مدرس: دکتر فاطمه منصوری

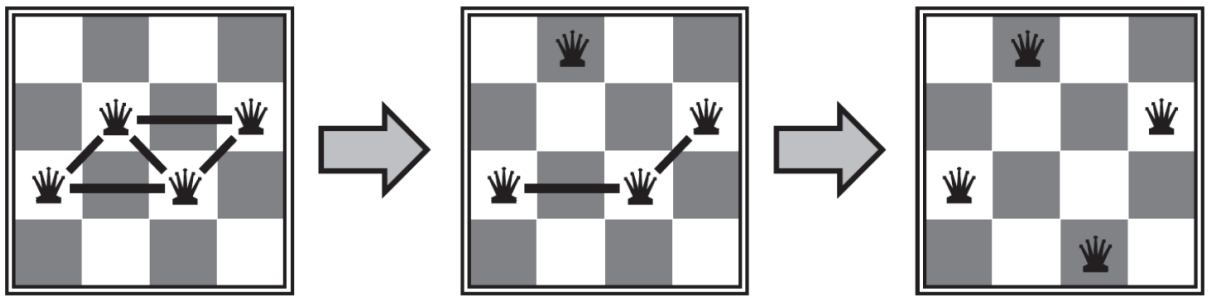
هوش مصنوعی

تهیه کننده : فاطمه حسین زاده

These lecture notes are heavily based on notes written for the Artificial Intelligence Course at Berkley University

جستجوی محلی[5]

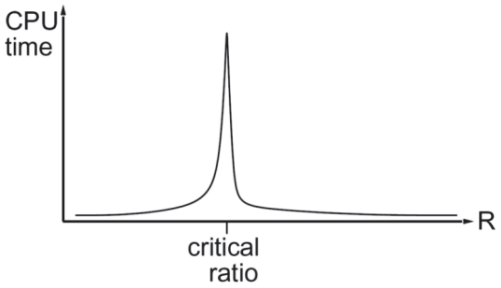
الگوریتم جستجوی عقب‌گرد تنها الگوریتمی نیست که برای *CSP* وجود دارد. یکی دیگر از الگوریتم‌های پرکاربرد، الگوریتم جستجوی محلی است که ایده آن بسیار ساده اما به‌طور قابل توجهی مفید است. جستجوی محلی با تکرار بهبود پیدا می‌کند. ابتدا مقادیر را تصادفی انتخاب می‌کنیم، سپس یکی از متغیرها را که با بقیه تعارض دارد به‌طور تصادفی انتخاب می‌کنیم و مقدارش را به متغیری با کمترین تعداد محدودیت اختصاص می‌دهیم. این کار را تا زمانی تکرار می‌کنیم که دیگر محدودیتی نقض نشود.(سیاستی که به عنوان کمترین تضادها شناخته می‌شود.) با استفاده از این سیاست، مسئله‌های *CSP* مانند مسئله *n*–وزیر در زمانِ و فضای کارآمد حل می‌شوند. در مثال زیر با ۴ وزیر، تنها پس از ۲ تکرار به یک راه حل می رسیم:



در واقع، در جستجوی محلی به نظر می‌رسد، نه تنها برای مسئلهN –وزیر با N های خیلی بزرگ، بلکه برای هر مسئله CSP تصادفی تولید شده‌ای، تقریبا زمان اجرا ثابت و احتمال موفقیت بالا است. علی‌رغم این مزایا، جستجوی محلی نه کامل و نه بهینه است و بنابراین لزوماً به یک راه حل بهینه همگرا نمی شود. علاوه بر این، یک نسبت بحرانی وجود دارد که استفاده از جستجوی محلی را بسیار گران می‌کند:

[5] Local Search

$$R = \frac{\text{number of constraints}}{\text{number of variables}}$$



شکل بالا نمودار یک بعدی یک تابع هدف در فضای حالت را نشان می دهد. برای این تابع می خواهیم حالتی با بالاترین مقدار هدف پیدا کنیم. ایده اصلی الگوریتم‌های جستجوی محلی این است که از هر حالت به حالتی در حوالی خود که دارای ارزش هدف بالاتری هستند حرکت می‌کند تا زمانی که این ارزش به حداکثر (در حالت بهینه به بیشترین ارزش) برسد.

[در جستجوی محلی ما به دنبال کاوش نظام‌مند فضای جستجو در الگوریتم های کلاسیک هستیم. یک درخت جستجو را در نظر بگیرید؛ جستجوی محلی

۱. مسیرهای مختلف جستجو را به صورت همزمان بررسی می‌کند.
۲. گزینه‌های دیگر کاوش را در هنگام بررسی یک مسیر نگهداری می‌کند.

در جستجوی محلی ما به دنبال یک مسیر برای رسیدن به هدف هستیم؛ این در حالی‌است که در بسیاری از مسائل مسیررسیدن به هدف مهم نیست.

جستجوی محلی حالت فعلی را نگهداری می‌کند و نیازی به نگهداری حالت‌های کاوش شده نیست. حالت بعدی هم از بین حالت‌های مجاور حالت فعلی انتخاب می‌شود. یکی دیگر از مزایای این نوع جستجو نیاز به حافظه بسیار کم و در بیشتر موارد حافظه‌ای ثابت است اما همان‌طور که گفته شد لزوما بهترین جواب را پیدا نمی‌کند. در این مسئله دو حالت بهینه محلی(Local) و سراسری یا جهانی (Global) وجود دارد.]

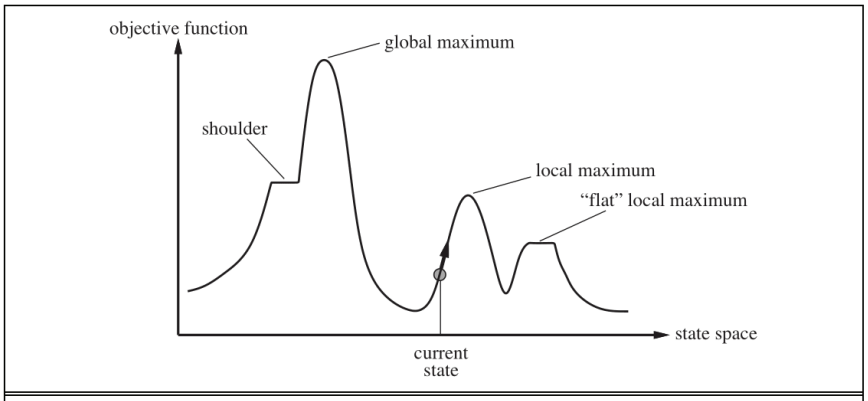


Figure 4.1 A one-dimensional state-space landscape in which elevation corresponds to the objective function. The aim is to find the global maximum. Hill-climbing search modifies the current state to try to improve it, as shown by the arrow. The various topographic features are defined in the text.

ما سه الگوریتم تپه نوردی، تبرید شبیه سازی شده و ژنتیک را در این جزوه پوشش خواهیم داد. همه این الگوریتم‌ها در بهینه‌سازی مسائل برای به حداکثر رساندن یا به حداقل رساندن یک تابع هدف استفاده می‌شوند.

جستجوی تپه نوردی

الگوریتم جستجوی تپه‌نوردی از حالت فعلی به سمت یکی از حالات همسایه خود حرکت می‌کند که مقدار هدف را افزایش می‌دهد. الگوریتم، درخت جستجو را حفظ نمی کند؛ بلکه فقط حالت ها و مقادیر متناظر هدف را حفظ می کند. در حالت حریصانه، این الگوریتم ممکن است در ماکزیمم محلی قرار بگیرد و آن را به عنوان مقدار هدف در نظر می‌گیرد. در این صورت الگوریتم آسیب‌پذیر است.(عکس بالا را مشاهده کنید). در این الگوریتم ممکن است به حالتی برخورد کنید که هم حالت فعلی و هم همه حالت‌های همسایه از یک مقدار برخوردار هستند و جلورَوی باعث هیچ‌گونه پیشرفتی نمی‌شود. در این حالت ما به یک حالت ماکزیمم مسطح محلی[6] برخورد کرده‌ایم. حالت دیگری که ممکن است به وجود بیاید این است که حالت‌های همسایه با حالت فعلی دارای یک مقدار باشند اما پیشرفت در ادامه راه میسر باشد و بعد از گذراندن حالات یکسان به پیشرفت برسیم یا به اصطلاح پیشرفت کند باشد، به این حالن اصطلاحا شانه[7] گفته می‌شود. انواع تپه‌نوردی، مانند تپه‌نوردی تصادفی که یک عمل را به‌طور تصادفی از بین حرکات ممکن انتخاب می‌کند، وجود دارد. این نمونه از تپه نوردی که در عمل هم استفاده شده است، نشان می‌دهد که به قیمت تکرارهای بیشتر به حداکثرهای بالاتر همگرا می شود.

[6] flat local maxima
[7] shoulder

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

شبه‌کد الگوریتم تپه‌نوردی در بالا قابل مشاهده است. همان‌طور که از نام آن پیداست، الگوریتم به طور مکرر به حالتی با ارزش هدف بالاتر حرکت می کند تا زمانی که چنین پیشرفتی امکان پذیر نباشد. تپه‌نوردی الگوریتمی کامل نیست. از سوی دیگر، تپه‌نوردی با شروع مجدد تصادفی، که با بکارگیری یک دنباله از تپه‌نوردی‌ها با شروع از حالت اولیه به‌طور تصادفی انجام می‌شود، کامل است زیرا در نهایت، حالت اولیه-ای که به‌طور تصادفی انتخاب شده است با مقدار حداکثر مورد نیاز منطبق خواهد شد.

]الگوریتم تپه‌نوردی هربار حالت‌های مجاور حالت فعلی را بررسی می‌کند و بهترین آنها را انتخاب می‌کند و با رسیدن به یک قله(دره) متوقف می‌شود. بسته به نقطه شروع جستجو الگوریتم می‌تواند به نقطه ماکزیمم سراسری یا ماکزیمم محلی برسد.

- گونه‌های دیگر الگوریتم
  - تپه‌نوردی تصادفی (stochastic)
    - انتخاب تصادفی از بین مجاورین بهتر
    - احتمال حالت بعدی بسته به تندی شیب رسیدن به آن دارد
  - تپه‌نوردی اولین گزینه (first-choice)
    - تولید تعدادی حالت مجاور بصورت تصادفی تا یافتن یک مجاور بهتر
    - مناسب برای مسائلی که هر حالت دارای تعداد بسیار زیادی (نامحدود) حالت مجاور است
  - تپه‌نوردی با شروع مجدد تصادفی (random-restart)
    - بکارگیری یک دنباله از تپه‌نوردی‌ها با شروع از حالت‌های اولیه‌ی تصادفی
    - بهترین حالت یافته شده در میان تمام تپه‌نوردی‌ها جواب است

- ویژگی‌های الگوریتم
  - تقریباً هیچ گونه‌ای از الگوریتم کامل نیست
    - در بهینه‌های محلی گیر می‌کنند
  - احتمال کامل بودن تپه‌نوردی با شروع مجدد تصادفی با افزایش تکرارها به یک میل می‌کند
    - نهایتاً یکی از حالت‌های شروع تصادفی در حوزه حالت هدف خواهد بود
    - اگر p احتمال موفقیت هر تپه‌نوردی باشد، تعداد شروع‌های مجدد لازم $1/p$ خواهد بود
    - هزینه جستجو بر حسب تعداد گام‌های مورد نیاز برای یافتن هدف:

$$N_s + \frac{1-p}{p} N_f$$

متوسط گام‌ها در تکرارهای موفق — متوسط گام‌ها در تکرارهای ناموفق

[

تبرید شبیه‌سازی‌شده

دومین الگوریتم جستجوی محلی که به آن خواهیم پرداخت، تبرید شبیه سازی شده است. هدف تبرید شبیه‌سازی‌شده ترکیب گام‌برداری تصادفی (گام‌برداری تصادفی به حالت‌های مجاور) و تپه نوردی برای به دست آوردن یک الگوریتم جستجوی کامل و کارآمد است. در تبرید شبیه‌سازی‌شده، اجازه حرکت به حالت‌هایی داده می‌شود که هدف را کاهش می‌دهند. به طور خاص، الگوریتم در هر حالت یک حرکت تصادفی را انتخاب می کند. اگر حرکت به هدف بالاتر منتهی شود، همیشه پذیرفته می شود. اگر از طرف دیگر به اهداف کوچکتری منتهی شود، حرکت با احتمال کمی پذیرفته می شود. این احتمال توسط پارامتر دما تعیین می شود که در ابتدا زیاد است و طبق یک زمانبندی مشخص کاهش می یابد. اگر دما به اندازه کافی به آرامی کاهش یابد، الگوریتم تبرید شبیه‌سازی‌شده با احتمال نزدیک به ۱ به مقدار حداکثر جهانی می رسد.

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

]

- جستجوی گام‌برداری تصادفی (random walk)
  - از بین مجاورین یک حالت بصورت تصادفی با توزیع یکنواخت انتخاب می‌شود
    - امکان انتخاب حالت‌های مجاور بدتر
  - کامل است اما کارایی بسیار ضعیفی دارد
- جستجوی تپه‌نوردی کارایی بالایی دارد اما کامل نیست
  - همیشه حالت‌های مجاور بهتر را به صورت حریصانه انتخاب می‌کند
- سعی در تلفیق این دو راه‌برد: تبرید شبیه‌سازی شده
  - سعی در دستیابی به کمال و کارایی بصورت همزمان
  - جستجوی تبرید شبیه‌سازی شده با این رویکرد طراحی شده است

هدف تبرید کاهش انرژی جنبشی است که در واقعیت برای آبدیده کردن فلزات با گرم کردن و سپس سرد کردن تدریجی اتفاق می‌افتد. تبرید شبیه‌سازی‌شده را می‌توان به توپی تشبیه کرد که در یک صفحه نامسطح با کمک لرزاندن قرار است به پایین‌ترین دره برسد، پس امکان فرار از بهینه‌های محلی وجود دارد.

قبل از شروع جستجوی ژنتیکی این سایت را مطالعه کنید واین ویدیو را هم ببینید.]

الگوریتم ژنتیکی

در نهایت، الگوریتم‌های ژنتیکی را ارائه می‌کنیم که نوعی جستجوی پرتو محلی[8]هستند و همچنین به طور گسترده در بسیاری از کارهای بهینه‌سازی استفاده می‌شوند. الگوریتم‌های ژنتیکی با جستجوی پرتویی با k حالت اولیه‌سازی‌شده تصادفی به نام جمعیت آغاز می‌شوند. حالت-ها (یا افراد) به صورت چند رشته با یک الفبای محدود نشان داده می شوند. برای درک بهتر موضوع، دوباره مسئله ۸ وزیر را بررسی می‌کنیم. برای مسئله ۸-وزیر می توانیم هر یک از هشت مهره را با اعداد ۱ تا ۸ نشان دهیم که نشان دهنده مکان هر وزیر در ستون است. هر مهره با استفاده از یک تابع (تابع برازندگی[9]) ارزیابی می شود و بر اساس مقادیر آن تابع رتبه بندی می‌شود.

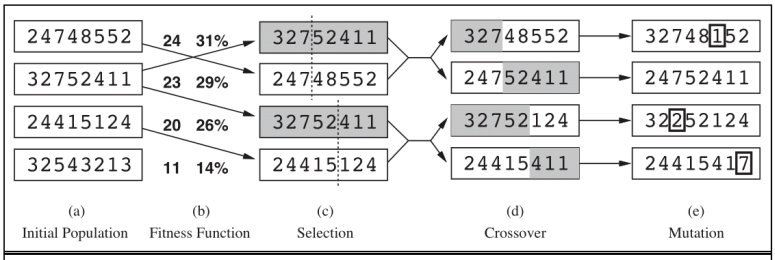


Figure 4.6 The genetic algorithm, illustrated for digit strings representing 8-queens states. The initial population in (a) is ranked by the fitness function in (b), resulting in pairs for mating in (c). They produce offspring in (d), which are subject to mutation in (e).

احتمال انتخاب یک حالت برای «تولید مجدد» به ارزش آن حالت بستگی دارد. فرزندان با عبور از روی رشته‌های والد در نقاط تقاطع تولید می شوند.

```
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 ← RANDOM-SELECTION(population, FITNESS-FN)
      y ← 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 ← LENGTH(x); c ← random number from 1 to n
  return APPEND(SUBSTRING(x, 1, c), SUBSTRING(y, c + 1, n))
```

Figure 4.8 A genetic algorithm. The algorithm is the same as the one diagrammed in Figure 4.6, with one variation: in this more popular version, each mating of two parents produces only one offspring, not two.

[8] Local beam search
[9] Fitness function

الگوریتم‌های ژنتیکی در حین کاوش در فضای حالت و تبادل اطلاعات بین رشته‌ها سعی می‌کنند افزایش پیدا کنند.

یادآوری این نکته که مسائل CSP به طور کلی الگوریتم کارآمدی ندارند که آنها را در زمان چند جمله ای با توجه به تعداد متغیرهای درگیر حل کند، مهم است. با این حال، با استفاده از اکتشافی های مختلف، ما اغلب می توانیم راه حل هایی را در مدت زمان قابل قبولی پیدا کنیم.

]

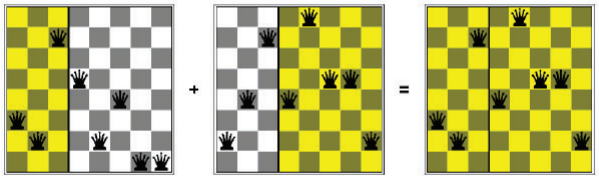
مطالب گفته شده برای الگوریتم ژنتیک به هیچ‌وجه کافی نیست. برای یادگیری کامل این الگوریتم پیشنهاد می‌کنم به این سایت، این سایت و این سایت مراجعه کنید.

جمع‌بندی الگوریتم ژنتیک

- نگهداری یک جمعیت از افراد (حالت‌های مسأله)
- ارزیابی افراد با استفاده از یک تابع برازندگی
  - تابع هدف مسأله جستجو
- بکارگیری نوع خاصی از نمایش فاکتوربندی شده برای نشان دادن افراد
  - هر فرد بصورت یک رشته (کروموزوم – chromosome) از متغیرها (ژن‌ها – Genes) با دامنه مقادیر (alleles) مشخص در نظر گرفته می‌شود
    - مثلاً در رشته‌های بیتی دامنه مقادیر 0 و 1 است
- استفاده از عملگرهای ژنتیکی بجای کنش‌های مسأله
  - عملگرهای تقطیع (crossover) و جهش (mutation)

- مراحل الگوریتم
  - تولید یک جمعیت اولیه از افراد (معمولاً به صورت تصادفی)
  - ارزیابی افراد جمعیت و تعیین برازندگی آنها
  - یک نسل (generation)
    - انتخاب برخی از افراد بر اساس برازندگی به عنوان والدین (parents)
    - بازتولید (reproduction)
      - جفت‌سازی (pairing) والدین
      - اعمال تقطیع روی هر جفت با احتمال P_c و تولید یک جفت فرزند
      - اعمال جهش روی هر یک از فرزندان با احتمال P_m
    - ارزیابی کلیه فرزندان (offspring) تولید شده و تعیین برازندگی آنها
    - جایگزینی (replacement) فرزندان در جمعیت افراد
  - بازگشت به مرحله انتخاب والدین تا محقق شدن شرط توقف الگوریتم

- عملگر تقطیع با توجه به نقطه (های) تقطیع در رشته
  - ترکیب قسمت‌های سمت چپ و راست نقطه تقطیع از دو والد
  - مثال: عملگر تقطیع در مسأله ۸ وزیر



  - تقطیع باعث افزایش سطح پیش‌روی (granularity) جستجو می‌شود
    - شبیه به اعمال چندین کنش در یک حالت از مسأله
  - فاصله والدین به مرور (طی نسل‌ها) نسبت به هم کم می‌شود

- عملگر جهش در یک رشته
  - مقدار هر ژن با احتمالی (P_m) تغییر می‌کند
  - انتخاب تصادفی یکی از مقادیر ممکن جدید برای متغیر (ژن) مشخص شده
  - شبیه به اعمال یک کنش در یک حالت از مسأله
- مثال: یک نسل از الگوریتم در مسأله ۸ وزیر