گزارش پروژه درس هوش مصنوعی

عنوان پروژه: حل معمای Rikudo به وسیله الگوریتم ژنتیک

نام و نام خانوادگي: امیرعباس زارعی، امیرحسین رجبی، نیایش سیف ، محمدرضا باطنی

شماره دانشجويي : ۹۷۱۳۰۶۶، ۹۸۱۳۰۱۳، ۹۸۱۳۰۶۸، ۹۸۱۲۰۰۶

دانشگاه صنعتي اميركبير دانشكده رياضي و علوم كامپيوتر

استاد درس: دكتر محمد اكبري

تیر ۱۴۰۱

تعریف مسئله

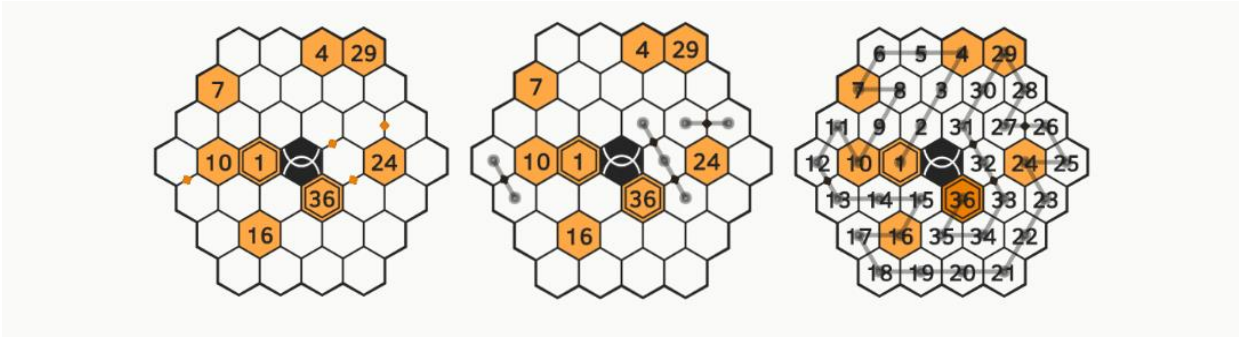
مسئله، حل معمای ریکودو (Rikudo) یک نوع پازل اعداد میباشد.

بازی ریکودو :

صفحه بازی، یک جدول شش ضلعی است که هر خانه آن نیز یک شش ضلعی میباشد. تعداد خانه ها میتواند ۳۶ یا ۶۰ یا ۹۰ باشد. در ابتدا در تعدادی از خانه های جدول اعداد متفاوتی (از بین اعداد ۱ تا تعداد خانه های جدول) نوشته شده است . کوچک ترین عدد (۱) و بزرگترین عدد (۳۶ یا ۶۰ یا ۹۰) همواره در این اعداد وجود دارد. بر روی بعضی از ضلع های خانه ها هم نقاط نارنجی رنگ وجود دارد.

هدف پر کردن بقیه خانه های جدول با اعداد باقی مانده به نحوی میباشد که در انتها از خانه عدد ۱ به ترتیب پشت سر هم تا خانه عدد انتهایی یک مسیر موجود باشد. به علاوه برای هر نقطه نارنجی دو خانه دو طرف ضلعی که آن نقطه نارنجی روی آن قرار دارد؛ دو خانه شامل اعداد پشت سر هم باشند.

شکل سمت چپ یک مثال از ابتدای بازی با تعداد خانه ۳۶ و شکل سمت راست یک جواب این پازل میباشد.



هدف ما یافتن جواب این معما است.

راه حل مورد استفاده:

ما برای حل این پازل از الگوریتم ژنتیک استفاده کردیم و پارامتر های الگوریتم ژنتیک را برای این بازی بهینه کردیم. در این مقاله توابع و پارامتر های مورد نیاز، توضیح داده میشوند و در نهایت نشان داده میشود که الگوریتم ژنتیک تا چه حد توانسته جواب این معما را به دست آورد.

توضیح کوتاهی از الگوریتم ژنتیک:

الگوریتم ژنتیک، یک الگوریتم تکاملی است که با یک جمعیت اولیه (جواب های احتمالی مسئله) که به هر کدام یک "ژن" میگوییم کار خود را شروع میکند. در هر مرحله سعی میکند با استفاده از تعدادی عملیات روی ژن ها (crossover و mutation) تعدادی ژن به جمعیت اضافه کند و در نهایت تعدادی از اعضای این جمعیت که مقدار fitness کمتری دارند را با احتمال بیشتری حذف میکند و جمعیت را از یک نسل به نسل بعدی ثابت نگه میدارد. در نهایت پس از تعدادی iteration جمعیت نهایی، مقدار fitness خوبی دارند و میتوان بهترین عضو آن جمعیت را به عنوان خروجی مدل در نظر گرفت.

توضیح الگوریتم ژنتیک برای حل معمای ریدوکو:

در ریدوکو تعدادی خانه ها مشخص اند، پس بهتر است مقدار آن خانه ها را در ژن دخیل نکنیم تا الگوریتم، دنبال مقدار آن ها نگردد (چون مقدار را از قبل فیکس کرده ایم.) به همین دلیل روی تمام خانه های پازل جستجو میکنیم، خانه هایی که خالی اند را در نظر میگیریم و مختصات آن ها را ذخیره میکنیم. همچنین ذخیره میکنیم که چه اعدادی باید در آن خانه ها باشند (مثلا اگر جدول ۵ تایی باشد و جایگاه ۱ و ۳ و ۵ مشخص باشد مان نیاز داریم جایگاه ۲و۴ را بدانیم.) بنابراین کافی است جایگاه آن خانه ها را بدانیم و این همان متغیری است که الگوریتم ژنتیک ما به دنبال آن میگردد.

به همین دلیل جایگشتی از اعداد گفته شده (مقادیر ممکن خانه های خالی) تشکیل میدهیم و در نظر میگیریم که هر جایگاه جایگشت متناظر چه نقطه ای در جدول است و این جایگشت را به عنوان یک ژن در نظر میگیریم و الگوریتم را اجرا میکنیم.

برای اجرای الگوریتم ژنتیک نیاز به توابع random_population (تابع ایجاد جمعیت اولیه) و mutation (تابع جهش ژنتیکی روی یک ژن) و crossover (تابع کراس اوور دو ژن) و objective (مقدار تابع فیتنس) و تعدادی پارامتر دیگر داریم. در ابتدا مقدار پارامتر ها را ذکر میکنیم سپس به نحوه کار کردن تابع ها میپردازیم. توجه شود که پارامتر های این مسئله از سعی و خطی به دست آمده و بهترین شاهد برای کارکرد این پارامتر ها جواب خروجی و نمودار ها هستند.

مقدار پارامتر ها:

population_size = 1000

crossover_coeff = 0.1

mutation_coeff = 2

epochs = 500

توضیح توابع:

Crossover : در اینجا هدف، کراس اوور دو جایگشت از اعضا است. برای این کار این دو لیست که شامل اعدادی بین ۱ تا n هستند را به اعداد ۱ تا k متناظر میکنیم (چون تعدادی از اعداد را رد کردیم) و از کراس اوور دو جایگشت استفاده میکنیم. کراس اوور دو جایگشت هم این است که هر بار عدد ai را از جایگشت اول دیدیم، عدد b(ai) را در جایگشت جدید قرار میدهیم و هر بار عدد bj را دیدیم عدد a(bj) را در جایگشت جدید دیگری قرار میدهیم. بدین ترتیب دو جایگشت جدید ایجاد میشوند. به مثال زیر دقت کنید.

Crossover: [1, 3, 2] [3, 2, 1] -> [3, 1, 2] [2, 3, 1]

Mutation: در اینجا دو عنصر از جایگشت به طور تصادفی انتخاب میشوند و با هم جابجا میشوند.

Objective: ایده اصلی این مقاله در یافتن این تابع بود. در این تابع پارامتری به یک استیت از مسئله نسبت میدهیم که هر چه بزرگ تر شود به جواب نزدیک تر میشود.

چند تابع fitness داریم که fitness1 ، تابع پیش فرض برای محاسبه objective میباشد.

در تابع fitness1 ۳ پارامتر تعداد نقاط سیاه ارضا شده ، تابعی از مجموع فاصله های سلول های اعداد پشت هم و بزرگترین دنباله صحیح اعداد پشت هم با شروع از ۱ هر کدام با ضریبی لحاظ شده اند.

توضیح ساختار کد ها:

در این پروژه چهار فایل وجود دارد:

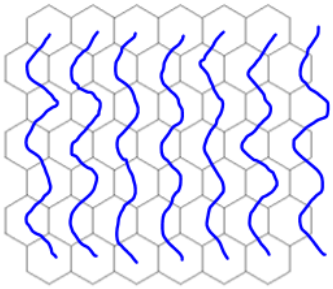
فایل genetic.py : در این فایل کد های خام مربوط به الگوریتم ژنتیک نوشته شده (مستقل از صورت بندی مسئله) و سه کلاس Gene (برای نگه داشتن مقدار عناصر یک ژن و objective آن) و GeneticAlgorithmBehaviour (برای نگه داشتن توابع مورد نیاز مدل) و GeneticAlgorithmModel (برای ساختن مدل و حل مسئله) در آن تعریف شده. تمام این کد ها توسط خوودمان نوشته شده و از هیچ ماژول آماده ای استفاده نشده.

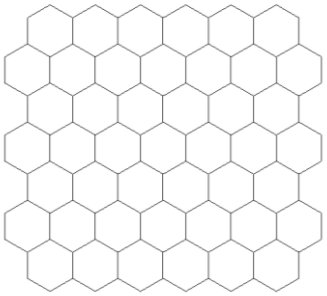
فایل gui.py : در این فایل تمام کد های مربوط به نمایش گرافیکی صفحه زده شده. در این فایل یک تابع draw_puzzle وجود دارد که با صدا زدن آن و ورودی دادن چیز هایی که میخواهیم در صفحه کشیده شوند، یک صفحه گرافیکی کشیده میشود و نمودار ها و شکل های مورد نیاز کشیده میشوند. کد های مربوطه در این فایل هم (کشیده صفحه پنج ضلعی ها و نمودار ها و...) هم توسط خود ما زده شده.

فایل puzzle.py : همانطور که مشخص است تعدادی از خواص پازل، به جایگاه نقاط در صفحه شش ضلعی بستگی دارد و به خواص خود پازل وابسته است، در این فایل تمام توابع مورد نیاز در یافتن objective و باقی نقاط کد، که به صفحه پنج ضلعی مرتبط است، نوشته شده.

شکل جدول استفاده شده :

لازم به ذکر است که مبنای یک صفحه ریدوکو، در این برنامه، صفحه شطرنجی n*m مانند شکل زیر در نظر گرفته شده

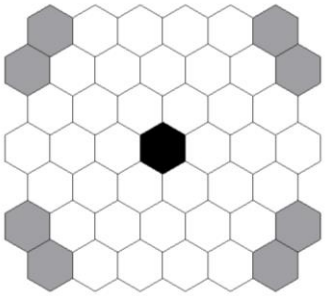




مثلا در این شکل ۷ سطر داریم از سطر ۰ تا ۶ از بالا شماره گذاری میشوند.

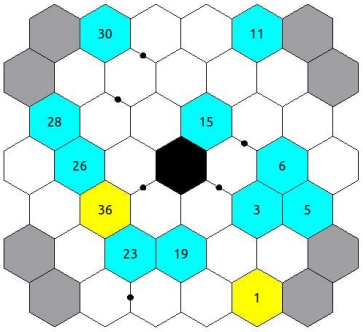
و ۷ ستون داریم که با خطوط آبی نمایش داده شده. (در ستون هفت تنها ۳ خانه در جدول موجود است).

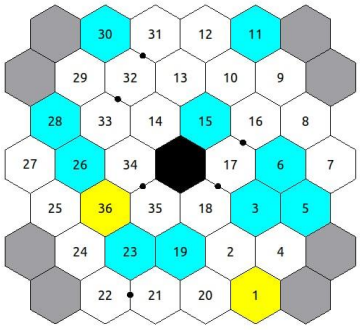
و اگر نیاز باشد نقاطی از اطراف وجود نداشته باشد، کافی است در فایل input به جای مقدار آن ها -۱ (برای نقاط خاکستری) و -۲ (برای نقاط مشکی) بگذاریم تا مثلا به صورت شکل زیر درآید.



به همین دلیل تمام خواص مختلف، مثل فاصله مختصات ها با استفاده از الگوریتم های مسیریابی به دست میایند تا این برنامه برای عمومی ترین حالت و هر ریدوکو ای جواب گو باشد، حتی ریدوکو ای که فرم صفحه به صورتی غیر متعارف باشد.

فایل main.py : در این فایل کار های کلی انجام میشود. فایل ریدوکو از input.txt خوانده میشود، توابع crossover و... و پارامتر های الگوریتم ژنتیک تعریف میشوند و الگوریتم ژنتیک اجرا میشود و در نهایت جواب نهایی روی صفحه نمایش داده میشود. و در هر بخش از فایل های گفته شده استفاده میشود.





شکل سمت راست: پازل حل نشده شکل سمت چپ: پازل حل شده

سلول های زرد : سلول های حاوی مقادیر ۱ و max number که همواره در ابتدای پازل مکانشان مشخص شده است.

سلول های آبی : سلول های حاوی مقادیری که در ابتدای پازل تعیین شده.

سلول های سفید : سلول هایی که مقادیر آن باید مشخص شود.

سلول های خاکستری : سلول هایی که هیچ مقداری نمیگیرند. (اضافی هستند)

جواب گزارش داده شده و جدول نتایج :

نتیجه برای پازل ورودی :