به نام خدا

گزارش پروژه سوم کاربرد های هوش مصنوعی

سید مهدی میرفندرسکی۹۷۲۳۰۹۳

ستایش ثانوی۹۶۲۸۰۲۴

مقایسه‌ی سرعت FC , MAC :

برای نمونه ورودی puzzle3 را در نظر گرفتیم و زمان مک در آن : ۰,۰۶۶۰۴۲۶۶۱۶۶۶۸۷۰۱۲

و در مورد FC زمان : ۰,۰۱۹۹۷۷۰۹۲۷۴۲۹۱۹۹۲۲

در این نمونه زمان مک بیشتر شده برای مورد بعدی پازل ۵ را در نظر میگیریم :

Mac : ۳,۸۵۶۹۸۰۸۰۰۶۲۸۶۶۲

Fc : ۲,۶۱۷۸۶۲۷۰۱۴۱۶۰۱۵۶

در هر دو مورد سرعت FC بهتر بوده چون همانطور که میدانیم در مک تمام حالات arc_consistency بررسی میشود و به همین دلیل درخت جست‌جوی کوچک تری دارد در حالی که سربار پردازشی را بالا میبرد و زمان بیشتری برای محاسبه میخواهد در حالی که در یک trade off در FC با اینکه احتیاج به گسترش بیشتر گره ها هستیم ولی پردازش سریع تر انجام میشود.

توضیح کلی توابع :

در ابتدا فایل را میخوانیم و بعد به شکل ماتریس آن را در توابع استفاده میکنیم. ما متغییر ها را همان سطر ها و ستون ها گرفتیم نه هر خانه تا بتوانیم با محدودیت های باینری کار کنیم. همچین برای تابع هیوریستیک نیز از MRV استفاده کردیم تا در هر لحظه با توجه به دامنه متغییر ها آن که محدود تر است را انتخاب کنیم.

ابتدا در کلاس tuple دو محدودیت برابر بودن تعداد صفر و یک ها در یک سطر یا ستون تحت عنوان EqualityOfZO و محدودیت متوالی نبودن بیش از دو صفر یا دو یک را در تابع ValidationOfPositionsZO بررسی میکنیم.

در تابع genBin به ازای هر خانه خالی در ماتریس صفر و یک قرار میدهد و به صورت بازگشتی تکرار میشود و اگر هیچ خانه خالی ای وجود نداشت و محدودیت ها هم برقرار بودند جواب بدست آمده را در res میریزیم.

در تابع selectUnassignedVariable ابتدا در یک guard clause بررسی میکنیم اگر دامنه‌ای برای سطر و ستون‌ها وجود نداشت، مقدار -۱ را برمیگردانیم تا به ادامه کد نپردازیم درغیر این صورت در این تابع ابتدا کمترین تعداد دامنه برای هر سطر و ستون را بدست می‌آوریم (به خاطر MRV) و آن را به عنوان متغییر بعدی برای بررسی انتخاب میکنیم. توجه کنید که اگر متغیر row برابر یک بود یعنی دامنه‌ی انتخاب شده برای سطر است و اگر صفر بود برای ستون خواهد بود.

در تابع isSafe دو محدودیت اولیه یعنی ValidationOfPositionsZO و EqualityOfZO برای متغیر داده شده بررسی میشود.

در تابع inference با توجه به پارامتر های ورودی : row سطر یا ستون بودن را تعیین میکند ، selected سطر یا ستونی که به عنوان متغیر انتخاب شده و value که دامنه‌ی ممکن برای هر سطر یا ستون است که در تابع backtack توضیح داده خواهد شد.

برای محدودیت FC اگر در سطر بودیم، تمام مقادیر یکسان با مقدار assign شده به متغیر جدید را از دامنه‌ی مابقی سطر ها به همراه شماره سطر را به یک لیست اضافه میکنیم و نهایتا لیست را به rowDict برای آن آیتم قرار میدهیم تا دامنه داده شده به هر متغیر را در دیکشنری ذخیره کنیم به همین منوال دیکشنری برای ستون نیز میسازیم. اگر یک دامنه را برای سطری انتخاب کردیم با توجه به آن دامنه های ستون ها را به روز رسانی میکنیم تا بین سطر و ستون ها هماهنگی باشد و به همین منوال برای ستون ها. اگر دیکشنری های ساخته شده خالی بودند یعنی نباید از دامنه ی آن متغیر مقداری حذف شود و مقدار False برگردانده میشود. درواقع rowDict و colDict نشان دهنده ی مقادیری هستند که از متغیرها باید حذف شوند.

برای محدودیت MAC اگر سطر انتخاب شده بود، از دامنه قبلا مقدار دهی نشده‌ی سطر ها یکی را تحت متغیر tempTuple اییجاد میکنیم تا بتوانیم آن را به queue اضافه کنیم و به همین شکل هم برای ستون یک دامنه را به صف اضافه میکنیم و به همین منوال اگر ستون انتخاب شده بود از دامنه‌ی ستون ها یکی از دامنه‌های قبلا مقدار دهی نشده را انتخاب کرده و به صف اضافه میکنیم تا در ادامه در تابع AC3 به ترتیب بررسی شوند.

در تابع backtrack ابتدا در یک guard clause بررسی میکنیم که اگر همه‌ی متغیر های موجود مقدار دهی شدند در این صورت مقدار true را برمیگرداند و نیازی به انجام ادامه‌ی کد نداریم ولی در غیر این صورت در این تابع با توجه به مقدار متغیر row و selected که از دامنه‌ی انتخاب شده با هیوریستیک MRV در selectUnassignedVariable بدست آمد متغیر value را با یک دامنه خاص مقدار دهی میکنیم و آن را در دیکشنری مقدار دهی شده ها قرار میدهیم تا دوبار بررسی نشود، بعد با صدا زدن تابع inference روی این دامنه‌ی انتخاب شده مقادیر ممکن و مجاز را برای انتخاب دامنه ها برمیگردانیم تا استنتاجی صورت بگیرد اگر مقداری مجاز وجود داشت و تابع مقدار true را برگرداند با توجه به سطر یا ستون بودن، آن مقدار را از دامنه حذف میکنیم که معادل انتخاب شدن است تا به صورت بازگشتی روی دامنه مراحل بعدی نیز این کار را انجام دهیم حال اگر تمام مقادیر در این تکرار ها مقدار دهی شدند مقدار true را برمیگرداند و درغیر این صورت یعنی آن مقدار انتخاب شده مناسب نبوده و دامنه برای مراحل بعدی تمام شده پس باید یک مقدار دیگر را انتخاب و بررسی کرد تا مقدار اگر جواب موجود بود بدست آید و در غیر این صورت ذکر شود که جواب وجود ندارد.

در تابع AC3 صف تشکیل شده از دامنه ها را که در تابع inference تشکیل دادیم به صورت ورودی میگیرد و تا زمانی که مقداری در این صف وجود دارد بررسی میکند :

متغیر سطر را به متغیر Xi و متغیر ستون را به Xj میدهیم و به تابع revise میدهیم در این تابع Xi نسبت به Xj ، arc_consistent میشود به این صورت که برای تمام مقادیری که در سطر وجود دارد اگر مقداری در دامنه‌ی ستون بود که محدودیت بین xj,xi را نقض میکرد باید از دامنه سطرها حذف شود و بعد اگر هیچ محدودیتی نقض نمیشد آن مقدار مناسب است و میتواند در دامنه باقی بماند. در ادامه‌ی تابع ac3 بعد از گرفتن دامنه ها بعد از تغییرات مورد نیاز اگر ناسازگاری بین دامنه ها با دیکشنری داشتیم مقدار false برگردانده میشود در غیر این صورت تا خالی نشدن صف به ترتیب عناصر را بررسی و حذف میکنیم به این صورت که اگر طول دامنه باقی مانده پس از REVISE صفر بود که مقدار false است و بعد برای تمام عناصر در همسایه های xi به جز xj آن را به صف اضافه کرده و مقدار true برگردانده میشود.