درس رایانش تکاملی

گزارش پروژه بازی Pac-Man

فاطمه پارسا

مرداد 1400

مقدمه:

در این پروژه با استفاده از الگوریتم Genetic Programing بازی Pac-Man را طراحی کرده ایم.

اهداف انجام این پروژه:

1. فرمولبندی مسائل پیچیده بهینه سازی
2. پیاده سازی الگوریتم تکاملی (EA) برنامه نویسی ژنتیک (GP)
3. انجام آزمایشهای علمی برای EA
4. تحلیل نتایج تجربی الگوریتمهای تصادفی از نظر آماری
5. نوشتن گزارشهای فنی مناسب

در این مسئله یک کنترولر با الگوریتم Genetic Programing برای Pac-Man. و یک کنترولر برای Ghost ها نوشته ایم. و در نهایت در بدنه اصلی برنامه، در هر دور بازی، این کنترولر ها اجازه انجام یک حرکت برای ایجنت مربوطه را دارند. و در نهایت ارزیابی نهایی این کنترولر ها، با بررسی چند دور بازی و امتیاز ها انجام شده است.

در ادامه ابتدا به شرح روش بازی میپردازیم و سپس روش پیاده سازی را تشریح کرده و کارایی برنامه را تحلیل میکنیم.

شیوه بازی:

GPac

در GPac، فضای مسأله یک صفحه (World) دو بعدی است. دو نوع کاراکتر وجود دارد: Pac-Man[۲] و Ghosts. Pac-Man، همیشه از سلول بالایی سمت چپ شروع می‌کند و اشباح (سه تا شبح) همیشه از سلول پایین سمت راست شروع می‌کنند. کاراکترها توسط کنترلرها هدایت می‌شوند، که بایستی توسط الگوریتم GP شما تکامل یابند. کاراکترها در چهار جهت اصلی (بالا ، پایین ، چپ ، راست) حرکت می‌کنند؛ Pac-Man قابلیت حفظ موقعیت خود (توقف) را دارد، اما اشباح خیر، آنها مدام به شکل گسسته از یک سلول صفحه به سلول دیگر منتقل می‌شوند. کاراکترها نمی‌توانند از لبه‌های صفحه خارج شوند. اشباح می‌توانند در سلولی که توسط اشباح دیگر اشغال شده قرار بگیرند. اگر Pac-Man و یک شبح در یک سلول قرار بگیرند، بازی تمام شده است. اگر Pac-Man و Ghost با هم برخورد کنند (یعنی از یک سلول عبور کنند)، بازی تمام شده است.

دیوارها (Walls)

هنگام ایجاد هر بازی، قبل از قرار دادن قرص‌ها (Pills)، دیوارها به طور تصادفی با محدودیت های زیر ایجاد می‌شوند:

۱- چگالی دیوار باید توسط کاربر قابل تنظیم باشد، چگالی دیوار (به درصد) احتمال وجود دیوار در یک سلول را نشان می‌دهد (شبیه چگالی قرص)،
۲- دیوارها را نمی‌توان در سلولهای شروع Pac-Man یا اشباح قرار داد،
۳- هیچ چیزی در یک سلول با دیوار قرار نمی‌گیرد، یا هیچکس از روی آن عبور نمی‌کند،
۴- باید تمام سلولهای غیر دیوار از سایر سلولهای غیر دیوار قابل دسترسی باشند.

توجه: اگرچه شما قرار است چگالی دیوار را مشخص کنید، اما می توانید از فرمول زیر برای تعیین تعداد مورد انتظار دیوارها استفاده کنید:

$$E[number\ of\ walls] = MAX\ (1\ ,\ wall\text{-}density\ .\ (total\ number\ of\ cells\ -\ 2))$$

قرص‌ها (Pills)

قبل از شروع بازی، سلول‌ها به طور تصادفی و بر اساس پارامتر از پیش تعیین شده برای چگالی قرص به عنوان"قرص" انتخاب می‌شوند. پارامتر چگالی قرص، درصد احتمال وجود یک قرص در یک سلول با محدودیت‌های زیر را مشخص می‌کند:

الف) حداقل یک سلول باید حاوی قرص باشد،

(ب) قرص ها را نمی‌توان روی دیوارها قرار داد و

ج) قرص‌ها را نمی‌توان در سلول شروع Pac-Man قرار داد.

بنابراین:

$$E[\text{number of cells containing a pill}] = MAX(1, \text{pill-density} \cdot (\text{total number of cells} - 1 - \text{number of walls}))$$

اگر Pac-Man سلولی را که حاوی قرص است اشغال کند، قرص برداشته می‌شود و امتیاز Pac-Man افزایش می‌یابد. وقتی همه قرص‌ها از صفحه حذف شدند، بازی تمام می‌شود.

میوه (Fruit)

هر نوبت که بازی اجرا می‌شود، یک فرصت (قابل تنظیم) به کاربر داده می‌شود تا یک تکه میوه را در صفحه قرار دهد. هر بار فقط یک تکه میوه می‌تواند در صفحه قرار بگیرد و این تکه میوه در سلول قرص، دیوار یا سلول فعلی Pac-Man نمی‌تواند قرار بگیرد. اگر Pac-Man سلول حاوی تکه میوه را اشغال کند، میوه برداشته می‌شود، و امتیاز Pac-Man به اندازه امتیاز (قابل تنظیم) تکه میوه افزایش می‌یابد.

زمان (Time)

هر بازی GPac با زمانی برابر با تعداد سلولهای صفحه ضربدر یک عدد زمان از قبل تعریف شده شروع می‌شود. هر نوبت یک بازه‌ی زمانی دارد. وقتی محدودیت زمانی به صفر رسید، بازی تمام می‌شود، که این عمل از گیر افتادن در حلقه‌های بی‌نهایت مانع می‌شود. همچنین باعث تکامل موثر کنترلر می‌شود.

اجرای بازی

در هر نوبت، بازی به هر یک از کنترلرهای کاراکترها وضعیت بازی فعلی را می‌دهد. این حالت حداقل شامل موارد زیر است: جایی که همه کاراکترها در حال حاضر واقع شده‌اند و همه قرص‌ها در آن قرار دارند. سپس هر کنترلر انتخاب می‌کند که چه حرکتی انجام شود (بالا، پایین، چپ، راست برای همه کنترلرها، و توقف فقط برای Pac-Man). یک بار که همه‌ی کاراکترها حرکت بعدی خود را تعیین کردند، وضعیت بازی موقعیت همه را به روزرسانی می‌کند و زمان باقیمانده یکی کاهش می‌یابد. پس از حرکت همه، بازی بررسی می‌کند که:

۱- اگر Pac-Man و هر یک از اشباح در یک سلول هستند، بازی تمام شود.
۲- اگر Pac-Man با یک شبح برخورد کرده است، بازی تمام شود.
۳- اگر Pac-Man در یک سلول همراه با یک قرص قرار گرفته است، قرص برداشته شود و امتیاز تغییر کند.
۴- اگر Pac-Man در یک سلول با یک تکه میوه قرار دارد، میوه برداشته شود و امتیاز تغییر کند.

۵- اگر تمام قرص‌ها برداشته شده‌اند، بازی تمام شود.

۶- اگر زمان باقیمانده برابر با صفر است، بازی تمام شود.

امتیاز (Score)

امتیاز Pac-Man برابر با امتیاز کل قرص‌هایی که او به طور خلاصه مصرف کرده به علاوه امتیاز میوه‌های مصرف شده است. اگر بازی به دلیل اتمام قرص‌ها به پایان برسد، امتیاز Pac-Man با توجه به درصد زمان باقیمانده با یک عدد صحیح افزایش می‌یابد. این امتیاز می‌تواند مستقیماً بعنوان مقدار شایستگی[۳] کنترلر Pac-Man استفاده شود. مقدار شایستگی شبح باید با عکس شایستگی Pac-Man متناسب باشد (به عنوان مثال، شایستگی او را منفی شایستگی Pac-Man در نظر بگیرید).

نحوه پیاده سازی بازی:

برنامه از چند قسمت تشکیل شده است.

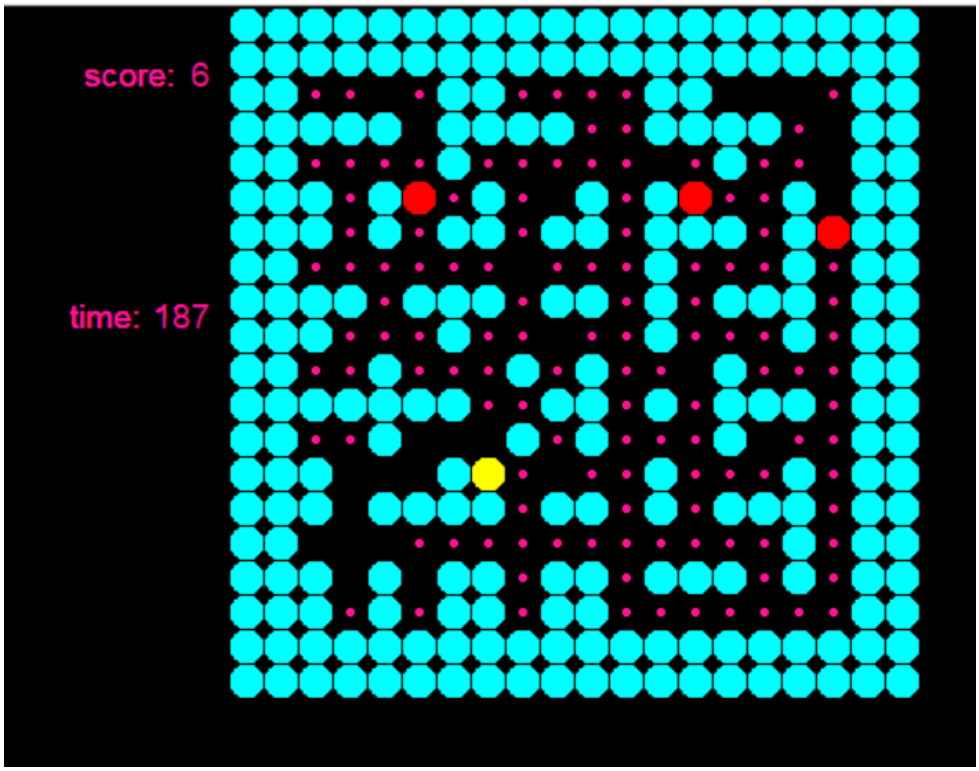
1. بدنه اصلی بازی
2. ساخت اسکلت دنیای بازی: شامل دیوار ها، راهرو ها و قرص ها
3. تابع کنترلر انجام یک دور بازی توسط هر ایجنت و چک کردن شرایط برنده و بازنده
4. تابع کنترلر پکمن
5. تابع کنترلر گوست ها
6. میکرو توابع فرعی برای انجام بعضی کار های مورد نیاز در برنامه
7. رابط گرافیکی بازی برای تحلیل بهتر کنترولر

بدنه اصلی برنامه:

بدنه اصلی برنامه با مقدار دهی اولیه به طول و عرض صفحه بازی و نسبت دیوار های موجود در صفحه به راهرو ها و درصد قرص ها در راهرو ها شروع می شود. سپس محل شروع پکمن و گوست ها مشخص می شود. زمان بازی 200 و امتیاز اولیه 0 قرار داده می شود.

سپس دنیای بازی شامل دیوار ها، راهروها و قرص ها با فراخوانی تابع world در این برنامه ساخته می شود. بدین صورت که ماتریسی 20 *20 شامل اعداد 0 و 1 ساخته می شود که 0 نشان دهنده دیوار و 1 نشان دهنده راهرو است. سپس قرص ها با عدد 2 جایگزین تعدادی از عدد های 1 در ماتریس می شوند. به این صورت تعدادی از خانه ها شامل قرص می شوند.

عمل دیگری که در این تابع اتفاق می افتد، ساخته شدن گرافیک بازی است. با استفاده از کتابخانه turtle قسمت های دیوار دنیای پکمن را با دایره های آبی، گوست ها را با دایره های قرمز، پکمن را با دایره زرد و قرص ها را با دایره های کوچک صورتی نشان داده ایم. هر مربع 20*20 پیکسلی از صفحه برابر با یک درایه از ماتریس دنیای پکمن است. هر حرکت در چهار جهت اصلی معادل یک واحد حرکت در ماتریس و 20 پیکسل در صفحه گرافیکی است.



شکل شماره 1 : تصویری از شکل گرافیکی بازی

در این مرحله که دنیای بازی طراحی شده است، حالا هر ایجنت باید یک حرکت انجام دهد. این کار با فراخوانی تابع move انجام می شود. و این تابع هر 10 میلی ثانیه یک بار صدا زده می شود تا جایی که یکی از شرایط توقف ارضا شوند. شرایط توقف شامل اتمام زمان، اتمام قرص و برخورد یکی از گوست ها با پکمن است. تابع move خود حاوی دو تابع اصلی است. یک تابع شامل کنترولر پکمن و یک تابع حاوی کنترولر گوست. هر بار اجرا شدن تابع move مساوی است با یک نوبت حرکت در بازی.

ساختار این تابع به این گونه است که نقشه دنیای بازی و موقعیت ایجنت ها را به تابع کنترولر پکمن می دهد و این تابع یک رشته از حرکات بعدی را پیشنهاد می دهد. که اولین عنصر این رشته بررسی و در صورت معتبر بودن اجرا می شود. طول این رشته می تواند متغیر باشد. در قسمت های آینده این گزارش نحوه عملکرد تابع کنترولر آورده شده است.

پس از انتخاب حرکت توسط پکمن، موقعیت پکمن آپدیت شده و در صفحه گرافیکی نیز یک واحد حرکت می کند. اگر خانه جدید که پکمن به آن وارد می شود در ماتریس دنیای بازی حاوی شماره 2 باشد، یک عدد به امتیاز پکمن اضافه شده . و شماره آن درایه از ماتریس به 1 تغییر کرده و گرافیک آن آپدیت شده و قرص از آن خانه حذف می شود. و پکمن به جای آن خانه منتقل می شود.

سپس کنترولر گوست ها برای هر گوست یک حرکت جداگانه پیشنهاد می دهد و آنها نیز مشابه پکمن جا به جا می شوند و موقعیتشان آپدیت می شود.

سپس زمان یک واحد کم می شود.

در این زمان که همه ایجنت ها یک حرکت انجام داده اند، شرط توقف برنامه چک می شود. شرط زمان =0 و شرط برخورد

پکمن با گوست باعث پایان بازی و باخت پکمن می شود.

شرط اتمام قرص ها باعث برنده شدن پکمن می شود.

تابع move تا زمانی که یکی از شرط ها ارضا شود هر بار بعد از 10 میلی ثانیه تکرار می شود.

تابع کنترولر پکمن:

در این تابع با داشتن نقشه دیوار ها، راهرو ها، قرص ها و محل گوست ها باید تصمیمی اتخاذ کنیم برای بهترین حرکت ممکن. اما چون این کار عملا غیر ممکن است، با استفاده از الگوریتم تکاملی حرکتی مناسب درزمانی معقول پیدا می کنیم. اما تضمینی وجود ندارد که این جواب بهینه باشد. بنابراین ساختارر این کنترلر بر اساس برنامه نویسی ژنتیک است.

هر رشته حرکت که حاوی چندین مرحله حرکت های بعدی پکمن است یک کرومزوم محسوب می شود. اما طول این رشته به دلیل ساختار درختی برنامه نویسی ژنتیک، متغیر است. زیرا شکل درخت در طول همبری های متعدد تغییر می کند.

در این تابع پکمن دو انتخاب دارد. یک انتخاب ادامه رشته مسیر قبلی که در اختیار داشته و یک انتخاب از طریق الگوریتم ژنتیک بر اساس حرکت های جدید گوست ها به دست می آید. این دو جواب با هم مقایسه شده و بر اساس معیار ها ارزیابی می شوند. رشته حرکت بهتر جایگزین رشته حرکت قبلی می شود. و حرکت اول این رشته به عنوان حرکت پیشنهادی برگردانده می شود.

در تابعی که الگوریتم ژنتیک را پیاده سازی می کند از توابع مخصوص انتخاب و همبری و جهش استفاده شده است.

ابتدا مقادیر اولیه تنظیم می شود. یک جمعیت اولیه با طول 2 و 3 حرکت در چهار جهت های اصلی به وجود می آید.

تعداد بار مشخصی الگوریتم تکرار می شود و بهترین جواب بدست آمده در دور آخر برگزیده می شود.

یک رشته به طور مثال می تواند به این شکل باشد:

[U,D,L,R,U,U,R]

در تابع شایستگی بر اساس پنج فاکتور به یک رشته حرکت یا همان کرومزوم جواب یک ارزش نسبت می دهیم.

1. فاصله حرکت آخر تا نزدیک ترین گوست(مثبت)
2. فاصله حرکت آخر تا نزدیک ترین قرص (منفی)
3. تعداد قرص های موجود در مسیر(مثبت)
4. وجود دیوار در مسیر (منفی)
5. وجود گوست در مسیر (منفی)

بر اساس این پنج ویژگی یک عدد صحیح به عنوان شایستگی برای این کرومزوم محاسبه می شود.

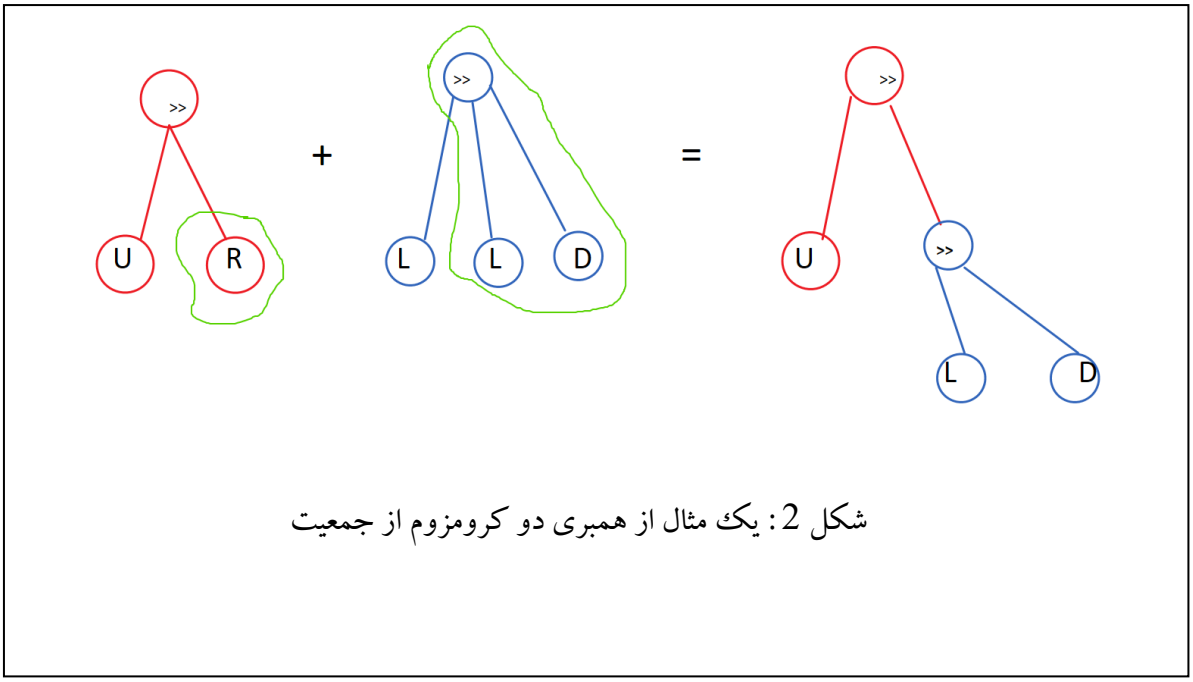
پس از مشخص شدن شایستگی تمام جمعیت تابع انتخاب تنها جمعیتی را انتخاب میکند که به هیچ وجه دیوار در مسیرشان

وجود نداشته باشد.

و بقیه جواب ها را بر اساس کیفیت شایستگی تکثیر می کند. و به حوضچه همبری منتقل می نماید.

در حوضچه همبری چون کروموزوم ها به صورت درخت هستند، همبری باعث می شود طول کروموزوم ها متغیر شود.

به طور مثال:



شکل 2 : یک مثال از همبری دو کرومزوم از جمعیت

کرومزوم هایی که طولشان بیشتر از 10 باشد را کوتاه می کنیم. زیرا جواب نباید شامل رشته ای خیلی بلند از حرکات باشد. به علت پویایی دنیای بازی و تفاوت زیاد در چند حرکت.

حرکات اضافی که رفت و برگشت به یک خانه هستند را اصلاح میکنیم. مثلا :

[U,L,R,U] = [U,U]

تابع کنترولر گوست ها:

برای گوست ها نیز می توان از همان توابع ژنتیک استفاده کرد. اما به دلیل اینکه گوست ها سه تا هستند و تصمیم گیری برای گرفتن پکمن باعث می شود قدرت آنها نسبت به پکمن خیلی نامتناسب زیاد شود و پکمن بلا استثنا ببازد برای کنترل قدرت در بازی ترجیح می دهیم کمی شانسی عمل کند. به این صورت که از بین چهار جهت اصلی جهتی را انتخاب کرده که به دیوار برخورد نکند. و اگر خیلی به پکمن نزدیک بود او را بگیرد. اما از فواصل زیاد نمیتوان پکمن را دنبال کند.

نمونه ای از رشته های انتخاب شده در هر نوبت برای پکمن:

```
['U', 'U', 'L', 'L', 'R', 'R', 'R']
['U', 'L', 'L', 'R', 'R', 'R']
['R', 'D', 'R', 'U', 'R', 'L', 'U']
['D', 'R', 'U', 'R', 'L', 'U']
['R' 'U' 'U' 'D' 'D']
['U', 'R', 'R', 'R', 'L']
['R', 'R', 'R', 'L']
['R', 'L', 'U', 'D', 'U', 'D', 'U']
['D', 'D', 'U', 'U', 'D']
['D', 'R', 'R', 'D', 'U', 'U', 'R']
['R', 'R', 'D', 'U', 'U', 'R']
['R' 'D' 'U' 'U' 'R']
['D' 'U' 'U' 'R']
['U' 'U' 'R']
['U', 'U', 'D', 'U', 'D', 'D', 'U']
['U', 'L', 'R', 'R', 'U', 'U', 'L', 'R']
['L', 'R', 'R', 'U', 'U', 'L', 'R']
['R', 'R', 'U', 'R', 'D', 'D', 'U']
['R', 'D', 'R', 'L', 'R']
['D', 'R', 'L', 'R']
['R', 'U', 'U', 'R', 'R', 'U', 'L']
['U', 'U', 'R', 'R', 'U', 'L']
['U', 'R', 'R', 'U', 'L']
['R', 'R', 'D', 'D', 'U']
['L', 'L', 'U', 'D', 'U']
['L', 'U', 'U', 'U', 'L', 'L']
['U' 'U' 'U' 'L' 'L']
['R', 'R', 'U', 'R', 'L']
['R', 'U', 'R', 'L']
['D', 'R', 'D', 'U', 'R']
['U', 'D', 'R', 'D', 'U', 'D']
['D', 'R', 'D', 'U', 'D']
['R', 'D', 'U', 'D']
['U', 'U', 'D', 'D', 'U', 'U', 'U', 'U']
['D', 'D', 'D', 'D', 'R']
```

نتیجه گیری:

این روش به هوشمند سازی انتخاب های پکمن کمک می کند. با چندین دور اجرای بازی و تحلیل انتخاب های پکمن می توان هوشمندی عامل را به وضوح دید. با تغییراتی در پارامتر ها و معیار ها می توان هوشمندی را بیشتر کرد. حتی میتوان به دلخواه گوست ها را نیز با همین روش هوشمند تر کرد. تنها تفاوت با پکمن در معیار شایستگی رشته حرکت انتخابی است. که در گوست ها می توان آن را فاصله تا پکمن قرار داد. اما باید دید گوست را کم کرد تا تناسب قدرت در بازی به هم نریزد.