دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

درس هوش مصنوعی و کارگاه

گزارش ۴: پیاده سازی بازی Sudoku با استفاده از روش‌های ارضای محدودیت و روش‌های فرا ابتکاری پیشرفته آن

نگارش

کیارش مختاری دیزجی

۹۸۳۰۰۳۲

استاد اول

دکتر مهدی قطعی

استاد دوم

بهنام یوسفی مهر

اردیبهشت ۱۴۰۲

چکیده

در این پروژه با استفاده از الگوریتم Backtracking که یکی از الگوریتم‌های ارضای محدودیت می‌باشد، بازی Sudoku پیاده‌سازی شده است. برای بهبود سرعت حل جدول با اندازه بزرگ مثل ۱۶ و ۲۵ از الگوریتم‌های MRV و LCV و همچنین Forward checking استفاده شده است.

واژه‌های کلیدی:

روش ارضای محدودیت csp، Backtracking، Forward checking ، LCV، MRV

لینک پروژه:

https://github.com/Kiarashmo/AI-course/tree/main/Project%204

فهرست مطالب

صفحه

فهرست اشکال

صفحه

۱. فصل اول

مقدمه

۱-۱- توضیح مختصری از بازی سودوکو و قوانین آن

سودوکو جدول اعدادی است که یکی از بهترین تمرین‌های مغز و تقویت آی کیو است. نوع متداول سودوکو یک جدول ۹×۹ است که کل جدول هم به ۹ جدول کوچک‌تر ۳×۳ تقسیم شده‌است. در این جدول چند عدد به طور پیش فرض قرار داده شده که باید باقی اعداد را با رعایت سه قانون زیر یافت:

- قانون اول: در هر سطر جدول اعداد ۱ الی ۹ بدون تکرار قرار گیرد.
- قانون دوم: در هر ستون جدول اعداد ۱ الی ۹ بدون تکرار قرار گیرد.
- قانون سوم: در هر ناحیه ۳×۳ جدول اعداد ۱ الی ۹ بدون تکرار قرار گیرد.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

۱- نمونه‌ای از جدول سودوکو ۹x۹

۱-۲- الگوریتم Backtracking و MRV و LCV و Forward checking

Backtracking یک الگوریتم ساده و پرکاربرد مبتنی بر جستجو برای CSPها است. با انتخاب یک متغیر، اختصاص یک مقدار به آن، و سپس تلاش بازگشتی برای حل بقیه مسئله کار می‌کند. اگر با شکست مواجه شود، به متغیر قبلی بر می‌گردد و مقدار دیگری را امتحان می‌کند، تا زمانی که راه حلی پیدا کند یا تمام احتمالات را تمام کند. بازگشت به عقب را می‌توان با استفاده از روش‌های هیوریستیک، مانند MRV یا LCV، برای هدایت جستجو و کاهش تعداد بک ترک ها، بهبود بخشید.

- **MRV** مخفف Minimum Remaining Values است و به معنی یافتن متغیر با کمترین تعداد مقدار باقیمانده (unassigned values) است. با انتخاب متغیری که کمترین تعداد مقدار باقیمانده را دارد، احتمال رخ دادن نقض (contradiction) کاهش پیدا می‌کند.
- **LCV** مخفف Least Constraining Value است و به معنی یافتن مقداری است که کمترین تعداد محدودیت را بر روی متغیرهای دیگر اعمال می‌کند. برای استفاده از الگوریتم LCV در یک الگوریتمBacktracking ، در هر مرحله از جستجو، متغیری که کمترین تعداد مقدار باقیمانده را دارد، انتخاب می‌شود و سپس برای هر مقدار ممکن از آن متغیر، تعداد محدودیت‌هایی که بر روی متغیرهای دیگر اعمال می‌شود در صورت انتخاب آن مقدار، محاسبه می‌شود. سپس امتیاز کلی برای هر مقدار محاسبه می‌شود و مقداری که کمترین امتیاز را دارد، انتخاب می‌شود.
- **Forward Checking** پیش‌بینی می‌کند که انتخاب یک مقدار برای یک متغیر، چه تأثیری بر روی متغیرهای دیگر و بازهم چه تأثیری بر روی انتخاب‌های بعدی خواهد داشت. با انتخاب مقداری برای یک متغیر، فرض بر این است که مقدار انتخاب شده برای آن متغیر قابل قبول است. پس باید بررسی کرد که آیا انتخاب این مقدار با محدودیت‌هایی که بر روی متغیرهای دیگر اعمال شده، سازگاری دارد یا خیر. برای بررسی این موضوع، برای هر متغیر دیگری، مقادیری را که می‌تواند بگیرد، به دنبال تأثیر مستقیم انتخاب مقدار بر روی آن متغیر بررسی می‌شود. اگر یک مقدار برای یک متغیر باعث نقض محدودیت‌هایی بر روی یک یا چند متغیر دیگر شود، آن مقدار از لیست مقادیر ممکن برای آن متغیر حذف می‌شود.

۲. فصل دوم

پیاده سازی و تست بازی در پایتون

۲-۱- توابع بازی

```
def solve_sudoku(board):
```

این تابع همان الگوریتم backtrack می‌باشد که با استفاده از تابع find_empty ابتدا چک می‌کند مکان‌های خالی جدول در چه موقعیتی از جدول قرار دارند و سپس دامنه این مکان‌های خالی را با استفاده از تابع get_domain بدست آورده و با ایتریت کردن در مقادیر دامنه و صدا زدن تابع backtrack به صورت بازگشتی بدنبال مقدار مناسب برای متغیرهایمان می‌گردیم.

```
def forward_check(board, row, col, val):

def undo_forward_check(board, row, col, val):
```

این دوتابع که در داخل تابع solve_sudoku برای هرس کردن برخی مقادیر استفاده می‌شوند.در forward_check قیود انتساب داده شده متغییرهای باقی مانده منتشر می‌کند. اگر انتشار موفقیت آمیز باشد، True، در غیر این صورت False را برمی‌گرداند. تابع undo_forward_checking برعکس تابع forward_checking می‌باشد.

```
def is_valid_move(board, row, col, val):
```

این تابع برسی می‌کند که آیا قرار دادن مقدار داده شده در موقعیت داده شده معتبر است یا خیر. در اینجا الگوریتم LCV پیاده شده است.

```
def get_domain(board, row, col):
```

این تابع با چک کردن مقادیر ثابت در سطر و ستون و زیر مربع های جدول دامنه اعدادی که متغییر می‌تواند داشته باشد را برسی می‌کند.

```
def find_empty_cell(board):
```

این تابع متغییر با کمترین مقادیر باقیمانده در دامنه خود را پیدا می کند. که در اینجا الگوریتم MRV پیاده سازی شده است.

۲-۲- تست بازی

در ادامه یک نمونه از تست کد برای جدول ۹x۹ آمده است:

```
============= Unsolved Board =============
2 0 0 | 0 8 0 | 3 0 0
0 6 0 | 0 7 0 | 0 8 4
0 3 0 | 5 0 0 | 2 0 9
-------------------------------------------------------
0 0 0 | 1 0 5 | 4 0 8
0 0 0 | 0 0 0 | 0 0 0
4 0 2 | 7 0 6 | 0 0 0
-------------------------------------------------------
3 0 1 | 0 0 7 | 0 4 0
7 2 0 | 0 4 0 | 0 6 0
0 0 4 | 0 1 0 | 0 0 3
============= Solved Board =============
2 4 5 | 9 8 1 | 3 7 6
1 6 9 | 2 7 3 | 5 8 4
8 3 7 | 5 6 4 | 2 1 9
-------------------------------------------------------
9 7 6 | 1 2 5 | 4 3 8
5 1 3 | 4 9 8 | 6 2 7
4 8 2 | 7 3 6 | 9 5 1
-------------------------------------------------------
3 9 1 | 6 5 7 | 8 4 2
7 2 8 | 3 4 9 | 1 6 5
6 5 4 | 8 1 2 | 7 9 3
============= Time spent to solve =============
0.006997585296630859
```

و همچنین یک نمونه تست کد برای جدول ۱۶x۱۶:

```
============= Unsolved Board =============
 8 15 11  1 |  6  2 10  0 |  0  0 13  3 | 16  9  4  0
10  6  0  0 | 12  5  8  4 | 14 15  1  9 |  2 11  7 13
 0  5  9  0 |  0  0  0  0 |  8  0  0  4 |  0  0  0  6
 0 13  0  0 |  0  9  7  0 |  0 10  5 11 |  3  0  8 14
-------------------------------------------------------
 9  0  6  0 | 14  1  0  7 |  3  0 10  0 |  4  8  0  0
 3  0 12  8 |  2  4  6  9 |  0  0  7  0 | 10  0  0  0
11  0  5  0 |  0 12  3  0 |  1  9  0  0 |  7  6 14 16
 1  4  7  0 |  0 10  0  5 | 15  0  8  0 |  9  2  3 11
-------------------------------------------------------
 0  7  0  0 |  9  6  1  0 |  0  8  3  0 |  0 14  0  4
 0 12  0 11 |  7  0 14  0 |  5  4  6 15 |  0 13  9 10
 0  3  0  4 |  0  0  0  8 |  7  0  9  0 |  0  0  0  2
 0  1  0  9 |  4 11  5  0 |  0 16  0  0 |  8  3  6  0
-------------------------------------------------------
 0  0  4  0 |  0  7  0  0 |  0  0  0  0 |  0  5  0  0
 0  0 13  2 |  3  8  4  6 |  0  0 15  0 | 14  0 12  9
 7  9  0  6 | 15 14 12 11 |  0  3  2  5 | 13  4 10  8
 0 14  0 10 |  5 13  9  0 |  0 12  0  0 |  6  7  0  3
============= Solved Board =============
 8 15 11  1 |  6  2 10 14 | 12  7 13  3 | 16  9  4  5
10  6  3 16 | 12  5  8  4 | 14 15  1  9 |  2 11  7 13
14  5  9  7 | 11  3 15 13 |  8  2 16  4 | 12 10  1  6
 4 13  2 12 |  1  9  7 16 |  6 10  5 11 |  3 15  8 14
-------------------------------------------------------
 9  2  6 15 | 14  1 11  7 |  3  5 10 16 |  4  8 13 12
 3 16 12  8 |  2  4  6  9 | 11 14  7 13 | 10  1  5 15
11 10  5 13 |  8 12  3 15 |  1  9  4  2 |  7  6 14 16
 1  4  7 14 | 13 10 16  5 | 15  6  8 12 |  9  2  3 11
-------------------------------------------------------
13  7 16  5 |  9  6  1 12 |  2  8  3 10 | 11 14 15  4
 2 12  8 11 |  7 16 14  3 |  5  4  6 15 |  1 13  9 10
 6  3 14  4 | 10 15 13  8 |  7 11  9  1 |  5 12 16  2
15  1 10  9 |  4 11  5  2 | 13 16 12 14 |  8  3  6  7
-------------------------------------------------------
12  8  4  3 | 16  7  2 10 |  9 13 14  6 | 15  5 11  1
 5 11 13  2 |  3  8  4  6 | 10  1 15  7 | 14 16 12  9
 7  9  1  6 | 15 14 12 11 | 16  3  2  5 | 13  4 10  8
16 14 15 10 |  5 13  9  1 |  4 12 11  8 |  6  7  2  3
============= Time spent to solve =============
0.04099845886230469
```

فصل سوم

جمع‌بندی و نتیجه‌گیری

با توجه به تست‌های که در بخش قبل دیدیم می‌توان نتیجه گرفت که الگوریتم بک ترک با استفاده از الگوریتم‌های MRV و LCV و Forward checking به خوبی عمل کرده و در زمان بسیار کمی جدول‌های ۹ و ۱۶ تایی را حل کرده اما زمانی که جدول ۲۵ تایی به کد داده شد زمان مورد انتظار از ۱۰ ثانیه بیشتر شد و این احتمال می‌رود به دلیل استفاده از الگوریتم Forward checking می‌باشد زیرا این الگوریتم در مقادیر کم سرعت خوب اما در مقادیر بزرگ سرعت نسبتا کمی دارد و اگر از الگوریتم ARC-3 استفاده می‌شد ممکن بود جدول ۲۵ تایی نیز در زمان کمتر از ۱۰ ثانیه اجرا شود زیرا الگوریتم ARC-3 در مقادیر بزرگ بهتر عمل می کند.

منابع

https://www.linkedin.com/advice/0/what-pros-cons-using-backtracking-csps-skills-problemsolving#:~:text=What%20is%20backtracking%3F,the%20rest%20of%20the%20problem.