دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

درس هوش مصنوعی و سیستم‌های خبره

گزارش کار فاز اول و دوم پروژه

دانشجویان:

رضا براتی

۹۷۳۶۱۳۰۱۰

سروش ذوالفقاری

۹۷۳۶۱۳۰۲۴

تعهدنامه اخلاقی:

ما (رضا براتی و سروش ذوالفقاری) تعهد مینماییم که پروژه تحویل داده شده نتیجه کار ما بوده و در هیچ یک از بخشهای انجام شده از کار دیگران کپی برداری نشده است. در صورتی که مشخص شود که این پروژه کار ما نبوده است، طبق ضوابط آموزشی با ما برخورد شده و حق اعتراض نخواهیم داشت.

فاز اول

مرحله اول: پیاده سازی الگوریتم مسیریابی

فضای حالت:

- گره: به هر خانه از جدول بازی یک گره گفته می شود
- حالت: به انتخاب هایی که در هر گره وجود دارد، که شامل گره های همسایه آن (درواقع چهارخانه ای که در جهات اصلی با گره همسایه اند) و کنش تله و همچنین کنش تله پورت حالت گفته می شود.

در زمین بازی الماس ها و موانعی قرار دارند. هدف عامل پیدا کردن بهترین مسیر (بهترین مسیر درواقع کوتاه ترین مسیر بدون مانع می‌باشد) به سمت الماسی است که بیشترین امتیاز را با توجه به مسافت پیمایش به عامل بدهد. پس درواقع در هر نوبت عامل نیاز به انجام دو عمل میباشد: ۱- انتخاب بهترین الماس ۲- انتخاب بهترین مسیر برای رسیدن به آن. اگر هر الماس را یک قله و هر دیوار را یک دره تصور کنیم میتوانیم با پیمایش به سمت نقاطی با ارتفاع بیشتر به کسب امتیاز بپردازیم.

به صورت فرضی هر خانه از صفحه درواقع یک گره به حساب می آید و عامل می تواند در چند حالت قرار بگیرد:

۱- در قله قرار بگیرد : به این معنا که هیچ خانه ای با ارزش تر برای انتخاب وجود نداشته باشد

۲- در خارج از قله قرار داشته باشد: به این معنا که خانه ای در چهار جهت اصلی بالا، پایین، چپ و راست وجود دارد که ارزش بیشتری نسبت به وضعیت فعالی دارد و عامل باید بر روی آن گره قرار بگیرد.

۳- سیاه چاله: درصورتی که طبق الگوریتم بازی (در ادامه به صورت دقیق توضیح داده شده است) با ارزش ترین حالت اطراف عامل سیاه چاله باشد، عامل به سمت آن حرکت کرده و تلپورت میکند.

درواقع ما برای مسیریابی در بازی نیاز به تبدیل نقشه به یک صفحه با دیدگاه تپه نوردی هستیم.

پیاده سازی این الگوریتم خود شامل سه قسمت میباشد:

قسمت اول: مقداردهی اولیه خانه های نقشه

در قسمت اول با استخراج محل دیوارها و الماس ها و سیاه چاله ها، یک آرایه از نقشه ساخته و مقداردهی اولیه خانه های آرایه را به نحو زیر انجام میدهیم:

خانه های خالی: برابر صفر درنظر گرفته میشوند

خانه های شامل الماس: درصورتیکه ۱) عامل حداقل امتیاز مورد نیاز برای به دست آوردن الماس را داشته باشد و ۲) عامل کمتر از حداکثر الماس ممکن برای آن نوع الماس را خورده باشد، برابر امتیاز آن الماس در نظر گرفته میشود. در غیر اینصورت، تا زمانیکه عامل شروط لازم برای به خوردن آن نوع الماس را به دست آورد، با آن خانه همانند یک خانه معمولی برخورد شده و امتیاز آن برابر صفر درنظر گرفته میشود

خانه های دیوار: برابر -۱ در نظر گرفته میشوند

سیاهچاله ها: برابر صفر درنظر گرفته میشوند

همچنین مکان عامل و سیاهچاله ها در آرایه هایی مجزا ذخیره میشوند.

(پیاده سازی این قسمت در فایل map_read.py انجام شده است)

قسمت دوم: پیاده سازی تابع هدف

در این قسمت با استفاده از آرایه ساخته شده در قسمت قبل، تابع هدف خود را به نحو زیر برروی آرایه اعمال میکنیم:

اگر بخواهیم تابع هدف را تابعی از امتیاز هر خانه از آرایه ساخته شده در نظر بگیریم، مشکلی که بوجود می آید، امکان وجود شانه(shoulder) در تابع است که مسیریابی را با مشکل مواجه میکند.

راه حلی که برای حل این مشکل ارائه کردیم اعمال شرط زیر بود:

امتیاز هر خانه(به جز خانه های دیوار) باید حداقل نصف ماکزیمم امتیاز ۴ خانه اطراف خود(بالا، پایین، چپ، راست) باشد.

این شرط تضمین میکند که اگر الماسی(تپه) در محیط بازی وجود داشته باشد که عامل بتواند آنرا بدست آورد، حتما مسیری بین عامل و الماس توسط تابع هدف ایجاد خواهد شد.

برای اعمال این شرط، هربار آرایه را پیمایش میکنیم و درصورتیکه خانه ای پیدا شود که از نصف ماکزیمم خانه های ۴ طرف خود کمتر باش، آنرا برابر نصف ماکزیمم خانه های ۴ طرفش قرار میدهیم. این کار را تا جایی انجام میدهیم که هیچ خانه باقی نماند که شرط مذکور در آن رعایت نشده باشد

نحوه برخورد با سیاهچاله ها

با توجه به اینکه قبلا مکان خانه های سیاهچاله را ذخیره کردیم، در این مرحله یکبار امتیاز هر خانه سیاهچاله را برابر با میانگین امتیاز تمام سیاهچاله ها قرار میدهیم و سپس دوباره الگوریتم فوق را جهت اعمال شرط مذکور اجرا میکنیم. دلیل این کار این است که اگر عامل در شرایطی مشابه با شرایط نقشه map1_2 قرار گرفت و تنها گزینه آن تلپورت کردن از سیاهچاله بود یا در شرایط دیگری که بهترین گزینه حرکت به سمت سیاهچاله است، عامل بتواند این حرکت را شناسایی کند. در نهایت گزینه تلپورت کردن در خانه های سیاهچاله به گزینه های کنش عامل اضافه میشود و امتیاز آن برابر میانگین جدید امتیاز خانه های سیاهچاله درنظر گرفته میشود.

از آنجایی که پس از تلپورت کردن عامل به یک سیاهچاله تصادفی میرود، پس باید امید ریاضی امتیاز پس از تلپورت را به عنوان امتیاز کنش تلپورت انتخاب کرد. دلیل استفاده از میانگین برای تعیین امتیاز کنش تلپورت همین است.

(الگوریتم های شامل این قسمت در فایل objective_function.py پیاده سازی شده اند.)

فضای حالت الگوریتم مسیریابی

با توجه به توضیحات ارائه شده، فضای حالت این الگوریتم برابر امتیاز هر خانه است که در محدوده -۱(کمترین امتیاز و مربوط به دیوار) و ۷۵(بیشترین امتیاز و مربوط به الماس آبی) میباشد.

مرحله دوم: پیاده سازی الگوریتم راهبرد حل مسئله

برای حل مسئله، عامل باید به سمت بیشترین امتیاز 4 خانه اطراف خود برود. در صورتیکه عامل در خانه سیاهچاله باشد، کنش تلپورت هم به ۴ گزینه خانه های اطراف اضافه شده و عامل باید بیشترین مقدار را از بین این ۵ گزینه انتخاب کرده و کنش مربوط به آن گزینه را اعمال کند (در واقع عامل باید به سمت تپه حرکت کند).

(الگوریتم این قسمت در فایل algorithm.py پیاده سازی شده است)

فضای حالت الگوریتم راهبرد

فضای حالت این الگوریتم، تمام خانه های نقشه هستند.

محدودیت زمانی:

طبق اندازه گیری که تیم ما از الگوریتم پیاده سازی شده بدست آورد، در بد ترین حالت (زمانی که اندازه جدول بازی ۲۰x۲۰ باشد و کل خانه های بازی نیز دارای الماس باشد) پیچیدگی زمانی آن برابر با ۱۶۱۰۰۰ عمل در هر نوبت می‌باشد. طبق اندازی گیری که ما بر روی چند سیستم انجام دادیم تقریبا زبان پایتون ۱۰۰۰۰۰۰۰ عمل را در یک ثانیه انجام میدهد (توان هر سیستم در نتیجه تاثیر گذار است). بنابر این این الگوریتم به شکل بهینه ای در محدوده زمانی ۱ ثانیه عمل میکند.

استراتژی که تیم ما برای کنترل محدودیت زمانی در الگوریتم تپه نوردی انتخاب کرده است محدود کردن طول زمین بازی می‌باشد. به این صورت که اگر نیاز به انجام عملیات کمتری در واحد زمان بود محدودی بررسی زمین بازی را به NxN کاهش میدهیم و در واقع محیط بازی برای عامل از مشاهده پذیر کامل به مشاهده پذیر جزئی تغییر میابد.

نقش اعضا در پروژه:

هر دو عضو گروه تقریبا وظایف یکسانی داشتند. از آنجایی که پیشبرد پروژه بیشتر بصورت تعاملی بوده، هر دو نفر امر تحقیقات و تصمیم گیری را انجام میدادیم. وظیفه کدنویسی هم بصورت pair-coding انجام میشد.

فاز دوم

۱) تصمیم گیری عامل به هفت حالت تقسیم میشود که روند و چگونگی تصمیم گیری در هر کدام را به اختصار توضیح میدهیم:

مفاهیم اولیه:

- گره: به هر خانه از جدول بازی یک گره گفته می شود
- حالت: به انتخاب هایی که در هر گره وجود دارد، که شامل گره های همسایه آن (درواقع چهارخانه ای که در جهات اصلی با گره همسایه اند) و کنش تله و همچنین کنش تله پورت حالت گفته می شود.
- ارزش: مقدار عددی که تابع ارزیابی تخمینی به هر حالت میدهد ارزش آن حالت می باشد.

تمام روند تصمیم گیری عامل بر اساس الگوریتم تپه نوردی که در فاز اول به صورت کامل توضیح داده شده است انجام میشود. و در این فرایند از تابع ارزیابی تخمینی استفاده شده است که اندکی با مرحله اول متفاوت است و براساس شرایط تعریفی مسئله کنش های عامل را ارزش گذاری میکند.

فرایند ارزش گذاری حالت های عامل به شرح زیر می‌باشد:

- حرکت در چهار جهت اصلی: تابع ارزیابی تخمینی به ازای هر الماس مقدار آن را در خانه وی قرار میدهد و در انتها ارزش خانه هایی که الماس در آن ها قرار داده شده است را با ضریب ۱/۲ در گره هایی که در چهار جهت اصلی گره فعلی هستند پخش میکند و به صورت بازگشتی این کار برای تمام زیر گره های موجود در بازی صورت میگیرد.(در صورتی که در طی این فرایند یک گره چندین بار ارزش گذاری شود، بالاترین ارزش برای آن در نظر گرفته می شود) به این شکل ارزش مربوط حرکت در چهار جهت اصلی انجام میگیرد.

- حرکت تلپورت: ارزش هر کنش(حالت) تلپورت برابر با میانگین ارزش سیاه چاله ها است.(ارزش هر سیاه چاله مشابه سایر گره های بازی که در قسمت قبل بیان شد مقدار دهی می شود)
- حرکت تله: این کنش حالت خاص به شمار می آید در زمانی که برقرار باشد در بالاترین اولویت عامل قرار خواهد گرفت. زمانی این شرط برقرار است که عامل حریف در حال حرکت به سمت عامل باشد و همچنین در ردیف و یا ستون مشابه قرار گرفته باشند.
- حرکت ضربه : زمانی که عامل حریف درحال نزدیک شدن به عامل باشد و همچنین عامل از نظر امتیاز نسبت به عامل حریف برتری داشته باشد تابع ارزشیابی تخمینی ارزشی معادل با ۲۰ امتیاز را برای عامل حریف قائل میشود و مشابه قبل گره های بازی را ارزش گذاری میکند.

۲) راهبرد عامل:

در هر نوبت پس از این که تابع ارزش گذاری تخمینی تمام فضای حالت بازی را ارزش گذاری کرد عامل بر اساس الگوریتم تپه نورد به سمت تپه حرکت می کند. درواقع با ارزش ترین حالت (کنش) موجود بین حالت ها را انتخاب میکند.

۳) محدودیت زمانی:

طبق اندازه گیری که تیم ما از الگوریتم پیاده سازی شده بدست آورد، در بد ترین حالت (زمانی که اندازه جدول بازی ۲۰x۲۰ باشد و کل خانه های بازی نیز دارای الماس باشد) پیچیدگی زمانی آن برابر با ۱۶۱۰۰۰ عمل در هر نوبت می‌باشد. طبق اندازی گیری که ما بر روی چند سیستم انجام دادیم تقریبا زبان پایتون ۱۰۰۰۰۰۰۰ عمل را در یک ثانیه انجام میدهد (توان هر سیستم در نتیجه تاثیر گذار است). بنابر این این الگوریتم به شکل بهینه ای در محدوده زمانی ۱ ثانیه عمل میکند.

استراتژی که تیم ما برای کنترل محدودیت زمانی در الگوریتم تپه نوردی انتخاب کرده است محدود کردن طول زمین بازی می‌باشد. به این صورت که اگر نیاز به انجام عملیات کمتری در واحد زمان بود محدودی بررسی زمین بازی را به NxN کاهش میدهیم و در واقع محیط بازی برای عامل از مشاهده پذیر کامل به مشاهده پذیر جزئی تغییر میابد.