

بسمه تعالی

هوش مصنوعی

جستجو در محیطهای پیچیده - ۴

نیمسال اول ۱۴۰۴-۱۴۰۳

دکتر مازیار پالهنک

آزمایشگاه هوش مصنوعی

دانشکده مهندسی برق و کامپیوتر

دانشگاه صنعتی اصفهان

یادآوری

- الگوریتمهای جستجوی محلی
- حالت فعلی را نگهدار - سعی کن آن را بهبود دهی
- جستجوی تپه نوردی
- سردشدن شبیه سازی شده
- جستجوی پرتو محلی، الگوریتم ژنتیک
- جستجوی محلی در فضای پیوسته
- جستجو با اعمال قطعی (پاسخ یک دنباله)
- جستجو با اعمال غیرقطعی
- حل یک طرح شرطی، استفاده از درخت AND-OR با استفاده از فضای حالت
- جستجو برای عامل بدون حسگر
- جستجو در فضای باور همانند حالت مشاهده پذیر با تعمیم تعریف اجزاء مسئله
- جستجو در محیط نیمه مشاهده پذیر
- حل یک طرح شرطی، استفاده از درخت AND-OR با استفاده از فضای باور

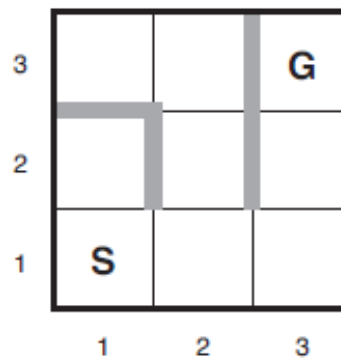
جستجوی برخط و محیطهای ناشناخته

- جستجوی برون خط
 - محاسبه حل کامل قبل از اجرای حل
- جستجوی برخط
 - انجام عمل، مشاهده، محاسبه عمل بعد
 - مناسب برای
 - محیطهای پویا و نیمه پویا (جلوگیری از محاسبات با دید طولانی)
 - در محیطهای غیرقطعی (تمرکز تلاش محاسباتی با توجه به موقعیت بوجود آمده نه تمرکز روی وضعیتهایی که بندرت ممکن است پیش آیند)
 - محیطهای ناشناخته (هنگامی که حالات و نتایج اعمال ناشناخته هستند)
- محیطهای ناشناخته – مسائل اکتشافی

مسائل جستجوی برخط

- فعلا با فرض محیط مشاهده پذیر و قطعی
- دانش عامل:
- $Actions(s)$: لیستی از اعمالی که عامل در حالت s می تواند انجام دهد باز می گرداند.
- تابع هزینه $c(s, a, s')$ - تا عامل نداند s' نتیجه عمل a در s است قابل استفاده نیست.
- $Goal-Test(s)$

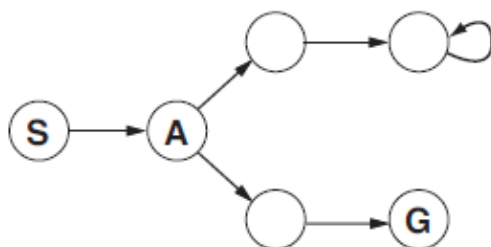
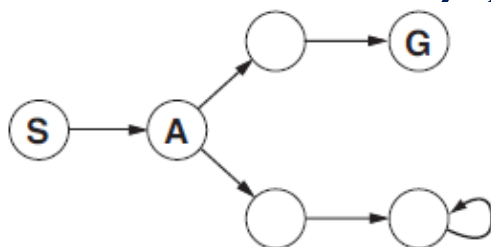
■ عامل نمی تواند مقدار $\text{Result}(s,a)$ را مشخص کند مگر در s عمل a را انجام دهد.



■ عامل ممکن است به یک مکاشفه قابل پذیرش $h(s)$ دسترسی داشته باشد.

- نوعاً عامل علاقمند به طی کردن کوتاهترین مسیر تا هدف است.
- یا ممکن است بخواهد تمامی محیط را اکتشاف کند.
- معمول است که هزینه مسیر انجام شده با هزینه مسیر در حالتی که عامل تمام محیط را می شناخت مقایسه شود.
- یعنی کوتاهترین مسیر (یا کوتاهترین اکتشاف کامل)
- به آن **نسبت رقابتی (competitive ratio)** گفته می شود.
- علاقمند که کمترین مقدار باشد.

- گاهی این مقدار ممکن است بی نهایت شود.
- مشکل ساز هنگامی که اعمال برگشت پذیر نباشند.
- امکان رسیدن به بن بست.



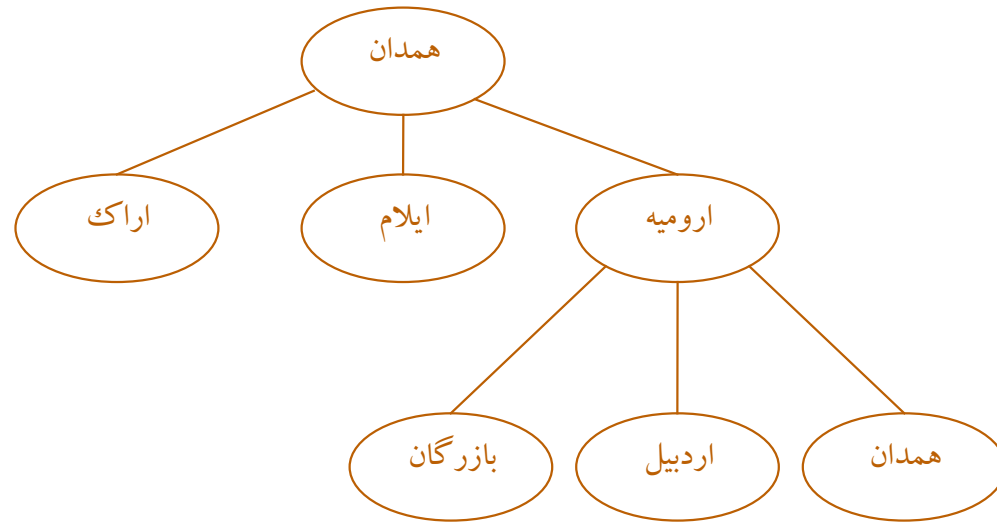
(a)

■ فرض می کنیم محیط بطور امن قابل اکتشاف (safely explorable) است.

■ یعنی برخی از حالات هدف از هر حالت قابل دسترسی قابل دستیابی می باشند.

عواملهای جستجوی برخط

- پس از عمل، عامل درک می کند در چه حالتی قرار دارد.
- بسط نقشه محیط
- استفاده از نقشه برای عمل بعدی
- متفاوت با روشهای جستجوی برون خط همانند A^*
- در A^* می توان به قسمتی دیگر از فضای حالت رفت،
- در جستجوی برخط نمی توان چنین کرد.
- جستجوی برخط در دنیای واقعی است ولی برون خط بر روی مدل دنیا



- فقط جستجو در حوالی حالتی که در آن قرار دارد.
- همانند عمق نخست (بجز هنگام عقبگرد)

Figure 4.21

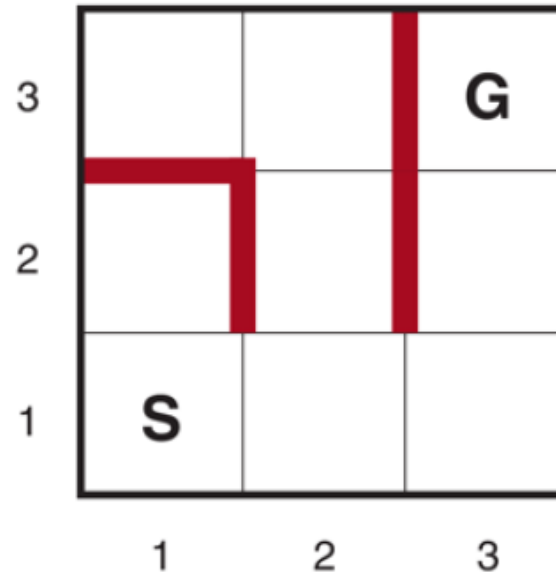
function ONLINE-DFS-AGENT(*problem*, s') **returns** an action
 s, *a*, the previous state and action, initially null
 persistent: *result*, a table mapping (*s*, *a*) to s' , initially empty
 untried, a table mapping *s* to a list of untried actions
 unbacktracked, a table mapping *s* to a list of states never backtracked to

if *problem*.IS-GOAL(s') **then return** *stop*
if s' is a new state (not in *untried*) **then** *untried*[s'] \leftarrow *problem*.ACTIONS(s')
if *s* is not null **then**
 result[*s*, *a*] \leftarrow s'
 add *s* to the front of *unbacktracked*[s']
if *untried*[s'] is empty **then**
 if *unbacktracked*[s'] is empty **then return** *stop*
 else *a* \leftarrow an action *b* such that *result*[s' , *b*] = POP(*unbacktracked*[s'])
else *a* \leftarrow POP(*untried*[s'])
s \leftarrow s'
return *a*

چون همهٔ اعمالی که می توان در s' انجام داد تلاش شده اند

An online search agent that uses depth-first exploration. The agent can safely explore only in state spaces in which every action can be “undone” by some other action.

■ تلاش کنید رد اجرای الگوریتم را روی محیط زیر بیابید:

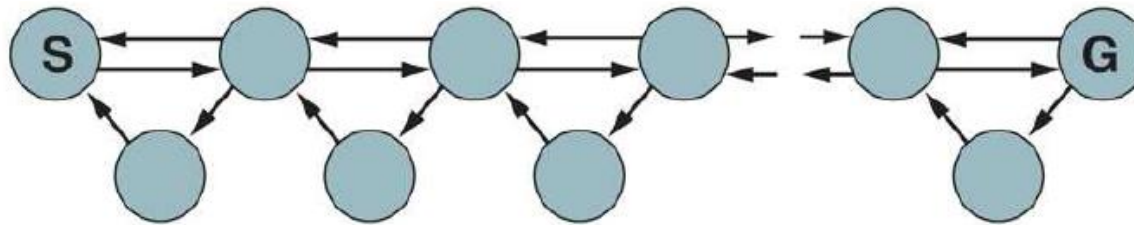


جستجوی محلی بر خط

- جستجوی تپه نوردی نیز همانند عمق نخست حالت محلی را دارد.
- مشکل بهینه محلی برای اکتشاف
- نمی توان باز شروع تصادفی داشت.
- گام زدن تصادفی
- انتخاب تصادفی یکی از اعمال
- می توان ثابت کرد که گام زدن تصادفی نهایتاً هدف را یافته (یا اکتشاف را کامل می کند).

■ گاهی خیلی طولانی

Figure 4.22



An environment in which a random walk will take exponentially many steps to find the goal.

■ می توان از بسط تپه نوردی با حافظه استفاده کرد.

■ ذخیره بهترین تخمین فعلی تا هدف $H(s)$

■ شروع با $h(s)$

■ اصلاح آن حین اکتشاف

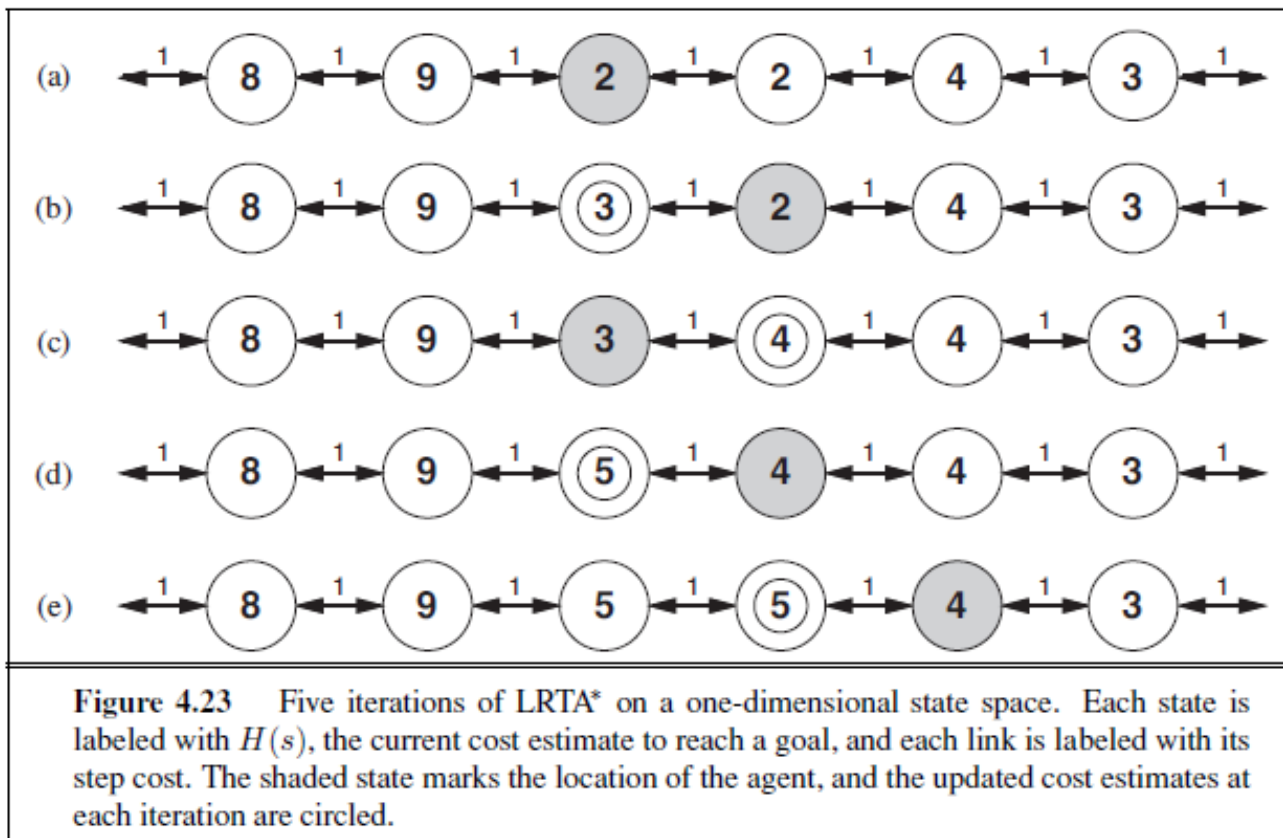


Figure 4.23 Five iterations of LRTA* on a one-dimensional state space. Each state is labeled with $H(s)$, the current cost estimate to reach a goal, and each link is labeled with its step cost. The shaded state marks the location of the agent, and the updated cost estimates at each iteration are circled.

Figure 4.24

function LRTA*-AGENT(*problem*, s' , h) **returns** an action
 s , a , the previous state and action, initially null
 persistent: *result*, a table mapping (s, a) to s' , initially empty
 H, a table mapping s to a cost estimate, initially empty

if IS-GOAL(s') **then return** *stop*
 if s' is a new state (not in H) **then** $H[s'] \leftarrow h(s')$
 if s is not null **then**
 $result[s, a] \leftarrow s'$
 $H[s] \leftarrow \min_{b \in \text{ACTIONS}(s)} \text{LRTA}^*\text{-COST}(s, b, result[s, b], H)$
 $a \leftarrow \operatorname{argmin}_{b \in \text{ACTIONS}(s)} \text{LRTA}^*\text{-COST}(problem, s', b, result[s', b], H)$
 $s \leftarrow s'$
 return a

function LRTA*-COST(*problem*, s, a, s', H) **returns** a cost estimate
 if s' is undefined **then return** $h(s)$
 else return $problem.\text{ACTION-COST}(s, a, s') + H[s']$

LRTA*-AGENT selects an action according to the values of neighboring states, which are updated as the agent moves about the state space.

- الگوریتم $LRTA^*$ ضمانت می کند که در محیطهای محدود و قابل اکتشاف امن هدف را بیابد.
- برای فضاهای حالت نامحدود کامل نیست.



- دقت نمائید که پاورپوینت ابزاری جهت کمک به یک ارائه شفاهی می باشد و به هیچ وجه یک جزوه درسی نیست و شما را از خواندن مراجع درس بی نیاز نمی کند.
- لذا حتماً مراجع اصلی درس را مطالعه نمائید.
- حضور فعال در کلاس دارای امتیاز است.