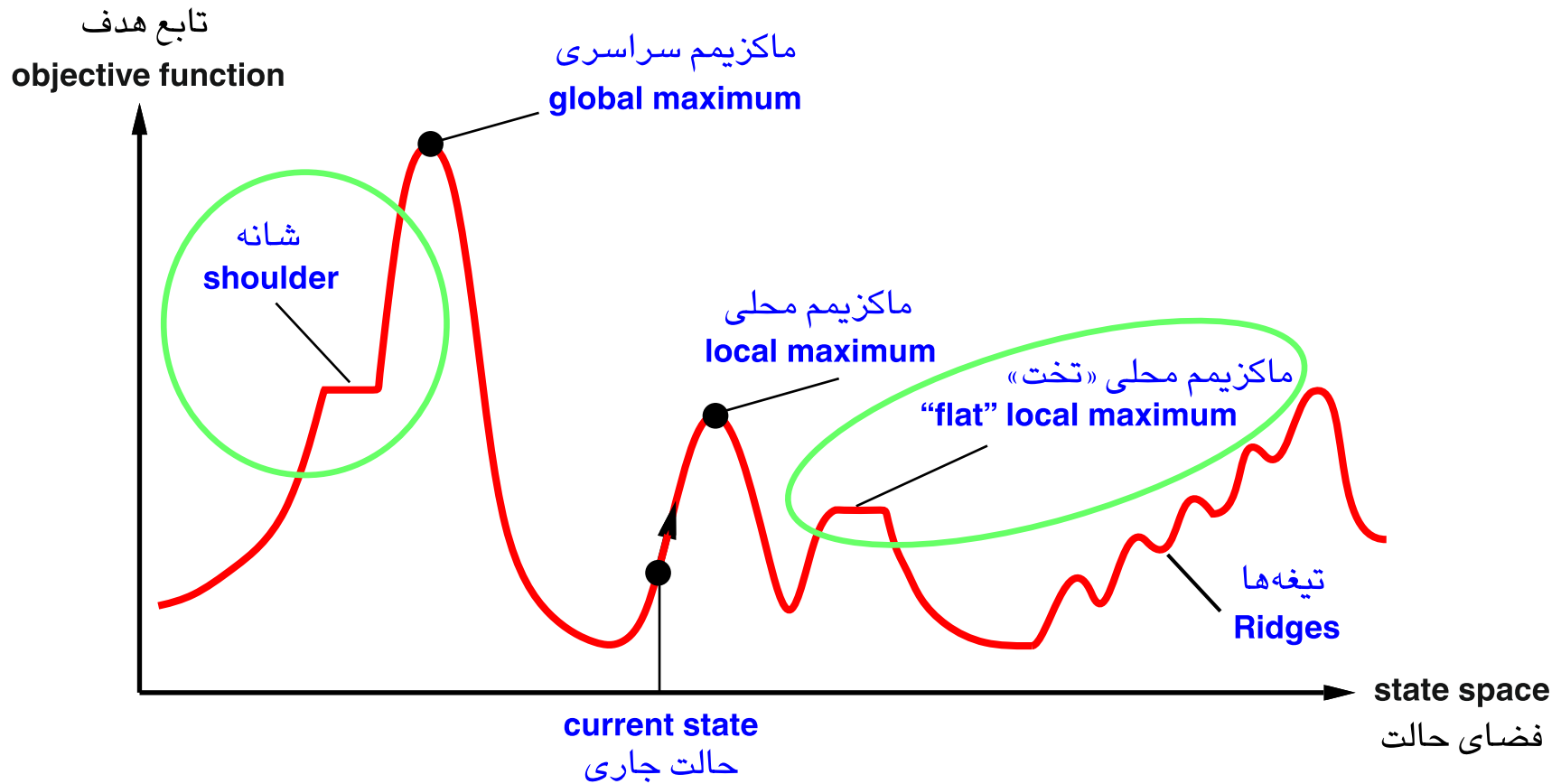


تپه‌نوردی

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 رفت).



تپه‌نوردی

مشکلات

گیر افتادن در ماکزیمم محلی



گیر افتادن در تیغه‌ها



گیر افتادن در فلات‌ها



تپه‌نوردی

مشکل تیغه‌ها در توابع چندمتغیره

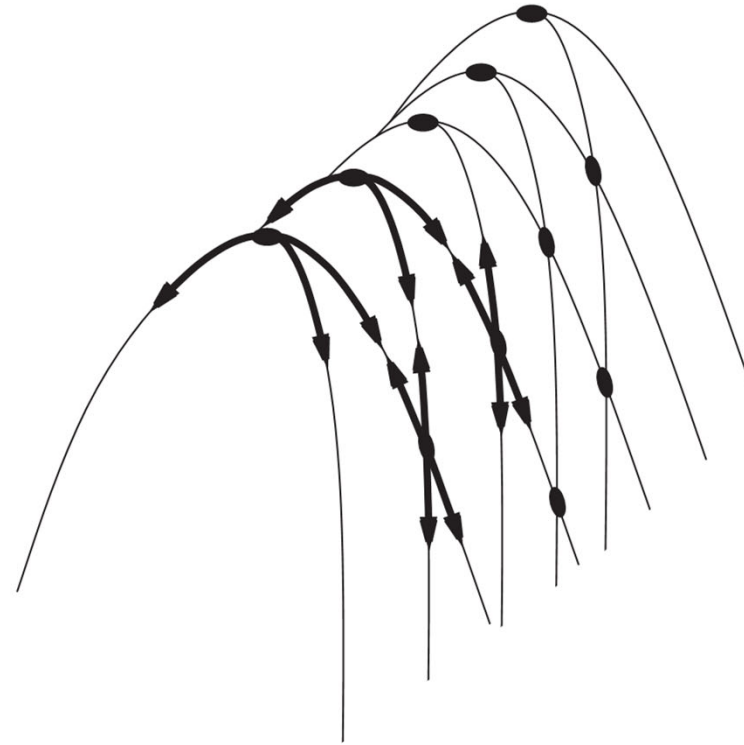


Figure 4.4 Illustration of why ridges cause difficulties for hill climbing. The grid of states (dark circles) is superimposed on a ridge rising from left to right, creating a sequence of local maxima that are not directly connected to each other. From each local maximum, all the available actions point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

گزینه‌هایی برای حل مشکلات تپه‌نوردی

شکل‌های دیگر تپه‌نوردی

انتخاب تصادفی از میان گزینه‌های حرکت به سمت بالا

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

تپه‌نوردی اتفاقی

Stochastic hill-climbing

نوعی پیاده‌سازی تپه‌نوردی اتفاقی:

مابدهای یک حالت به صورت تصادفی یکی پس دیگری، آن قدر تولید می‌شوند
تا حالتی تولید شود که بهتر از حالت جاری باشد.

مناسب برای زمانی که یک حالت تعداد زیادی مابعد دارد.

تپه‌نوردی نخستین-گزینه

First-choice 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 با شروع مجدد تصادفی

Random-restart hill-climbing

تپه‌نوردی

افزایش / کاهش گرادیانی

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

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

۳

تافت
شبیه‌سازی
شده

تافت شبیه‌سازی شده

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 $\Leftarrow$ به ماده امکان داده می‌شود تا در یک حالت کریستالی با انرژی پایین شکل بگیرد.

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

نمونه‌ای از الگوریتم‌های الهام گرفته شده از طبیعت
Nature-inspired

تافت شبیه سازی شده

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 می شود.
⇐ به مواد بستنی امکان داده می شود تا در یک حالت انرژی پایین ببنند.



تافت شبیه سازی شده

مثال

SIMULATED ANNEALING

توپ در یک ظرف موج دار:

می خواهیم یک توپ را در یک ظرف پر از شیار و پستی-بلندی به عمیق ترین نقطه برسانیم.

تکان دادن ظرف (گرما دادن به آن)

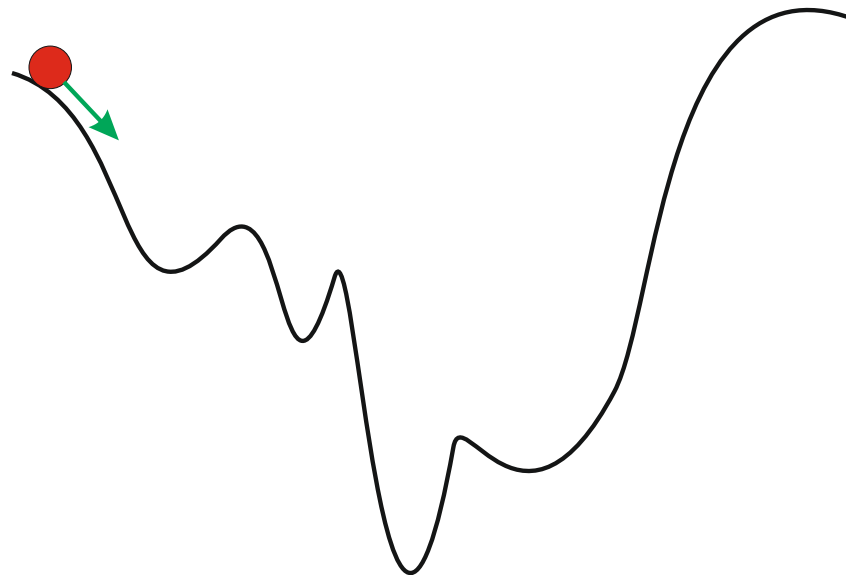
اجازه داده به توپ برای حرکت (خنک کردن تدریجی)

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 می شود.

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

تکان دادن ظرف: ابتدا زیاد و به مرور تعداد و اندازه ی تکان ها کم می شود.



تافت شبیه‌سازی شده

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 $\leftarrow$ 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 $e^{\Delta E/T}$

تافت شبیه‌سازی شده

«دما» به عنوان متغیر کنترلی

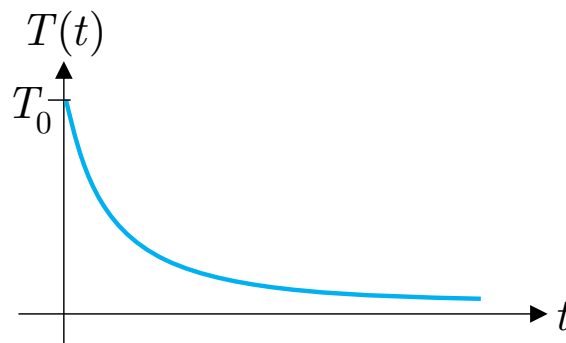
با متغیر دما میزان اجازه دادن به حرکت‌های بد کنترل می‌شود:

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 شبیه‌سازی شده

انواع گوناگون

تافت شبیه‌سازی شده استاندارد
(تافت بولتزمن)

Standard/Boltzmann Annealing (SA)

$$p(x) = \alpha e^{E(x)/kT}$$

استفاده از توزیع احتمال بولتزمن

- 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 شبیه‌سازی شده

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

استفاده از زمان‌بندی دمای به طور نمایی سریع‌تر از SA استاندارد

تافت سریع

Fast Annealing (FA)

$$T_n = T_0 / n$$

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

بازتافت شبیه‌سازی شده‌ی 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 اطفای + بازتافت

☺ تضمین همگرایی احتمالاتی به راه‌حل بهینه

تافت شبیه‌سازی شده وفقی

Adaptive Simulated Annealing (ASA)



هوش مصنوعی

فصل ۴

جستجو در محیط‌های پیچیده (۲)

Search in Complex Environments (2)

کاظم فولادی قلعه
دانشکده مهندسی، پردیس فارابی
دانشگاه تهران

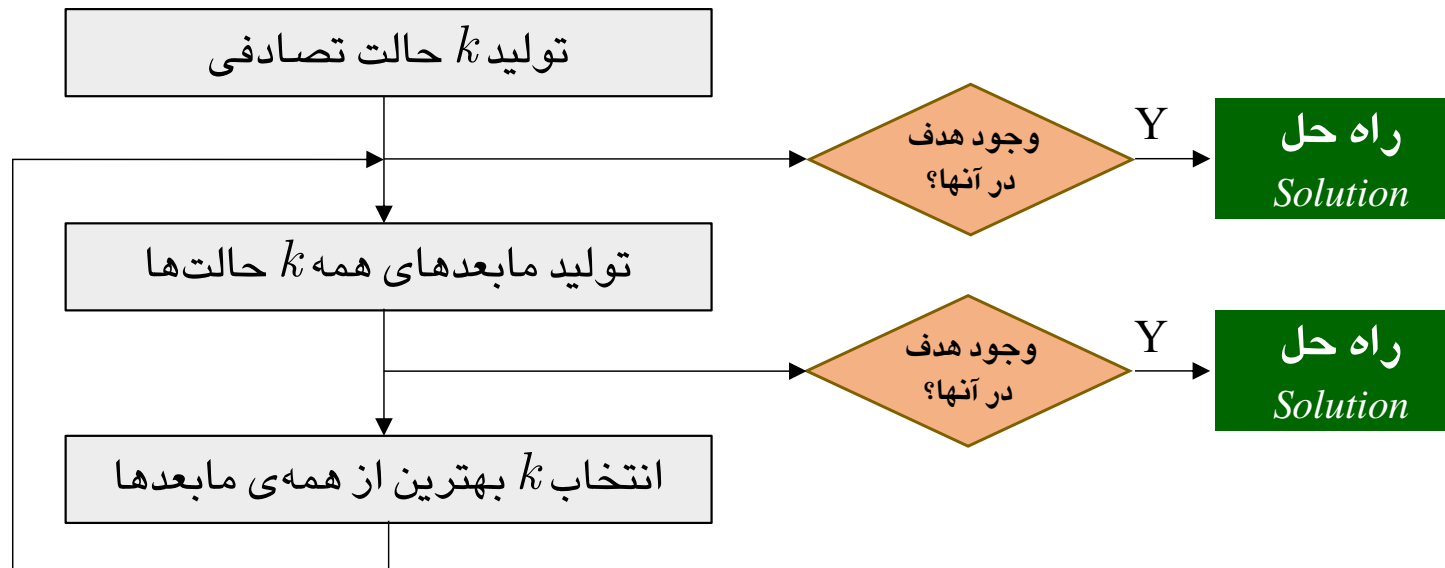
۱

جستجوی پرتوی محلی

جستجوی پرتوی محلی

LOCAL BEAM-SEARCH

ایده: نگهداری k حالت به جای ۱ حالت + انتخاب k مابعد بهتر هر حالت



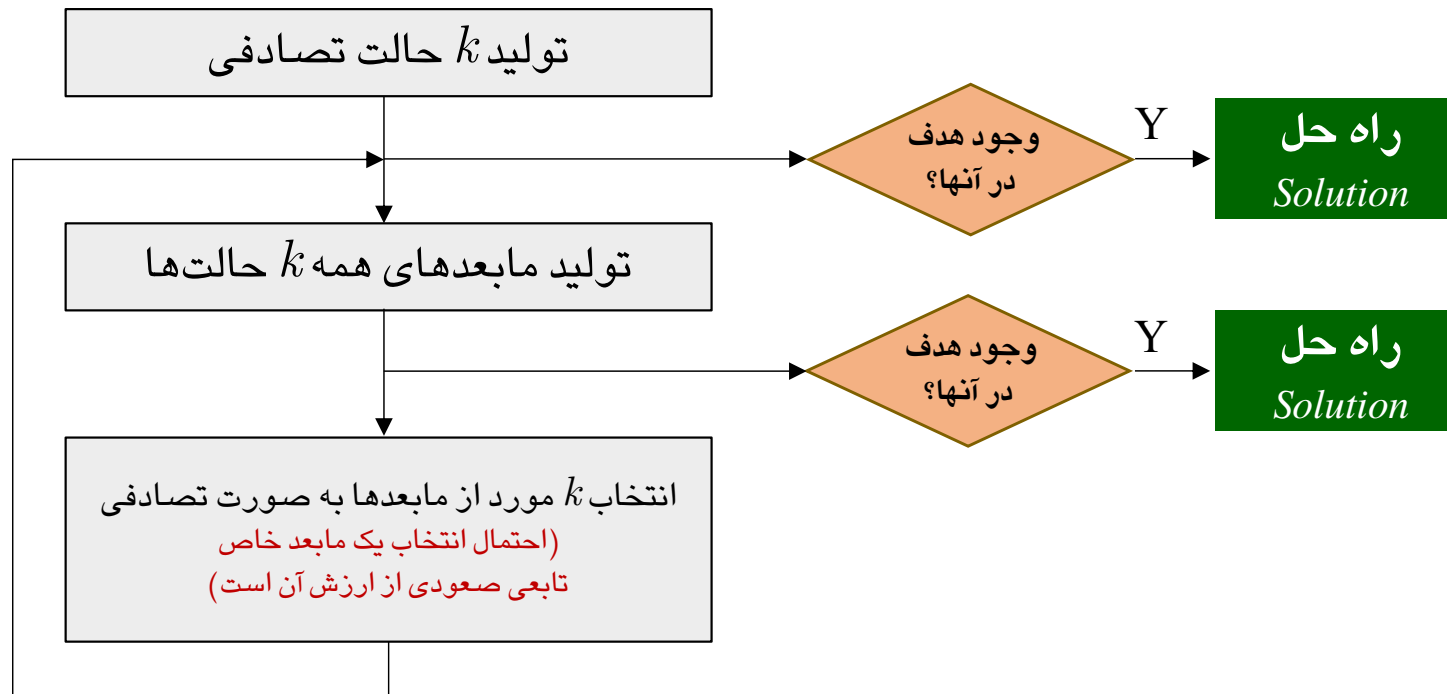
این الگوریتم با اجرای k جستجوی موازی با شروع تصادفی متفاوت است:
این جستجوها مستقل از هم نیستند و اطلاعات سودمند بین آنها رد و بدل می شود:
جستجوهای که حالت های خوبی پیدا می کنند، دیگر جستجوها را تشویق می کنند تا به آنها بپیوندند.

مشکل: اغلب همه ی k حالت به یک تپه ی محلی منجر می شوند! نبود تنوع در میان حالتها

جستجوی پرتوی محلی اتفاقی

STOCHASTIC LOCAL BEAM-SEARCH

ایده: نگهداری k حالت به جای 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 برآزش (fitness)

۲

الگوریتم‌های تطوری (تکاملی)

الگوریتم‌های تطوری (تکاملی)

EVOLUTIONARY ALGORITHMS

الگوریتم‌های تطوری *Evolutionary Algorithms*

به عنوان شکل‌های تغییر یافته از جستجوی پرتوی اتفاقی
که به صورت 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 **باز ترکیب** نامیده می‌شود).

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

الگوریتم‌های تطوری

EVOLUTIONARY ALGORITHMS

الگوریتم‌های تطوری *Evolutionary Algorithms*

هر فرد یک رشته روی یک الفبای متناهی است (معمولاً یک رشته‌ی بیتی)
[دقیقاً مانند DNA که رشته‌ای روی الفبای ACGT است]

الگوریتم‌های ژنتیک
Genetic Algorithm

هر فرد یک دنباله از اعداد حقیقی است.

استراتژی‌های تطوری
Evolution Strategies

هر فرد یک برنامه‌ی کامپیوتری است.

برنامه‌نویسی ژنتیک
Genetic Programming

انواع
استراتژی‌های
تطوری
بر اساس
بازنمایی
فرد

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

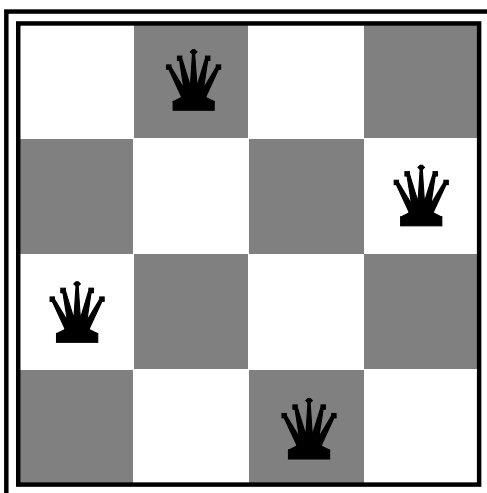
کروموزوم (ژنوم)

یک ساختمان داده برای کدگذاری راه‌حل‌های مسئله (حالت‌ها)
برای ذخیره‌سازی در کامپیوتر (معمولاً رشته / بردار)

کروموزوم
Chromosome

ژنوم
Genome

فرد
Individual



(3, 1, 4, 2)

مثال

الگوریتم‌های ژنتیک

جمعیت

مجموعه‌ای از افراد (کروموزوم‌ها)

جمعیت

Population

الگوریتم‌های ژنتیک

انتخاب

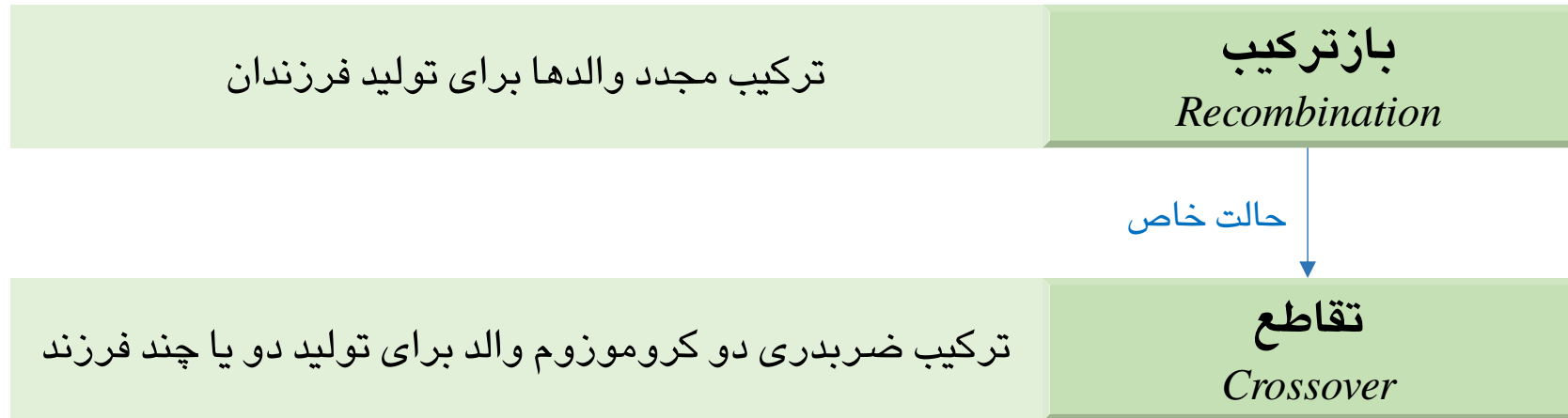
انتخاب افراد (کروموزوم‌ها) برای جفت شدن

انتخاب
Selection

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

الگوریتم‌های ژنتیک

باز ترکیب (مانند تقاطع)



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

الگوریتم‌های ژنتیک

جهش

تغییری تصادفی در یک کروموزوم

جهش

Mutation

- 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

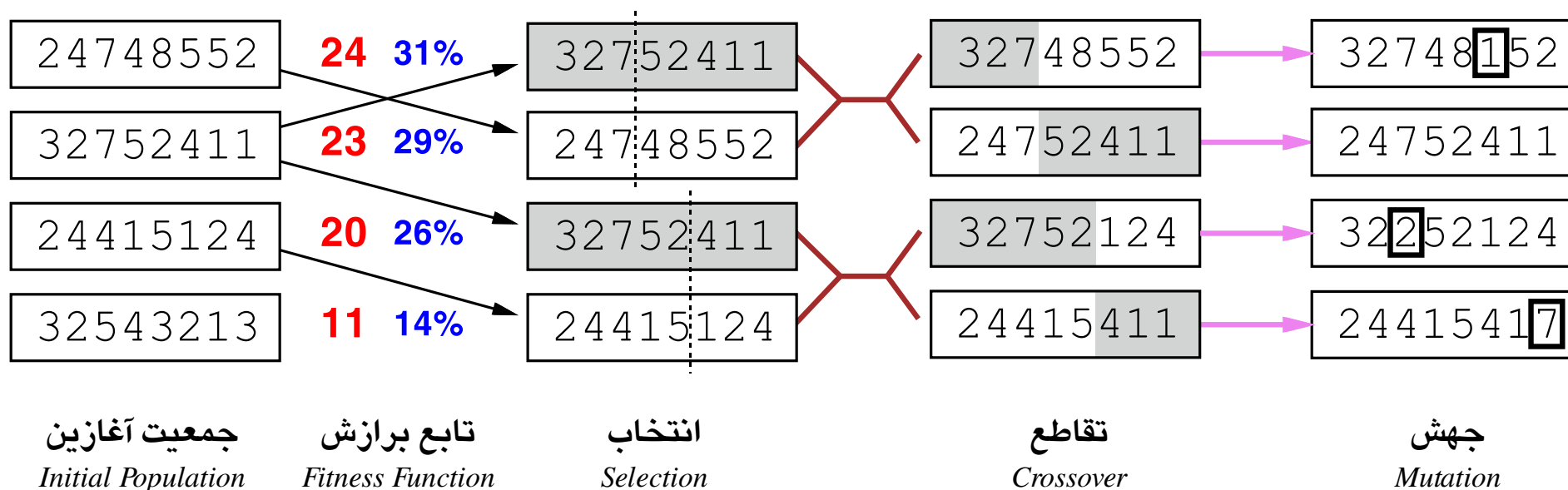
Objective Function

هدف، ماکزیمم‌سازی تابع برازش است.

الگوریتم‌های ژنتیک

مثال

الگوریتم ژنتیک = جستجوی پرتوی محلی اتفاقی + تولید مابعدا 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 $\leftarrow$ 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

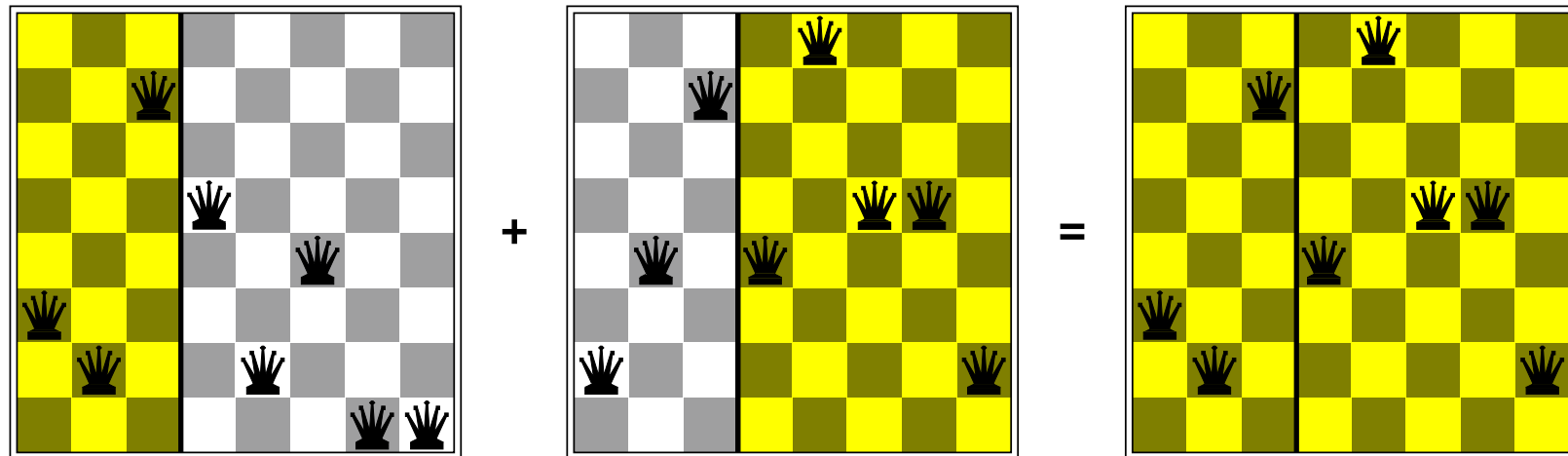
$n \leftarrow$ LENGTH(x); $c \leftarrow$ random number from 1 to n

return APPEND(SUBSTRING($x, 1, c$), SUBSTRING($y, c + 1, n$))

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

۳

جستجوی محلی در فضاهای پیوسته

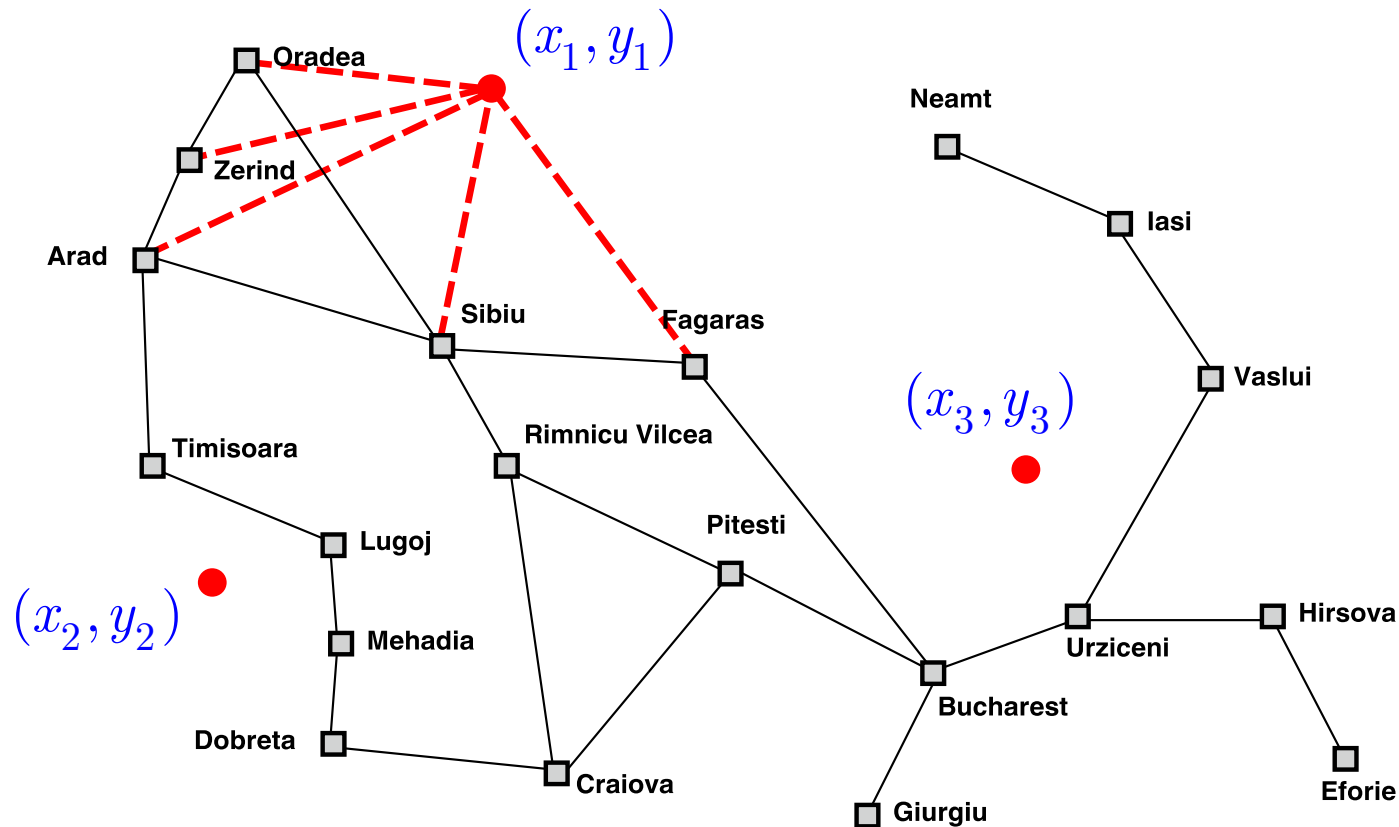
فضاهای حالت پیوسته

مثال

می‌خواهیم برای سه فرودگاه در رومانی مکان‌یابی کنیم.

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

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

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

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

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

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

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

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

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

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

۴

جستجو با کنش‌های غیر قطعی

۵

جستجو با مشاهدات جزئی

۶

عوامل‌های
جستجوی
برخط و
محیط‌های
ناشناخته

روش‌های جستجو

برون خط / برخط

روش‌های جستجو	
برخط <i>Online</i>	برون خط <i>Offline</i>
محاسبات و کنش‌ها یک در میان انجام می‌شوند.	راه حل پیش از اجرا مشخص می‌شود.

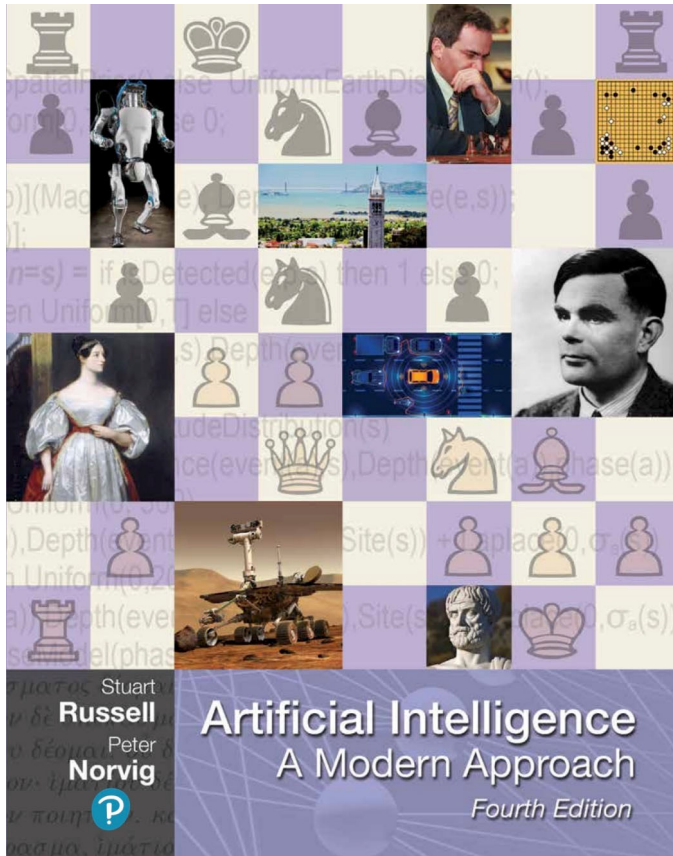
نخست انجام یک کنش،
سپس مشاهده‌ی محیط و
محاسبه‌ی کنش بعدی

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



منابع

منبع اصلی



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

Chapter 4