به نام خدا

پروژه هوش مصنوعی

مهرداد قصابی

۹۷۳۶۱۳۰۶۰

پیمان اخلاقی

بنده تعهد می نمایم که پروژه تحویل داده شده نتیجه کار خویش بوده
و در هیچ یک از بخش های پروژه از کار کسی کپی برداری نشده است
در صورتی که مشخص گردد این پروژه کار بنده نبوده است طبق ضوابط اموزشی
با من برخورد شده و حق اعتراضی نخواهم داشت

پیشگفتار

هدف از انجام این پروژه نگاشتن یک عامل معقول برای بازی پکمن است که مستندات قوانین بازی پیوست گشته است

انجام این مهم به چندین فاز تقسیم شده که در ادامه هر فاز به تفضیل توضیح داده میشود

فاز نخست

جمع اوری الماس ها

مسیریابی

از آنجایی که برای جمع اوری الماس ها ما نیازمند پیدا کردن مسیر از یک خانه به خانه دیگر هستیم اولین چالش ما در این فاز مسیریابی است

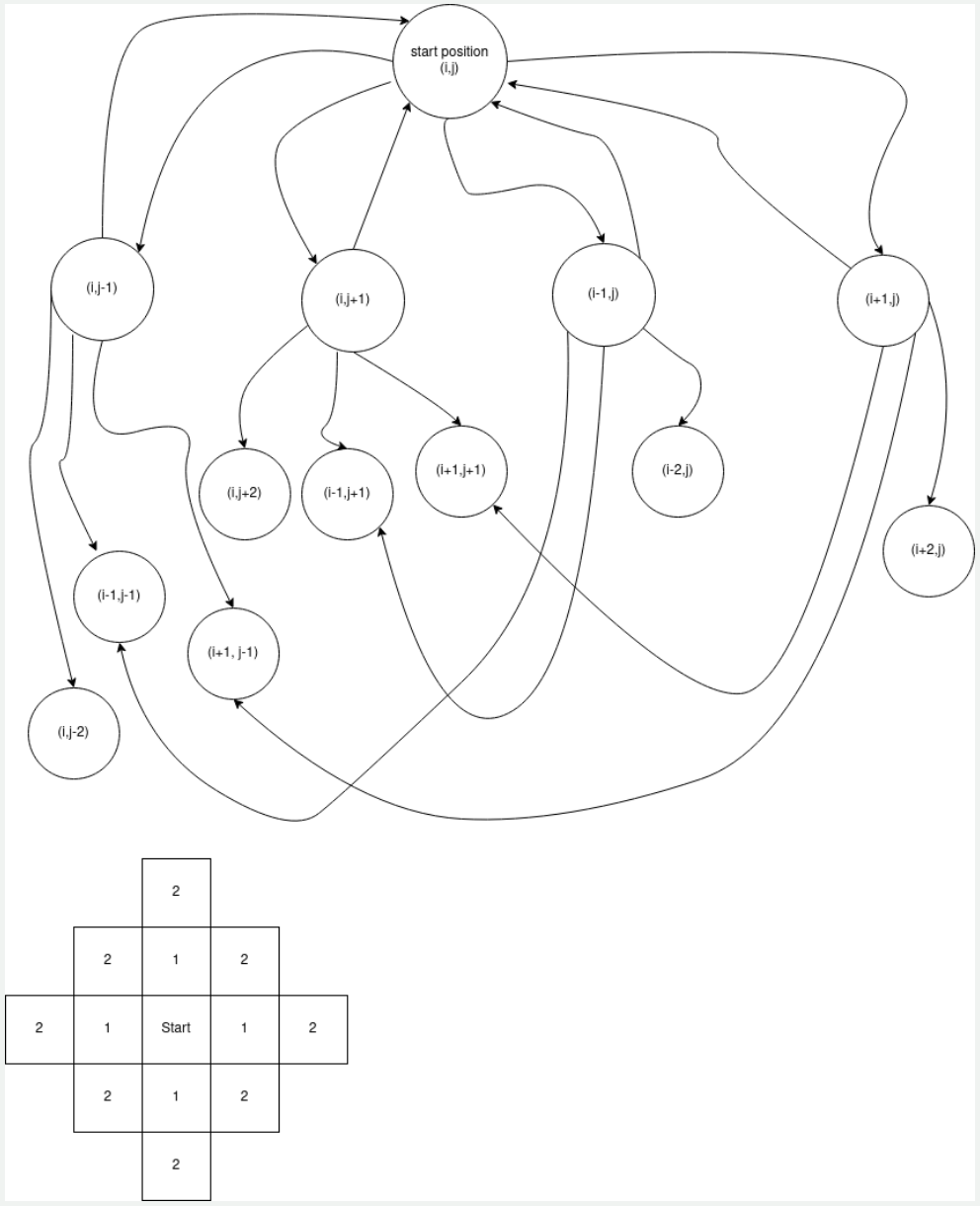
به گونه ای که پس از اولویت بندی جمع اوری الماس های موجود وظیفه عامل حرکت از خانه کنونی به خانه الماسی است که بیشترین اولویت را دارا است

همانطور که در درخت جستجوی صفحه بعد مشاهده خواهید کرد امکان ایجاد حلقه بینهایت در جستجو ما وجود دارد

برای رفع این مشکل با تعریف یک مجموعه پیشرو اجازه جستجو در خانه های تکراری را

به عامل نمیدهیم و از ایجاد حلقه جلوگیری میکنیم

فضای حالت جستجو مسیر مناسب در ژرفای دو



مدیریت زمان

زمان مناسب یا مناسب ترین مسیر ؟

شاید پاسخ به پرسش بالا مهم ترین چالش این فاز باشد چون بسته به محدودیت زمانی که عامل دارد ممکن است استفاده از یکی الگوریتم های مسیریابی به دیگری ترجیح داده شود

اگر عامل بهترین جواب ممکن را ترجیح دهد و زمان پاسخ برایش اهمیت نداشته باشد

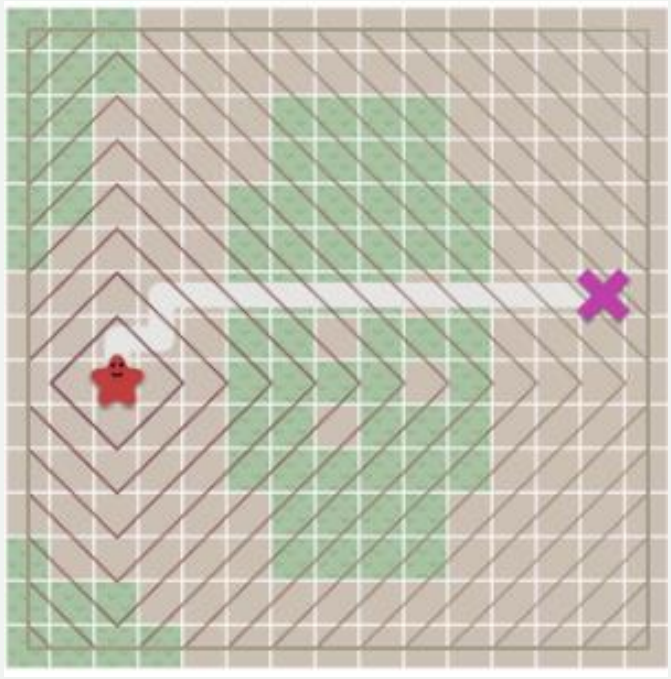
مجموعه پیشرو در تمامی جهات گراف حالت به صورت مساوی پیش رفته تا بهترین جواب ممکن را پیدا کند

اما اگر محدودیت زمانی به گونه ای باشد که امکان جستجو در تمامی جهات امکان پذیر نباشد

بایستی در جهت های امیدوار کننده تر پیشروی کرد

در ادامه به تعریف امیدوار کنندگی و شیوه های مسیر یابی پیاده سازی شده خواهیم پرداخت

مسیر یابی عرض اول



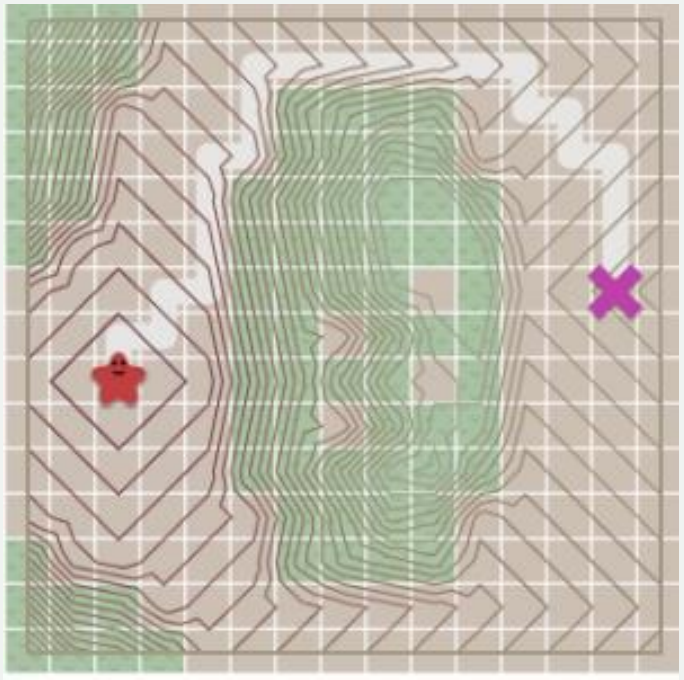
در این الگوریتم حریصانه ایده کلیدی این است که مجموعه پیشرو بایستی به طور برابر در تمامی جهات گسترش پیدا کند.

این گسترش در تمامی جهات تا زمانی ادامه خواهد داشت که خانه هدف پیدا گردد

این الگوریتم تنها در مسیر های امیدوار کننده برای جواب کاوش میکند بنابراین شاید بهترین جواب پیدا نشود .

راهبرد ما اینگونه است که از این الگوریتم وقتی استفاده کنیم که زمان اجرای مناسبی در اختیار نداریم و پیدا کردن یک جواب مناسب در زمان کوتاه را به بهترین جواب ترجیح میدهیم

الگوریتم ای استار



الگوریتم دایجکسترا برخلاف جستجوی عرض اول در جهت هایی که امیدوار کننده نیستند هم کاوش میکند این ویژگی باعث پیدا شدن بهترین جواب ممکنه میگردد

الگوریتم ای استار نیز بسیار شبیه به الگوریتم دایجکسترا است ولی برای بهینه سازی زمانی از تابع اکتشافی در ان استفاده مبگردد

ولی از آنجایی که الگوریتم دایجکسترا از تابع اکتشافی استفاده نمیکند ترجیح داده شد که برای وقتی عامل زمان مناسب در اختیار دارد الگوریتم ای استار برگزیده شود

تابع اکتشافی و امیدوارکنندگی

همانطور که پیاده سازی الگوریتم ها یاد شد الگوریتم حریصانه عرض اول و ای استار

جستجو در مسیر های امیدوار کننده را بر جستجو در تمامی مسیر ها ترجیح میدهند

طبق تعریف امیدوار کنندگی مسیری امیدوار کننده تر است که بتوان تخمین زد از آن مسیر با هزینه کمتری نسبت به مسیر های دیگر به مقصد میرسیم

البته که خود این تابع که ورودی های آن مبدا و مقصد هستند بایستی دارای پیچیدگی زمانی ثابتی باشد که عامل با هربار صدا زدن آن هزینه ثابتی بپردازد

از انجایی که فاصله منهتن بین مقصد و مبدا با هزینه حرکت دارای رابطه خطی است

لذا فاصله منهتن به عنوان تابع اکتشافی انتخاب گشته است

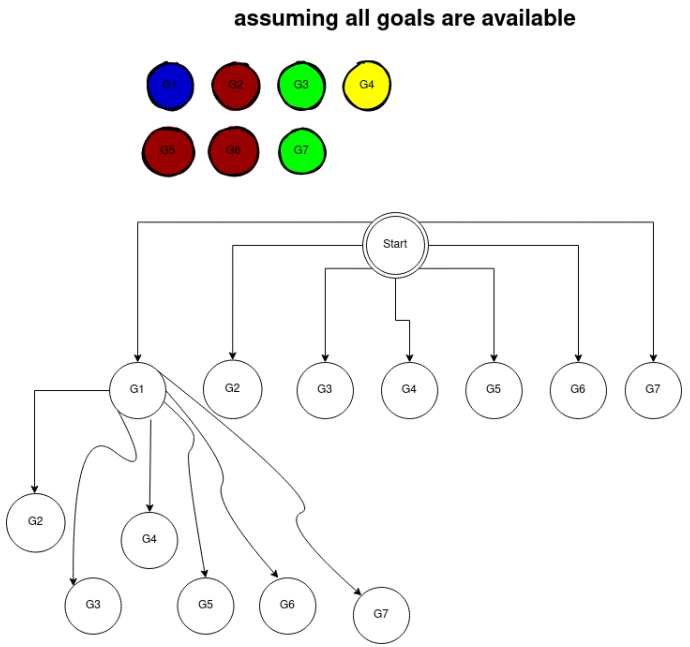
تابع انتخاب

پس از نگاشتن تابع مسیر مناسب حال نوبت به این میرسد که عامل هدف های موجود را اولویت بندی کند و طبق این اولویت بندی اقدام به جمع آوری تمامی الماس ها کند

در این مرتب سازی الماس ها خیلی از عوامل میتوانند تاثیر گذار باشند مانند فاصله عامل با الماس ارزشمندی الماس و ترس عامل از عامل دیگر یا حتی میل همکاری عامل (دو مورد آخر نقشی در این فاز ندارند چون فاز نخست تک عاملی است)

بنابراین نوشتن یک تابع انتخاب درست بسیار ضروری است

در این فاز این تابع نسبتی از امتیاز و فاصله عامل نوشته شده است



درخت جستجو الماس ها
یک درخت پویا

کنش سیاه چاله

همانطور که در توضیحات بازی به ان اشاره گشته است با رفتن داخل خانه سیاه چاله و انتخاب کنش آن عامل به صورت تصادفی در یکی دیگر از سیاهچاله پدیدار میگردد

لذا عامل ما بایستی بین ادامه دادن به جستجو و کنش سیاه چاله بتواند انتخاب کند

ترجیح دادن کنش سیاه چاله دو زمان اتفاق میفتد

یک زمانی که تمامی الماس های باقی مانده دارای هزینه بینهایت هستند

دو زمانی که مرکز سنگینی سیاه چاله ها(یک تخمین از مکان احتمالی عامل پس از کنش سیاه چاله) دارای موقعیت بهتری برای جمع اوری الماس ها باشد

فاز دوم

جستجوی خصمانه

ترس عامل

تعریف ترس یک عامل از عامل دیگر در این پروژه کمتر بودن امتیاز عامل اول از عامل دوم است

منطقا اگر یک عامل ترسیده باشد سعی بر فرار (دور شدن) از عامل دیگر دارد و اگر نترسیده باشد سعی بر ضربه زدن (نزدیک شدن) بر آن عامل دارد

طبیعی است که وقتی الماسی در بازی نمانده تنها هدف عامل ضربه زدن (ضربه نخوردن) از عامل دیگر است

ولی وقتی الماس ها هنوز در بازی حضور دارند بایستی تابع انتخاب را بر حسب ترس عامل نیز تغییر دهیم

در صفحه بعد درخت تصمیم عامل برای فرار را مشاهده میکنید

طبیعی است که درخت تصمیم برای حمله کاملا بر عکس این درخت است

الگوریتم تعقیب

برای دنبال کردن یا فرار از رقیب (بسته به ترس عامل) بایستی از الگوریتم تعقیب پیروی کنیم

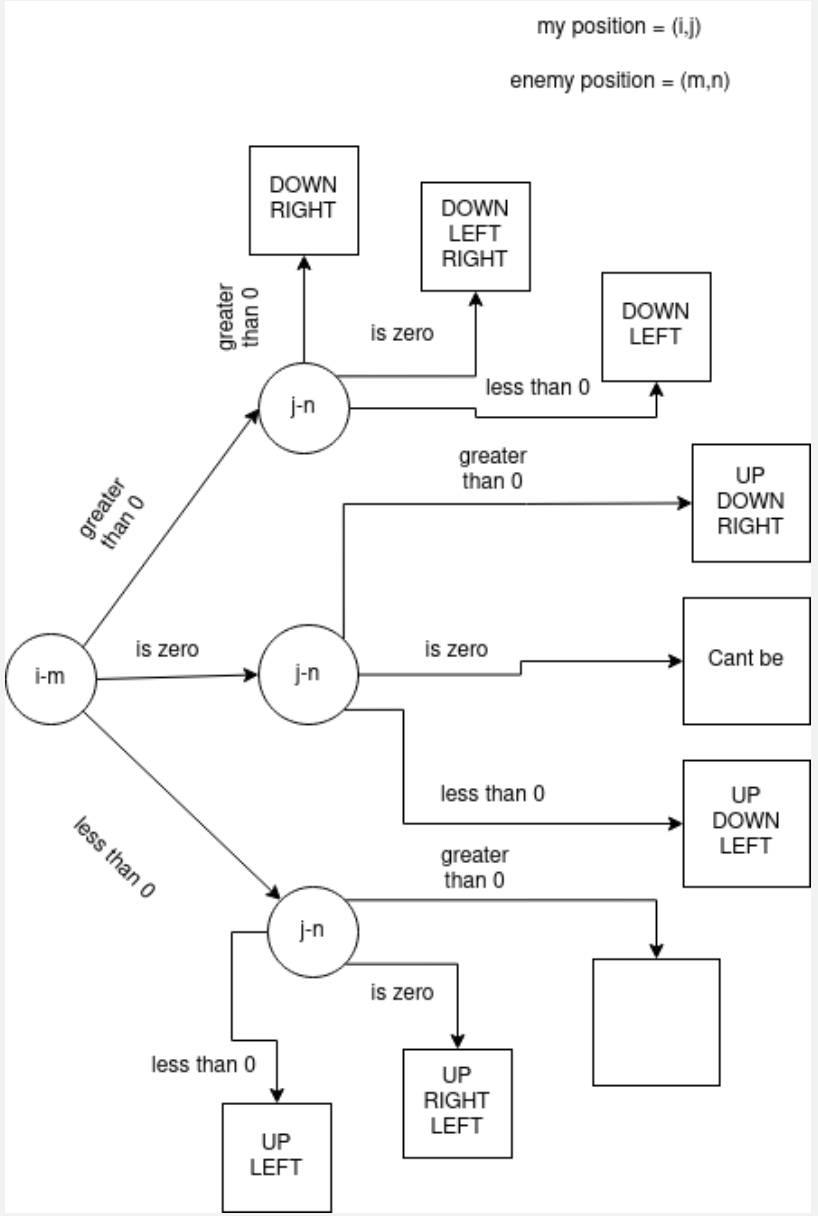
این الگوریتم بر اساس فاصله دو عامل عمل میکند (در این پروژه فاصله منهتن)

و در هنگام فرار سعی میکند فاصله خود را با رقیب زیاد کرده و در هنگام تعقیب

سعی میکند فاصله خود را با رقیب کمتر کند

عامل با توجه به مکان عامل و رقیب آن و درخت تصمیم که نسبت به ترس عامل تغییر میکند کنش مناسب را انتخاب میکند

درخت تصمیم
تعقیب
در هنگام ترس



درخت مینی ماکس

با در نظر گرفتن (امتیاز عامل منهای امتیاز عامل دشمن) به عنوان مقدار الگوریتم مینی ماکس میتوان درخت مینی ماکس را به گونه ای رسم کرد که عامل سعی در بیشینه سازی متغییر یاد شده دارد و عامل دشمن سعی در کمینه سازی آن

با در نظر گرفتن محدودیت زمان تا عمق مشخصی در این درخت پیش رفته و مناسب ترین شاخه را انتخاب میکنیم

با توجه به اینکه عامل دشمن از مکان تله های ما آگاه نیست با توجه به فاصله آن

کنش تله را انتخاب میکنیم

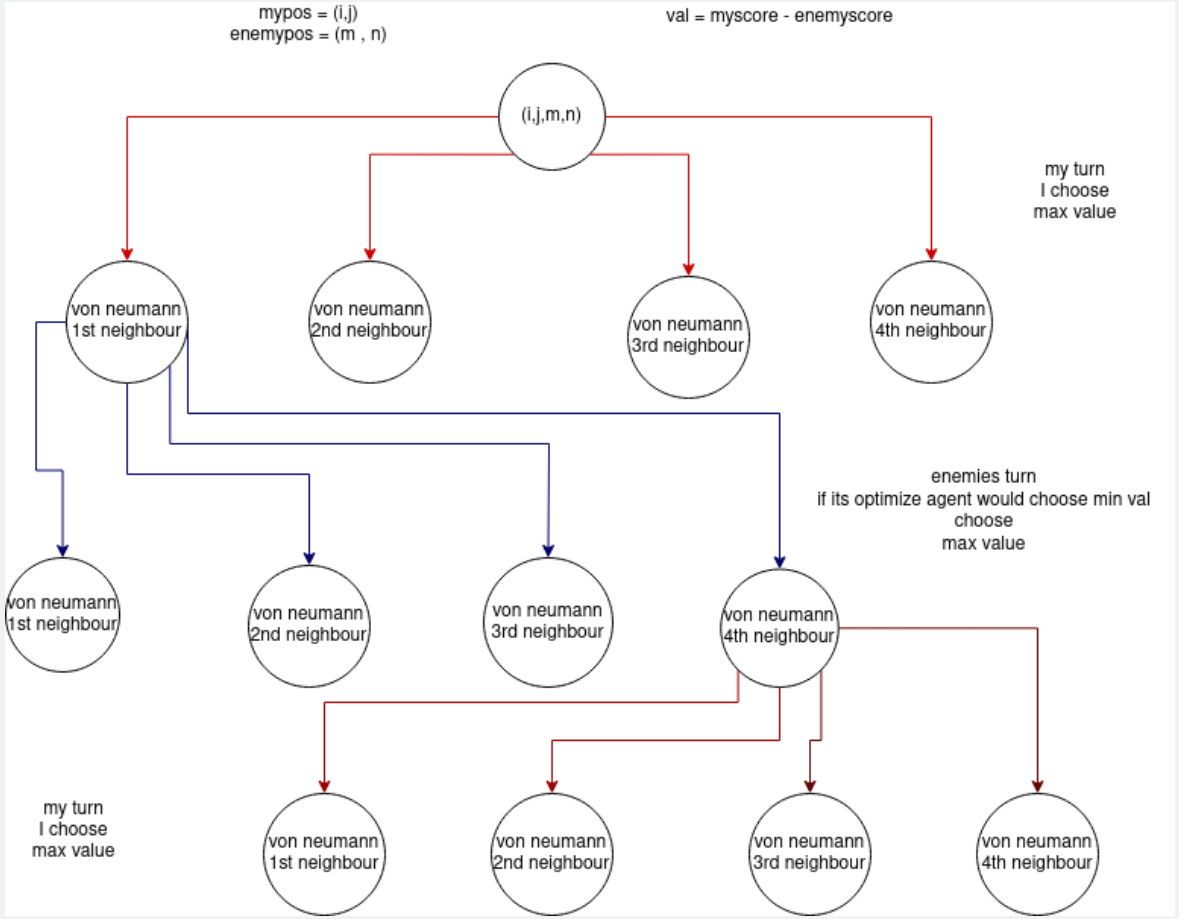
مدیریت زمان در درخت مینی ماکس

برای مدیریت زمان عامل همانطور که پیشتر به آن اشاره شد تنها مجاز به پیش روی تا عمق خاصی از درخت مینی ماکس را داریم تا زمان انجام کنش بیشتر از بیشینه زمان نشود

برای انتخاب این عمق راه های متفاوتی وجود دارد که ساده ترین آن انتخاب یک عدد ثابت مانند پنج است

در پروژه انجام شده این عدد با اندازه نقشه رابطه مستقیم دارد و ثابت نیست

درخت مینی ماکس



فاز سوم

یادگیری ماشین

کیو لرنینگ

کیو لرنینگ یک شیوه تقویتی یادگیری ماشین است که از یک ارایه دو بعدی به روزرسانی شونده برای یادگیری استفاده میکند که هر خانه در این ارایه نماینگر امتیاز انتخاب یک کنش در یک مرحله خاص است

در این پروژه با توجه به مکان عامل و دشمن آن و تعداد کنش های موجود

پس از هربار اجرا گشتن یک نقشه خاص بازی سعی گشت که برای هر نقشه یک مدل تمرین شده (که یک ارایه است) ایجاد گردد و از ان در مراحل یاد شده استفاده گردد

شیوه تمرین هم صرفا با نوشتن و خواندن در فایل انجام گشته است

نرخ پوسیدگی و تمرین کردن عامل

همانطور که پیشتر در معادله بلمن توضیح داده شد یادگیری تقویتی به شیوه کیو دارای یک مقدار به نام نرخ پوسیدگی است هرچه این نرخ بیشتر باشد عامل در تعداد دور کمتری به مدل تمرین شده دست میابد

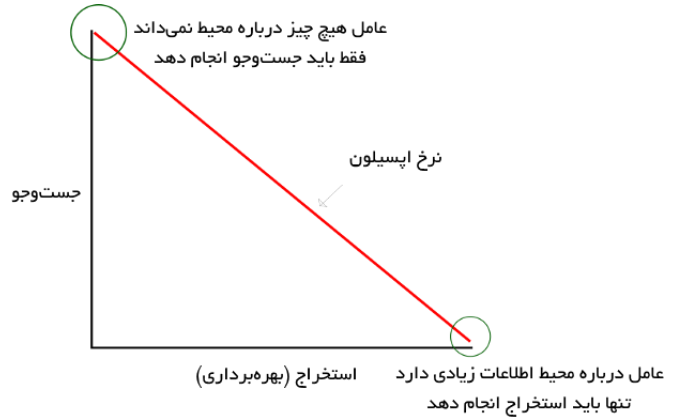
هر بار که عامل اجرا میشود یک عدد تصادفی بین صفر و یک انتخاب میشود و نرخ اپسیلون به صورت زیر تعیین میشود

```
epsilon = min_epsilon + (max_epsilon - min_epsilon) * math.exp(-decay_rate * episode)
```

شیوه استخراج

در هر نوبت اجرای عامل مورد نظر یک عدد تصادفی بین صفر و یک در نظر گرفته میشود و با نرخ اپسیلون مقایسه میشود اگر عدد تصادفی یاد شده کوچکتر از نرخ اپسیلون بود عامل هنوز اطلاعات زیادی درباره محیط ندارد و باید یک عمل تصادفی را برگزیند و طبق پاداش ان جدول را به روزرسانی کند

در غیر این صورت عامل اطلاعات کافی دارد و طبق ان میتواند کنش مناسب را انتخاب کند



معادله بلمن

مقادیر موجود در ارایه یادگرفته شده کیو تیبل بایستی توسط معادله بلمن به روزرسانی شوند

این معادله بدین گونه است که هر مرتبه پس از اجرا پاداش در نظر گرفته شده شده را به خانه مورد نظر در جدول کیو اضافه میکند

منابع

https://www.redblobgames.com/pathfinding/a-star/introduction.html

https://blog.faradars.org/reinforcement-learning-and-q-learning/