



دانشکده‌ی مهندسی کامپیوتر
پروژه‌ی دوم مبانی و کاربردهای هوش مصنوعی

نام استاد : بهنام روشنفکر
نام دانشجو : آریان بوکانی
شماره دانشجویی : ۹۷۳۱۰۱۲

نیم‌سال اول ۹۹-۰۰

نحوه‌ی پیاده‌سازی

برای حل مسئله از کلاس CSP در فایل `csp.py` استفاده می‌شود. هر شی این کلاس دارای پارامترهای `variables` و `domains` هستند. `Variables` لیستی از متغیرها و `domains` نیز مقادیر `assign` شده به هر کدام از این متغیرها است که در قالب یک `dictionary` که کلیدهای آن متغیرها و مقادیر آن نیز یک `dictionary` دیگر است که اطلاعات رنگ و عدد اختصاص داده شده را ذخیره می‌کند، نگهداری می‌شود. سپس بر اساس مسئله محدودیت‌ها به هر کدام از متغیرهای درگیر محدودیت اضافه می‌شود و در لغت‌نامه‌ی `constraints` ذخیره می‌شود.

نحوه‌ی فرموله‌بندی

متغیرهای مسئله‌ی سودوکوی رنگی هر کدام از خانه‌های جدول هستند. بنابراین ما در کل $n * n$ تا متغیر برای مقداردهی هستیم. دامنه‌ی هر یک از این متغیرها شامل دو قسمت رنگ و عدد می‌باشد. در این صورت برای هر یک از متغیرها $n * m$ مقدار برای مقداردهی داریم. برای حل دو نوع محدودیت رنگی و عددی تعریف شده است که به هر کدام می‌پردازیم: در ابتدا یک کلاس `Constraint` تعریف می‌کنیم و یک تابع `satisfied` به آن نسبت می‌دهیم تا آن را توسط کلاس‌های فرزند `Override` کنیم. همچنین این کلاس متغیرهای درگیر در محدودیت را در لیست `variables` در خود ذخیره می‌کند. کلاس `NumberConstraint` از کلاس `Constraint` ارث می‌برد و برای ارضاشدن باید تمامی اعداد نسبت داده شده به متغیرهای آن از هم‌دیگر متفاوت باشند. کلاس `ColorConstraint` از کلاس `Constraint` ارث‌بری می‌کند و برای ارضا شدن باید رنگ‌های نسبت داده شده به دو متغیر آن با هم متفاوت باشند و اگر به هر دوی آنها عدد نسبت داده شده است بسته به عدد نسبت داده شده دارای اولویت رنگی باشند. (متغیر با مقدار عددی بالاتر باید رنگ با اولویت بیشتری نیز داشته باشد)

توضیح هیوریستیک

ابتدا میان متغیرهایی که هنوز به صورت کامل مقداردهی نشده‌اند (رنگ و عدد)، آنهایی که مقدار مجاز کمتری دارند انتخاب می‌شوند (MRV) سپس از میان این متغیرها، آنهایی که در محدودیت‌های بدون مقداردهی شده‌ی بیشتری شرکت دارند (degree) یکی برای مقداردهی انتخاب می‌شود.

توضیح قسمت‌هایی از کد

توسط تابع `readInputs` مقادیر ورودی حل مسئله خوانده می‌شوند. اگر به این تابع هیچ پارامتری پاس نشود، برنامه ورودی‌ها را از ترمینال می‌خواند. در غیر این صورت ورودی‌ها توسط فایل `text` پاس داده شده به این مسئله گرفته می‌شوند. پس از خواندن ورودی‌ها بر اساس متغیرها، محدودیت‌ها ساخته می‌شوند. برای هر سطر و هر ستون یک محدودیت عددی و برای هر سلول و سلول همسایه‌اش نیز یک محدودیت رنگی نسبت می‌دهیم. حال مقدارهای نسبت داده شده به هر سلول را در متغیر `assignment` ذخیره می‌کنیم. فرمت `domains` و `assignment` برای مثال به شکل زیر می‌باشد:

```
{(0, 0): {'color': 'g', 'number': 2}, (0, 1): {'color': 'r'}, ...}
```

مسئله زمانی حل شده است که برای همه‌ی سلول‌ها رنگ و عدد نسبت داده شده باشد و این مقادیر سازگار باشند. توسط تابع `selectVariable` که در `backtrack` استفاده شده است، متغیر بعدی برای مقداردهی انتخاب می‌شود. این تابع برای انتخاب متغیر بعدی از هیوریستیک MRV که متغیری که کمترین مقادیر باقیمانده در دامنه را انتخاب می‌کند و هیوریستیک `degree` که متغیری را انتخاب می‌کند که در بیشترین تعداد محدودیت‌های `assign` نشده شرکت دارد، متغیر بعدی را پیدا می‌کند. اولویت مقداردهی اول برای شماره دهی است و اگر شماره نسبت داده شده باشد، اینبار رنگ را برای مقدار دادن انتخاب می‌کند. سپس یک مقدار برای نسبت دادن انتخاب می‌شود و اگر مقدار نسبت داده شده سازگار باشد و تمام محدودیت‌ها را ارضا کند، این‌بار توسط الگوریتم `forward checking` تمام مقادیر ناسازگار موجود در دامنه‌ی متغیرهای دیگر حذف خواهد شد. اگر در این مرحله دامنه‌ی یکی از متغیرها تهی شود، مقدار `false` برگردانده می‌شود و `backtrack` انجام می‌شود. در صورتی که هیچ مقداردهی مناسبی برای متغیرها پیدا نشد، عبارتی در ترمینال چاپ می‌شود. در غیر این صورت این مقدارهای نسبت داده شده برای هر متغیر چاپ می‌شوند.

پی‌نوشت

مسئله توسط کتابخانه‌ی `pygame` به صورت گرافیکی روند حل مسئله را نشان می‌دهد. برای فعال کردن این قابلیت باید تابع `visualize` در فایل `main.py` را که در خط 82 قرار دارد از حالت کامنت در بیاورید.

چند نمونه از اجرای کد

Input:

```
3 3
r g b
## ## ##
## ## ##
## ## ##
```

Output:

```
1g 3r 2g
3r 2g 1b
2g 1b 3r
```

Input:

```
3 5
r g b
## ## ## ## ##
## ## ## ## ##
## ## ## ## ##
## ## ## ## ##
## ## ## ## ##
```

Output:

```
1g 3r 2g 5r 4g
3r 2g 4r 1g 5r
2b 5r 3g 4r 1g
4g 1b 5r 2g 3r
5r 4g 1b 3r 2g
```

Input:

```
5 6
r g b y p
1# *# *# *# *# *#
*# *# *# *# *# *#
*# *# *# *# *# *#
*# *# *p *# *# *#
*# *# *# *# *# *#
*# *# *# *# *# 6r
```

Output:

```
1p 4y 5r 3p 6r 2g
4b 5g 2p 6r 3y 1p
3y 1p 6r 2p 4g 5r
6g 2y 1p 4y 5r 3p
2p 6r 3y 5r 1p 4y
5r 3p 4r 1p 2y 6r
```