هوش مصنوعی و سیستم‌های خبره

پاسخنامه تمرین سری سوم

مدرس:

دکتر محمدرضا محمدی

طراحان:

محمد یارمقدم، امیرعلی پاکدامن

سوال اول)

الف) همانگونه که می دانیم الگوریتم A*

نحوه ی EXPAND کردن نود ها در A* با منطق زیر اجرا میشود :

Expand a node n most likely to be on an optimal path

Expand a node n the cost of the best solution through n is optimal

Expand a node n with lowest value of g(n) + h * (n)

- g(n) is the cost from root to n
- h * (n) is the optimal cost from n to the closest goal

We seldom know h * (n) but might have a heuristic approximation h(n)

A * = tree search with priority queue ordered by f(n) = g(n) + h(n)

تابع هیوریستیکی ما هم admissible است هم consistant است :

در نتیجه با توجه به نکات بالا EXPAND می کنیم :

برای h1 خواهیم داشت : در هر مرحله f(n)=g(n)+h(n) را چک می کنیم و نود با f(n) کمتر را انتخاب می کنیم که g(n) value یال ما می باشد و huristic h(n) valueهر نود ما میباشد :

1

S(0+7=7) ➔ A(3+9=12)

2 D(2+5=7) ➔ E(6+3=9)

4

3 B(3+4=7) ➔ C(5+2=7) ➔ G(9+0=9)

5 E(4+3=7) ➔ G(7+0=7) 6

ابتدا از node s شروع می کنیم دو انتخاب داریم با توجه به اینکه مقدار تابع در d کمتر است d را پیمایش می کنیم و A را رها می کنیم حالا از d دو انتخاب داریم که با توجه به اینکه مقدار تابع در b کمتر است b را پیمایش می کنیم و e را رها حالا از b دو انتخاب داریم c و e که هر دو در نهایت دارای f(n)=7 هستند با توجه به فرض مسئله ابتدا c را پیمایش می کنیم و سپس به g می رویم اما با رفتن e به g در نهایت مقدار تابع کمتری خواهیم داشت .

در نتیجه

Path : S→D→B→E→G

Expand : S→D→B→C→E→G

Highest priorty = lowest Cost

This priority queue is known as the open set or fringe. At each step of the algorithm, the node with the lowest f(x) value is removed from the queue, the f and g values of its neighbors are updated accordingly, and these neighbors are added to the queue.

STEP 1 : S(7)

STEP 2: D(7) , A(12)

STEP3 : B(7) , E(9)

STEP 4: C (7) , E(7)

STEP 5: G (9) , E(7)

STEP 6: G (7) STOP ALGORITHM

همانگونه که گفته شد ابتدا ما یک آرایه خالی برای نگه داری مقدار تابعی خود که دارای f(N) تابع است در نظر گرفته سپس فرانتیر و PRIORTY خود را با توجه به آن می چینیم که میتواند در هر مرحله اپدیت شود و به ارایه اضافه شود اگر نسبت به قبل F(N) کمتر داشته باشم و در غیر این صورت REJECT می شود و اگرF(N)مقدار بزرگتری از مقدار تابعی جدید داشته باشد یا مقدار جدید جایگزین می شود.

ب) همانگونه که میدانیم نحوه ی عملکرد و policy جست و جوی حریصانه به این صورت است :

Strategy: expand a node that you think is closest to a goal state

- Heuristic: estimate of distance to nearest goal for each state

A common case: - Best-first takes you straight to the (wrong) goal

Worst-case: like a badly-guided DFS

بنابراین نود ها با کمترین هیوریستیک را انتخاب می کنیم :

S→D→E→G

ابتدا از s شروع می کنیم چون هیوریستیک d کم تر است ان را انتخاب می کنیم سپس بین دو نودی که در فرانتیر ما هستند یعنی b , e چون هیوریستیک e کم تر است ان را انتخاب کرده و سپس به g می رویم.

مقایسه و معایب و مزایا :

همانطور که میبینیم در الگوریتم greedy یک نود کمتر از A* پیمایش کردیم اما در نهایت جواب A* بدلیل در نظر گرفتن هزینه مسیر ها بهینه تر خواهد بود بدلیل در نظرگرفتن g(n)+f(n) و در الگوریتم گریدی سریع تر به جواب میرسیم چون تعداد نود های پیمایش شده کمتر است.

مزایای greedy :

۱.پیاده سازی رویکرد حریصانه آسان است.

معمولا پیچیدگی زمانی کمتری دارند.

الگوریتم های حریص را می توان برای اهداف بهینه سازی یا یافتن نزدیک به بهینه سازی در صورت بروز مشکلات سخت مورد استفاده قرار داد.

معایب greedy:

راه حل بهینه محلی ممکن است همیشه در سطح global بهینه نباشد.

مزایای A* :

کامل و بهینه است.

این بهترین تکنیک در بین تکنیک های دیگر است. برای حل مسائل بسیار پیچیده استفاده می شود.

* Aاین بهینه کارآمد است، یعنی هیچ الگوریتم بهینه دیگری تضمین شده برای گسترش گره های کمتر از وجود ندارد.

این الگوریتم جستجوی بهینه از نظر اکتشافی است.

این یکی از بهترین تکنیک های جستجوی اکتشافی است.

برای حل مشکلات پیچیده جستجو استفاده می شود.

هیچ الگوریتم بهینه دیگری تضمین شده برای گسترش گره های کمتر از A* وجود ندارد

معایب A* :

این الگوریتم در صورتی کامل می شود که braching factorمحدود باشد و هر عمل دارای هزینه ثابت باشد.

استفاده می شود، h (n)* به شدت به دقت الگوریتم اکتشافی که برای محاسبه Aسرعت اجرای جستجوی بستگی دارد.

مشکلات پیچیدگی دارد.

مقایسه ی A* و الگوریتم greedy:

هر دو الگوریتم در دسته الگوریتم‌های «بهترین جستجوی اول» قرار می‌گیرند، که الگوریتم‌هایی هستند که می‌توانند هم از دانش کسب‌شده در حین کاوش در فضای جستجو، که با g(n) نشان داده می‌شود و هم از یک تابع اکتشافی که با h (n) نشان داده می‌شود، استفاده کنند.

هر یک از این الگوریتم‌های جستجو یک «evaluation function » را برای هر گره n در نمودار (یا فضای جستجو) تعریف می‌کند که با f(n) نشان داده می‌شود. این تابع ارزیابی برای تعیین اینکه کدام گره، در حین جستجو، ابتدا "expand" شده است، استفاده می شود، یعنی کدام گره ابتدا از "frontier"حذف می شود تا "visit" شود. فرزندانش به طور کلی، تفاوت بین الگوریتم های دسته «بهترین-اول» در تعریف تابع ارزیابی f(n) است.

در مورد الگوریتم BFS حریص، تابع ارزیابی f(n)=h(n) است، یعنی الگوریتم greedy BFS ابتدا گره ای را که فاصله تخمینی آن تا هدف کوچکترین است، گسترش می دهد. بنابراین، greedy BFS از "دانش گذشته"، یعنی g(n) استفاده نمی کند. از این رو معنای آن "greedy" است. به طور کلی، الگوریتم حریصانه BST کامل نیست، یعنی همیشه این خطر وجود دارد که مسیری را طی کنید که به هدف نمی رسد. در الگوریتم حریصانه BFS، تمام گره های حاشیه (یا حاشیه یا مرز) در حافظه نگهداری می شوند و گره هایی که قبلاً گسترش یافته اند نیازی به ذخیره در حافظه ندارند و بنابراین می توان آنها را دور انداخت. به طور کلی، BFS حریص نیز بهینه نیست، یعنی مسیر یافت شده ممکن است مسیر بهینه نباشد. به طور کلی، پیچیدگی زمانی O(bm) است، جایی که b ضریب انشعاب (حداکثر) و m حداکثر عمق درخت جستجو است. پیچیدگی فضا با تعداد گره ها در حاشیه و با طول مسیر یافت شده متناسب است.

در مورد الگوریتم A*، تابع ارزیابی f(n)=g(n)+h(n) است که h یک تابع اکتشافی قابل قبول است. به این واقعیت اشاره دارد که A* از یک تابع اکتشافی قابل قبول استفاده می کند، که اساساً به این معنی است که A* بهینه است، یعنی همیشه مسیر بهینه را بین گره شروع و گره پیدا می کند. گره هدف A*نیز کامل است (مگر اینکه تعداد بی نهایت گره برای کاوش در فضای جستجو وجود داشته باشد). پیچیدگی زمانی O(bm) است. با این حال، A* باید در حین جستجو، همه گره‌ها را در حافظه نگه دارد، نه فقط گره‌هایی که در حاشیه هستند، زیرا A*اساساً یک «جستجوی جامع» را انجام می‌دهد (که «آگاه است»، به این معنا که از یک تابع اکتشافی استفاده می‌کند.).

سوال دوم)

الف)

To make h_3 admissible, $h_3(B)$ has to be less than or equal to the actual optimal cost from B to goal G, which is the cost of path B-C-D-F-G, i.e. 12. The answer is $0 \leq h_3(B) \leq 12$

ب)

All the other nodes except node B satisfy the consistency conditions. The consistency conditions that do involve the state B are:

$$h(A) \leq c(A,B) + h(B) \qquad h(B) \leq c(B,A) + h(A)$$
$$h(C) \leq c(C,B) + h(B) \qquad h(B) \leq c(B,C) + h(C)$$
$$h(D) \leq c(D,B) + h(B) \qquad h(B) \leq c(B,D) + h(D)$$

Filling in the numbers shows this results in the condition: $9 \leq h_3(B) \leq 10$

ج)

The A^* search tree using heuristic h_3 is shown below. In order to make A^* graph search expand node A, then node C, then node B, suppose $h_3(B) = x$, we need:

$$1 + x > 13 \quad , \quad 5 + x < 14 \quad \text{(expand } B'\text{)} \quad \text{or} \quad 1 + x < 14 \quad \text{(expand } B\text{)}$$