به نام پروردگار دانایی

هوش مصنوعی

درس هفتم : ماورای جستجوی کلاسیک

سید کاوه احمدی

به این مسائل فکر کنید

- مسائل بهینه‌سازی (Optimization)
  - شامل تابع هدفی که می‌خواهیم مقدار آنرا حداکثر یا حداقل کنیم و تعدادی محدودیت نیز جلوی روی ما قرار دارد.
    - معمولا رسیدن به پاسخ نزدیک به بهینه میسر است.
- مسائلی که مسیر رسیدن به هدف مهم نیست.
  - در این مسائل فضایی از همه‌ی حالت‌های کامل (و نه مرحله به مرحله) برای پیدا کردن پاسخ مورد نیاز است.
  - مثلا در هشت وزیر نیاز به همه وضعیت‌هایی که همه‌ی هشت وزیر در صفحه وجود دارد برای پیدا کردن حالت هدف نیاز داریم.
    - `state space = set of compelete configurations`
    - همه آنها بالقوه هدف هستند.

جستجوی محلی (Local Search)

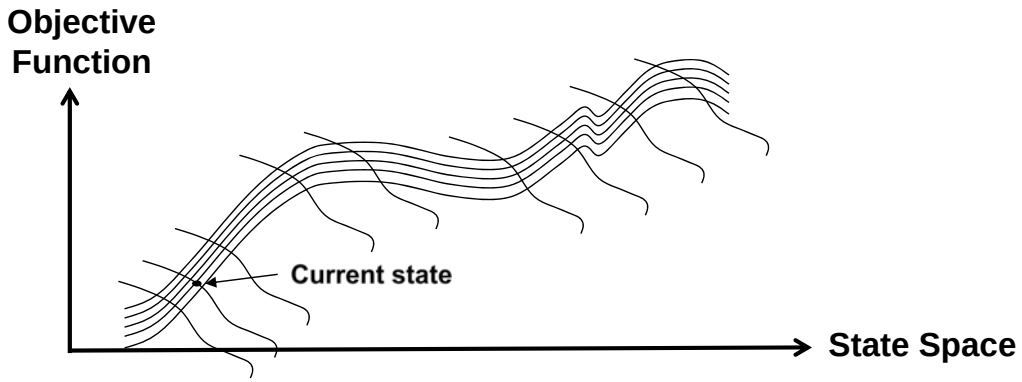
- الگوریتمهای جستجوی کلاسیک که تا به حال مطرح شدند برای پویش سیستماتیک فضاهای حالت طراحی شده‌اند این سیستمی بودن از طریق ذخیره یک یا چند مسیر در حافظه و با ثبت جایگزین‌های پویش شده در هر نقطه در طول مسیر، حاصل می شوند. زمانی که هدف یافت شد، مسیر به آن هدف نیز شامل پاسخی برای مسئله است.
- در بسیاری از مسائل مسیر تا هدف بی اهمیت است.
  - برای مثال، در مسئله هشت وزیر آنچه مهم است چیدمان وزیرها در نهایت است نه توالی چیدمان آنها.
  - این گروه مسائل شامل کاربردهای بسیار مهمی همانند طراحی مدارات مجتمع، طرح کف، برنامه ریزی کاری، برنامه نویسی خودکار، بهینه سازی شبکه های مخابراتی، مسیر گزینی متحرک و مدیریت پست است.

جستجوی محلی (Local Search)

- بر مبنای یک حالت جاری واحد (که بالقوه می‌تواند هدف باشد) به جای چندین مسیر عمل می‌کنند و عموما با یک تغییر کوچک تنها به حالت همسایگی منتقل می شوند.
- دارای خصوصیات و مزایای زیر هستند:
  - سیستماتیک نسیت
  - از حافظه بسیار کمی استفاده می‌کنند.
  - می توانند اغلب راه حل‌های منطقی در فضاهای حالت بزرگ یا نامتناهی (پیوسته) را که الگوریتم‌های سیستمی در آنها ناکارامد هستند، را پیدا کند.
- این الگوریتم‌ها علاوه بر یافتن هدف، برای حل مسائل بهینه‌سازی نیز مفیدند
  - یافتن بهترین حالت بر اساس تابع هدف
- تحت عنوان الگوریتم‌های اصلاح تکراری (Iterative Improvement Algorithms) نیز خطاب می‌شوند

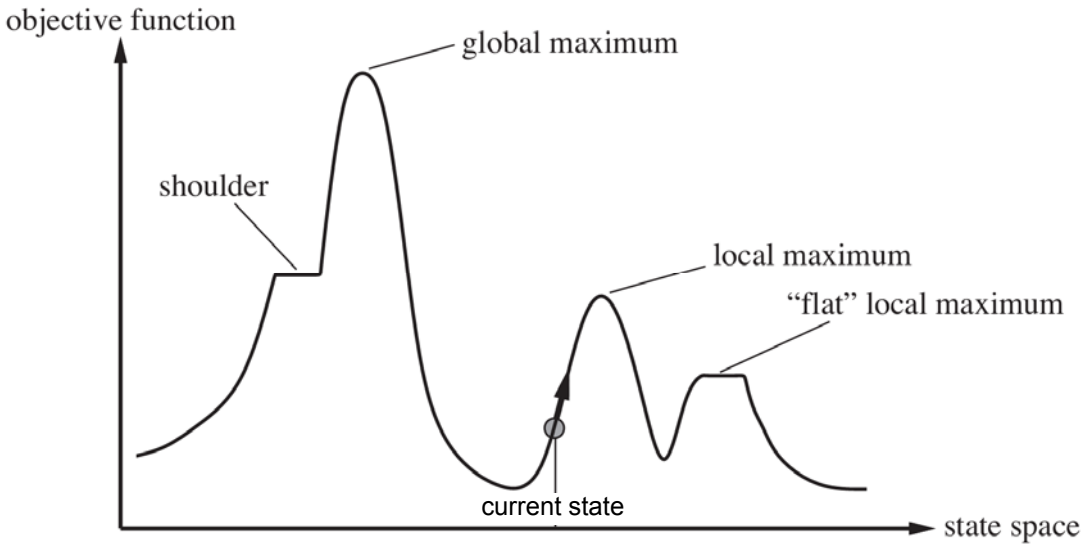
جستجوی محلی (Local Search)

- در این الگوریتم‌ها برای فضای حالت مسئله، سطحی بر اساس تابع ارزیابی تعریف شده رسم می‌شود.
  - در این نمودار ارتفاع هر نقطه از فضای حالت، مقدار تابع ارزیابی در آن حالت را نشان می‌دهد.
  - الگوریتم‌های اصلاح تکراری سعی بر یافتن قله‌هایی بر روی سطح حالات دارند.



چشم اندازی از فضای حالت

- نمایش دهنده مکان (وضعیت) و بلندی (مقدار تابع هدف یا اکتشافی) در فضای حالت
  - الگوریتم‌های جستجوی محلی این چشم انداز را پویش می‌کنند.



چشم اندازی از فضای حالت

- اگر بلندی وابسته به تابع هدف باشد، هدف پیدا کردن بلندترین نقطه (بیشینه عمومی) است.
- اگر بلندی وابسته به هزینه باشد، هدف پیدا کردن پست‌ترین دره (کمینه عمومی) است.
  - با ضرب یک منفی در تابع هزینه، آنرا تبدیل به مسئله بالایی می‌کنیم! (جستجوی بلندترین نقطه)
- الگوریتم‌های محلی چون فقط به همسایگی حالت فعلی نگاه می‌کنند، نمی‌توانند از نقاط بیشینه (و کمینه) محلی عبور کنند.

الگوریتم‌های جستجوی محلی

- جستجوی تپه نوردی (Hill-climbing)
- جستجوی بازپخت شبیه‌سازی شده (Simulated annealing)
- جستجوی پرتو محلی (local beam shearch)
- الگوریتم‌های ژنتیک

الگوریتم تپه‌نوردی (Hill-climbing)

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

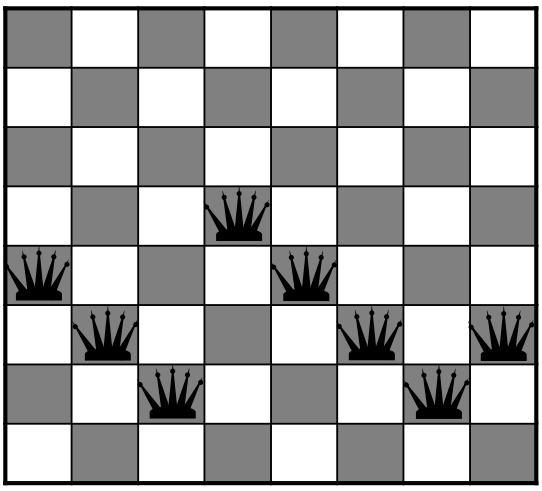
current ← MAKE-NODE(*problem*.INITIAL-STATE)
loop do
  neighbor ← a highest-valued successor of *current*
  if neighbor.VALUE ≤ current.VALUE **then return** *current*.STATE
  current ← *neighbor*

الگوریتم تپه‌نوردی (Hill-climbing)

- شامل حلقه‌ای که در جهت افزایش مقدار تابع هدف حرکت می‌کند. (به طرف بالای تپه – نسخه تندترین شیب)
- زمانی متوقف می‌شود که به نقطه اوجی برسد که همسایه‌ای بالاتر نداشته باشد.
  - رسیدن به بلندترین قله در همسایگی حالت فعلی، شرط خاتمه است.
- الگوریتم تپه نوردی جستجوی محلی حریصانه نیز نامیده می‌شود.
  - یک همسایه خوب را بدون فکر برای حرکت بعد انتخاب می کند
- این الگوریتم شامل درخت جستجو نیست بنابراین ساختار داده گره فقط وضعیت و مقدار تابع هدف آن حالت را نگهداری می‌کند.
- این الگوریتم ماورای همسایگان گره جاری را نگاه نمی‌کند.
  - این عمل مشابه یافتن نقطه بلند کوه اورست در یک مه غلیظ است

حل مسئله ۸ وزیر با استفاده از جستجوی تپه نوردی

- باید از فرموله سازی حالت کامل استفاده کنیم
  - یک وزیر در هر ستون
- تابع مابعد: حرکت هر وزیر در همان ستون
  - هر حالت ۵۶=۷*۸ مابعد دارد
- تابع هدف: تعداد تهدیدها صفر شود.



h = 3 + 5 + 4 + 4 + 5 + 5 + 5 + 3 = 34/2 = 17

حل مسئله ۸ وزیر با استفاده از جستجوی تپه نوردی

- مقادیر h تمام مابعدها
  - بهترین مابعدها علامت گذاری شده‌اند

18	**12**	14	13	13	**12**	14	14
14	16	13	15	**12**	14	**12**	16
14	**12**	18	13	15	**12**	14	14
15	14	14	♛	13	16	13	16
♛	14	17	15	♛	14	16	16
17	♛	16	18	15	♛	15	♛
18	14	♛	15	15	14	♛	16
14	14	13	17	**12**	14	**12**	18

خصوصیات جستجوی تپه نوردی

- در پنج حرکت از وضعیتی با h=17 به وضعیتی با h=1 که بسیار به پاسخ نزدیک است می‌رسد اما تمام مابعدهای این وضعیت هزینه بیشتری دارند.
  - پاسخ محلی کاذب
- الگوریتم تپه نوردی گاهی جستجوی محلی حریصانه نیز نامیده می‌شود. زیرا که یک همسایه خوب را بدون فکر برای حرکت بعد انتخاب می‌کند.
  - از حافظه برگشتی استفاده نمی‌کند و فقط رو به جلو حرکت می‌کند.
  - سریع است.
  - بسیار خوب به سمت پاسخ حرکت می‌کند. اما ممکن است متوقف شود و به پاسخ نرسد.

مشکلات جستجوی تپه نوردی

1. **بیشینه یا کمینه محلی (Local Maxima):**
   - یک ماکزیمم محلی، قله‌ای است که پائین‌تر از بلندترین قله درفضای حالت است. زمانی که روی ماکزیمم محلی هستیم، الگوریتم متوقف می‌شود چراکه نقطه‌ای است که از تمام همسایه‌هایش مرتفع‌تر است هرچند که پاسخ بهینه نیست.
     - فرزندان یک گره ماکزیمم محلی، همگی دارای مقدار کمتری از پدرشان هستند.

مشکلات جستجوی تپه نوردی

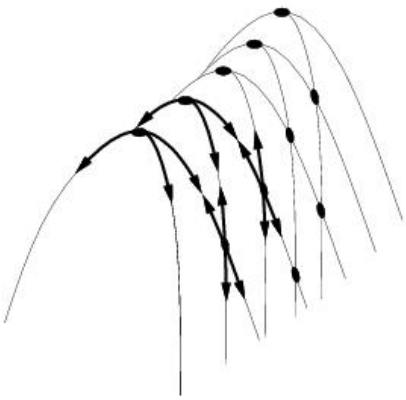
2. سطح صاف یا فلات (Plateaux - Flat):

- فلات یا سطح صاف محوطه‌ای از فضای حالت است که تابع ارزیابی یکنواخت باشد.
- سطح صاف ممکن است
  - محل صافی باشد که هیچ خروجی بالا رونده‌ای ندارد.
  - شانه‌ای باشد که از آن امکان پیشرفت وجود داشته باشد.
- جستجوی تپه نوردی ممکن است راهی برای خروج از سطح صاف (فلات) نیابد.
- جستجوی محلی مبتنی بر شیب و حرکت در جهت گرادیان (صعودی یا نزولی) است و در سطوح صاف یا فلات‌ها که گرادیان صفر است، جستجو معنی نخواهد داشت!

مشکلات جستجوی تپه نوردی

3. تیغه (Ridges):

- تیغه، یک ناحیه در فضای جستجو است که بالاتر از نواحی اطراف خود باشد اما عبور از آن با حرکتهای تکی در هیچ جهات امکان پذیر نباشد
- در یک ناحیه تعداد متعددی پاسخ محلی نزدیک به هم داریم که الگوریتم در آنها گیر خواهد کرد.
- تیغه حاصل یک دنباله از حداکثرهای محلی است که برای الگوریتم حریصانه، پویش آن بسیار دشوار است.
- تیغه، یک ناحیه در فضای جستجو است که بالاتر از نواحی اطراف خود باشد اما عبور از آن با حرکتهای تکی در هیچ جهات امکان پذیر نباشد.
  - شبکه حالات (دایره‌های تیره) در حین تپه نوردی از چپ به راست تقویت می‌شوند و دنباله‌ای از حداکثرهای محلی را تولید می‌کنند که بهم متصل نیستند.
  - از هر حداکثر محلی تمامی اعمال موجود بسوی پایین اشاره می کنند.



حل مشکل سطوح صاف

- جستجوی تپه نوردی ممکن است راهی برای خروج از سطح صاف نیابد.
  - اگر به سطح صافی برخورد کردیم (وضعیتی که بهترین مابعد هزینه‌ای مشابه گره جاری دارد)، حرکت را به یک طرف ادامه دهیم به این امید که سطح صاف، یک شانه باشد.
    - اگر چندین مرحله پیش رفتیم و موفقیتی حاصل نشد، می‌توانیم جهت حرکت را تغییر دهیم!
    - این ایده در مسئله هشت وزیر نمونه‌های حل شده را از ۱۴٪ به ۹۴٪ افزایش می‌دهد

انواع تپه نوردی: تپه نوردی تصادفی (Stochastic)

- تپه نوردی تصادفی یکی از حرکات در جهت شیب را تصادفی انتخاب می‌کند.
  - بطور تصادفی حرکت‌هایی را به سمت بالای تپه انتخاب می‌کند و نه لزوما مسیر با تندترین شیب را.
- احتمال انتخاب با زیاد شدن عمق حرکت کم می‌شود.
- این روش نسبت به روش "سربالایی زیاد" کندتر همگرا می شود ولی در بعضی موارد راه حل‌های بهتری را می‌یابد.

انواع تپه نوردی: تپه نوردی اولین انتخاب (First Choice)

- تپه نوردی تصادفی را به این صورت پیاده‌سازی می‌کند که تولید تصادفی مابعدها را آنقدر ادامه دهد که به گرهی برسد که از وضعیت فعلی بهتر است (نه لزوما بهترین گره).
  - اگر حالت دارای مابعدهای زیادی باشد این راهبرد مناسب است.

انواع تپه نوردی: تپه نوردی با شروع مجدد تصادفی (Random-Restart)

- ایده اصلی الگوریتم: اگر موفق نشدید مجددا سعی کنید.
- این الگوریتم مجموعه‌ای از جستجوهای تپه نوردی را اداره می‌کند که از حالت‌های اولیه تصادفی شروع می‌شوند و با رسیدن به هدف متوقف می‌شوند.
- این الگوریتم با احتمال نزدیک به ۱ کامل است. (سرانجام حالت هدف را می‌یابد)
- اگر احتمال موفقیت هر جستجوی تپه نوردی p باشد، تعداد شروع مجدد مورد نیاز $1/p$ است.
  - برای مسئله‌ی ۸ وزیر که حرکت یکطرفه در سطح صاف $p \approx 0.14$ است و با ۷ مرتبه تکرار به هدف می‌رسیم. (۶ شکست و ۱ موفقیت)
  - این رهیافت برای سه میلیون وزیر(3/000/000)کمتر از یک دقیقه می تواند راه حل پیدا کند.

انواع تپه نوردی: تپه نوردی با شروع مجدد تصادفی (Random-Restart)

- این الگوریتم برای مسائلی که تعداد زیادی حالت مابعد(هزاران) داشته باشد خوب است.
- بسیاری از مسائل واقعی دارای سطحی شبیه جوجه تیغی هستند. آنگاه این الگوریتم در تمام حالات نمی‌تواند در زمانی بهتر از زمان نمایی یک راه حل خوب و منطقی را پیدا کند.

جستجوی باز پخت شبیه سازی شده (Simulated Annealing)

- موفقیت تپه‌نوردی خیلی به ظاهر فضای حالت (سطح) بستگی دارد.
  - اگر سطوح صاف و حداکثرهای محلی کم باشند، تپه نوردی با شروع مجدد تصادفی پاسخ خوبی را به سرعت پیدا می‌کند.
    - مسائل بسیاری هستند که داری سطح صافند
    - مسائل NP-hard معمولا دارای بیشینه محلی‌های زیادی هستند

جستجوی باز پخت شبیه سازی شده

- الگوریتم تپه نوردی حرکت رو به پایین ندارد یعنی گره‌ای با هزینه بیشتر را انتخاب نمی‌کند.
- تپه نوردی تصادفی در برخی از مواقع حرکت رو به پایین نیز دارد و حرکات رو به پایین به راحتی در مراحل اولیه باز پخت قابل قبولند اما در مراحل پایانی به سختی پذیرفته می‌شوند.

جستجوی بازپخت شبیه سازی شده
- مزایای جستجوی تپه نوردی
- مزایای جستجو با حرکت تصادفی

جستجوی باز پخت شبیه سازی شده

- **بازپخت شبیه‌سازی شده:**
  - اگر فلز را تا نقطه جوشش ذوب کنیم، پیوندهای کوالانسی بین ملکول‌ها از بین می‌رود و ملکول‌ها به هم بسیار نزدیک می‌شوند. اگر آنرا ناگهان سرد کنیم، فلز فرصت بازسازی پیوندهای کوالانسی خود را پیدا نمی‌کند و بسیار مقاوم می‌شود (به خاطر فاصله کم ملکول‌ها)
- ایده اصلی الگوریتم پازپخت شبیه‌سازی شده از همین قانون می‌آید:
  - سیستم با درجه حالت بالا شروع می‌کند. در این دما رفتارهای غیرمنطقی (bad moves) در سیستم دیده می‌شود.
  - با کم شدن درجه حالت، رفتارهای بد و اعوجاجی کم می‌شود.
    - در دسته بزرگی از مسائل وجود دست‌اندازهای اولیه دیده می‌شود.
    - دمای بالا اول به ما کمک می‌کند که دست اندازهای اولیه در مسیر را رد کنیم. با کم شدن دما، سیستم رفتار مشابه تپه‌نوردی خواهد بود.
  - این فرآیند کاملا تصادفی است.

جستجوی باز پخت شبیه سازی شده

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 $e^{\Delta E/T}$

- T: درجه حرارت
- t: زمان
- schedule: تابع کاهش دما نسبت به زمان

جستجوی باز پخت شبیه سازی شده

- جستجوی بازپخت شبیه سازی شده کامل و کارآمد است
- گرادیان نزولی (حداقل سازی هزینه) در برابر تپه نوردی و باز پخت شبیه سازی شده:
  - توپ در فرود از تپه به عمیق‌ترین شکاف می‌رود. با تکان دادن سطح توپ از بیشینه محلی خارج می‌شود (نباید طوری تکان داده شود که از بیشینه‌ی عمومی خارج شود.)
    - با تکان شدید شروع می‌شود. (دمای زیاد)
    - به تدریج مدت تکان دادن‌ها کاهش می‌یابد. (به دمای پایین‌تر)

جستجوی باز پخت شبیه سازی شده

- ایده اصلی (اول بیشتر explore سپس بیشتر exploit)
  - انتخاب تصادفی یک حرکت.
    - اگر وضعیت را بهبود دهد پذیرفته می‌شود. در غیر اینصورت با احتمال کمتر از یک پذیرفته می‌شود.
      - احتمال با توجه به بد بودن وضعیت به طور نمایی کم می‌شود.
      - با پیش رفتن جستجو (کاهش دما) این احتمال نیز کاهش می‌یابد.
- اگر کاهش زمانبندی دما به اندازه کافی آهسته باشد، الگوریتم یک بهینه عمومی را با احتمال ۱ می‌یابد.
  - این الگوریتم ابتدا برای مسئله های طرح VLS I در دهه ی ۱۹۸۰ بکار گرفته شد.

جستجوی پرتو محلی (local beam search)

- نگهداری فقط یک گره در حافظه برای رفع مشکل حافظه بسیار افراطی است.
- به جای یک حالت، K حالت را در حافظه نگهداری می‌کند.
- الگوریتم با K حالت تصادفی شروع می‌کند.
  - حالت اولیه: k حالت تصادفی
  - تمامی مابعدهای همه kها حالت تولید می‌شوند
    - اگر یکی از مابعدها هدف بود، تمام می‌شود
  - وگرنه بهترین مابعد (kتا بهترین) را انتخاب کرده، تکرار می‌کند

جستجوی پرتو محلی

- تفاوت با جستجوی شروع مجدد تصادفی، فقط اجرای موازی به جای ترتیبی نیست.
  - در جست و جوی شروع مجدد تصادفی، هر فرایند مستقل از بقیه اجرا می‌شود
  - در جست و جوی پرتو محلی، اطلاعات مفیدی بین k فرایند موازی مبادله می‌شود
    - اگر حالتی چندین مابعد خوب داشته باشد به بقیه اطلاع می‌دهد.
    - به هم می‌گویند علف کجا سبزتر شده!
  - مشکل فقدان پراکندگی
    - همه به سمت یک منطقه حرکت می‌کنند
- جست و جوی پرتو تصادفی
  - به جای انتخاب بهترین k از مابعدها، k جانشین تصادفی را بطوریکه احتمال انتخاب یکی تابعی صعودی از مقدارش باشد، انتخاب می‌کند (تا حدی به انتخاب طبیعی شباهت دارد)

جستجوی محلی در فضاهای پیوسته

- مثالی از یک مسئله:
  - فرض کنید می‌خواهیم سه فرودگاه در رومانی ایجاد کنیم که مجموع مربعات فواصل هر شهر به نزدیکترین فرودگاه کمینه باشد. فضای حالت توسط مختصات فرودگاه‌ها مشخص می‌شود
  - (x1, y1), (x2, y2), (x3, y3) (یک فضای ۶ بعدی)
  - یک روش اجتناب از مسئله های پیوسته، گسسته کردن همسایه‌ی هر حالت و بکارگیری یکی از روش های جستجوی عنوان شده می‌باشد
    - هر فرودگاه می‌تواند به اندازه ی $\pm\partial$ در جهت x یا y حرکت کند یعنی برای هر حالت ۱۲ جانشین داریم.
  - روش دیگر استفاده از تپه نوردی تصادفی و شبیه سازی حرارت بدون گسسته کردن فضا است برای این کار می‌توان بردارهای تصادفی به طول ∂ را تولید کرد.

$$\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla f \; where \nabla f = \left\{ \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots \right\}$$

  - روش کارآمد دیگر، روش نیوتن رافسون است. (حل معادلاتی به شکل g(x)=0)

نکته مهم

- در تمام مواردی که با یک مجموعه از پاسخ‌های احتمالی مواجه هستیم (K حالت در پرتو محلی) و قصد انتخاب حالت بهتری را داریم (مابعد)، نحوه انتخاب تاثیر مستقیمی در نتیجه الگوریتم خواهد داشت.
  - در الگوریتم‌های بهینه سازی مبتنی برجمعیت، parent selection و servival selection دو چالش بزرگ پیش روی ما هستند.

نکات پایانی

- ادامه فصل را خودتان مطالعه کنید!
  - جستجوی موضعی در فضای پیوسته
  - جستجو با مشاهدات نسبی
  - مسائل جستجوی بر خط
- الگوریتم‌های ژنتیک
  - در کارگاه مفصل بحث می‌کنیم.
- نحوه عملکرد جستجوی باز پخت شبیه سازی شده در عمل:
  - بعد از بررسی سیستم‌های فازی در کارگاه در موردش بیشتر صحبت می‌کنیم.
  - مقاله پایه را ببینید:

– Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. (1983). "*Optimization by Simulated Annealing*". *Science* (4598): 671–680

هوش مصنوعی ۹۱

- فرض کنید الگوریتم Local Beam Search با k=1 اجرا می‌شود. این جستجو معادل کدام یک از جستجوهای زیر است؟

1. Hill Climbing
2. Simulated Annealing
3. Genetic Algorithm
4. Constraint Satisfaction

هوش مصنوعی ۹۱

- گزینه ۱ صحیح است.
  - اجرای Local Beam Search با k=1 به معنی اجرای یکبار تپه نوردی است.

آی تی ۹۱

- کدامیک از گزینه‌های زیر غلط است؟

1. الگوریتم شبیه‌سازی حرارت حتما از بهینه محلی فرار می‌کند
2. الگوریتم تپه نوردی حتما در بهینه محلی گیر می‌کند
3. الگوریتم پرتو محلی ممکن است بهینه عمومی را پیدا کند
4. الگوریتم ژنتیک ممکن است از بهینه محلی فرار کند

آی تی ۹۱

- گزینه ۱ صحیح است.
  - احتمالا منظور طراح از گزینه ۲ این بوده که در صورتی که جستجو بهینه محلی را بیابد نمی‌تواند از آن فرار کند. در غیر این صورت بهینه پیدا شده توسط تپه نوردی ممکن است محلی یا عمومی باشد.
  - الگوریتم شبیه‌سازی حرارت با احتمال نزدیک به ۱ از بهینه محلی فرار می‌کند.

آی تی ۹۰

- در الگوریتم Simulated Annealing در صورتی که دما بالا باشد، کدام یک از عبارات زیر صحیح است؟

1. فقط جستجوی عمومی انجام می‌شود.
2. فقط جستجوی محلی انجام می‌شود.
3. جستجوی عمومی و محلی انجام می‌شود.
4. جستجوی عمومی و ممکن است جستجوی محلی هم انجام شود.

آی تی ۹۰

- گزینه ۳ صحیح است.

– اساسا جستجوهای محلی، محلی جستجو می‌کنند! اما Simulated Annealing ممکن است با یک احتمالی به وضعیت بدتری نسبت به وضعیت فعلی خود نیز برود (جستجوی عمومی). این احتمال در اوایل جستجو زیاد است و با پیش رفتن جستجو کاهش می‌یابد. با توجه با این توضیحات گزینه ۴ هم می‌تواند صحیح باشد. اما با توجه به اینکه ماهیت این دسته از جستجوها برپایه جستجوی محلی است و نه عمومی، گزینه ۳ مد نظر طراح بوده است.

آی تی ۸۸

- کدام عبارت صحیح است؟

1. حافظه مصرفی A* از تپه‌نوردی کمتر است.
2. A* جواب‌های بهتری نسبت به جستجو با هزینه یکسان می‌یابد.
3. پیچیدگی فضایی جستجوی دوسویه از جستجوی عرض اول کمتر است.
4. پیچیدگی فضایی جستجوی تعمیق تکراری از جستجوی عمق اول بیشتر است.

آی تی ۸۸

- گزینه ۳ صحیح است.

آی تی ۹۰

- کدام یک از موارد زیر در مورد الگوریتم ژنتیک نادرست است؟

1. یکی از مراحل الگوریتم ژنتیک تولید فرزندان است.
2. یکی از مراحل الگوریتم ژنتیک انتخاب والدین است.
3. یکی از مراحل الگوریتم ژنتیک انتخاب بازماندگان است.
4. شرط خاتمه در الگوریتم ژنتیک می‌تواند نسل باشد.

آی تی ۹۰

- گزینه ۳ صحیح است.

آی تی ۹۱

- کدامیک از عبارات زیر در مورد الگوریتم ژنتیک غلط است؟

1. جمعیت اولیه باید به صورت تصادفی یکنواخت تعیین گردد
2. انتخاب بازماندگان متناسب با شایستگی است
3. فرزندان توسط بازترکیبی و جهش تولید می‌شوند.
4. شرط خاتمه می‌تواند ماکزیمم تعداد نسل باشد

آی تی ۹۱

- گزینه ۲ صحیح است.

آی تی ۸۴

- هرگاه در یک مساله به دنبال کلیه جواب‌ها باشیم کدام یک از روش‌های زیر مناسب‌تر است؟

1. A*
2. عمق اول
3. تپه نوردی
4. تولید و آزمون

آی تی ۸۴

- گزینه ۴ صحیح است.

آی تی ۸۳

- روش جستجوی تولید و آزمون (Generate & Test) برای حل کدام یک از مسائل زیر مناسب‌تر است؟

1. شطرنج
2. معمای ۸
3. ۸ وزیر
4. فروشنده دوره گرد

آی تی ۸۳

- گزینه ۳ صحیح است.