وزارت علوم، تحقیقات و فناوری
دانشگاه تحصیلات تکمیلی علوم پایه
گاوازنگ، زنجان

پازل سوکوبان

دکتر پروین رزاقی

طراحی سیستم‌های هوشمند

یاسمین سادات میرزابابا

بهار ۹۷

هدف از انجام پروژه

حل کردن پازل سوکوبان با استفاده از اگوریتم‌هایی که در حوزه یادگیری تقویتی ارائه شده‌اند.

یکی از الگوریتم‌های این حوزه Q Learning ا ست که هدف آن یافتن سیا ستی اســـت که بتواند در هر مرحله حرکت؛ حرکتی را انتخاب کند که بهترین حرکت ممکن بین سایر حرکت‌ها باشد.

سیا ست بهینه از دو روش value Iteration و policy iteration به د ست میاید که در ادامه توضیح داده خواهند شد.

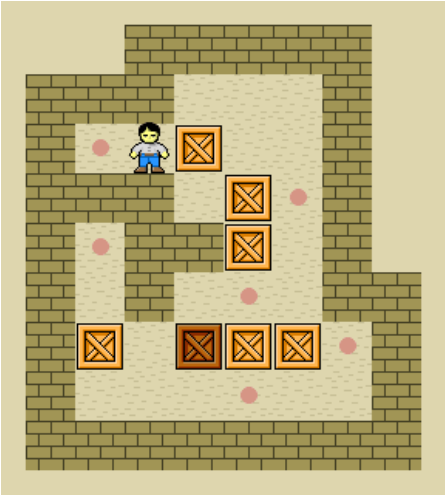
پازل سوکوبان

نوعی پازل جابجایی اســـت که در آن بازیکن جعبه یا صـــندوق‌ها را در یک برای قرار دادن آن‌ها در محل نگه‌داری‌شان در انبار هل می‌دهد. این پازل در سال ۱۹۸۱ توسط ایمابایاشی معرفی شده است.

قوانین بازی:

این بازی بر روی یک برد از مربع ها و بعضی از مربع ها شامل جعبه ها، و بعضی از مربع ها به عنوان مکان های ذخیره سازی مشخص شده اند.

بازیکن می‌تواند به صـــورت افقی یا عمودی بر روی مربع های خالی حرکت کند اما نه از طریق دیوار یا جعبه. این پازل زمانی حل می شـــود که تمام جعبه ها در مکان های ذخیره سازی قرار داشته باشند.



یادگیری تقویتی

یکی از حوزه‌های یادگیری ماشین است و بر رفتارهایی تمرکز دارد که ماشین باید برای بیشینه کردن پادا شش انجام دهد. این مسئله، با توجه به گستردگی‌اش، در زمینه‌های گوناگونی بررسی می‌شود .

یک مدل ابتدایی یادگیری تقویتی از:

۱- یک مجموعه از حالات مختلف مسئله

۲- یک مجموعه از تصمیمات قابل اتخاذ

۳- قوانینی برای گذار از حالات مختلف به یکدیگر

۴- قوانینی برای میزان پاداش به ازای هر تغییر وضعیت

۵- قوانینی برای توصیف آنچه که ماشین می‌تواند مشاهده کند

Value Iteration

یک روش محا سبه سیا ست مطلوب در محیط‌های MDP ا ست. این الگوریتم به صورت عقب‌گرد عمل می‌کند و از آخرین حالت شروع به کار میکند.

در ابتدا مقدارvalue برای تمامی حالتهایی که ممکن اســـت زمین بازی داشـــته باشـــد(جایگاه عامل و جعبه) صـــفر درنظر گرفته می‌شـــود ســـپس فرایندی اجرا می‌شـــود که در آن مقدار value برای تمام حالتها به روز شـــده و کامل میشـــود و تا زمانی که به همگرایی نرســـد توقف نمیکند.همگرایی در اینجا یعنی مقدارvalue برای تمام حالتها پس از چند بار اجرا شدن فرایند با مقدارها در حالت قبل تفاوت چندانی نداشته باشد.

$$R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V(s')$$

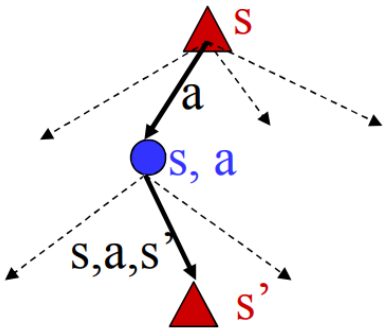
عبارت بالا همان فرایندی است طی اجرای الگوریتم انجام میشود.

پارامترهای عبارت بالا، پاداش حالتی که اکنون عامل در آن قرار دارد اســـت که آن را از محیط در یا فت کرده و اجرای همین فرا ید برای حالتهای بعدی که عا مل میتواند در یک مرحله آنها را ایجاد کند(در این مسئله بین ۰ تا ۴ میتواند باشد)

در این الگوریتم به دلیل عملکرد عقبگرد، در هربار اجرای فرایند ذکر شـــده تنها حالتهایی که یک حالت قبل از حالت نهایی(یعنی حالتی که در آن عامل، جعبه را به خانه هدف برده) هستند مقدار میگیرند.

پیچیدگی این الگوریتم $O(|A||S|^2)$ برای هربار اجرای فرایند ذکر شده است.

A تعداد حرکتهای مجاز در هر مرحله اســت (در این مســئله حرکتهای مجاز: بالا، پایین، چپ و راســت اســت که در این صــورت A = ۴) و S تعداد حالتهایی که در بازی ایجاد میشود.



پیچیدگی مکانی این الگوریتم نیز به مقدار تعداد حالتهای تولید شـــده بتگی دارد. چرا که باید تمامی آنها نگهداری شود.

Policy Iteration

یک روش محاسبه سیاست مطلوب در محیط‌های MDP است. این الگوریتم به صورت عقب‌گرد عمل می‌کند و از آخرین حالت شروع به کار میکند.

در این الگوریتم برای هر حالت گفته میشود حرکت بعد، چه حرکتی باید باشد. درواقع ساست حرکت عامل را بیان میکند.

در ابتدا به صورت اتفاقی از بین حرکتهای ممکن یکی را به هر حالت ممکن در بازی اختصاص میدهیم. سپس با آن حرکت Value Iteration را انجام میدهیم و مقدار Val ue را به روز رسانی میکنیم. اگر حرکت به دست آمده برای آن حالت در تابع value با حرکت اتفاقی انتخاب شده در ابتدا مشابه نبود، جدول policy به روزرسانی میشود و مجدد Value Iteration انجام میشود. این روند تا همگرا شدن ادامه میابد.

$$V_\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V_\pi(s')$$

$$\pi'(s) := \arg\max_a \left(R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V_\pi(s') \right)$$

پیچیدگی زمانی این الگوریتم $O(|S|^3 + |A| \cdot |S|^2)$ و پیچیدگی مکانی آن نیز ه تعداد حالتهای یجاد شده در کل طول بازی است.

مقایسه دو الگوریتم

الگوریتم value iteration چون پیچیدگی زمانی کمتری دارد برای مسائلی با محیطهای بزرگ بهتر از policy iteration است.

در الگوریتم policy iteration در هربار اجرا تمام حالتها مقدار جدید میگیرند و نه تنها حالتهای یک مرحله قبل.