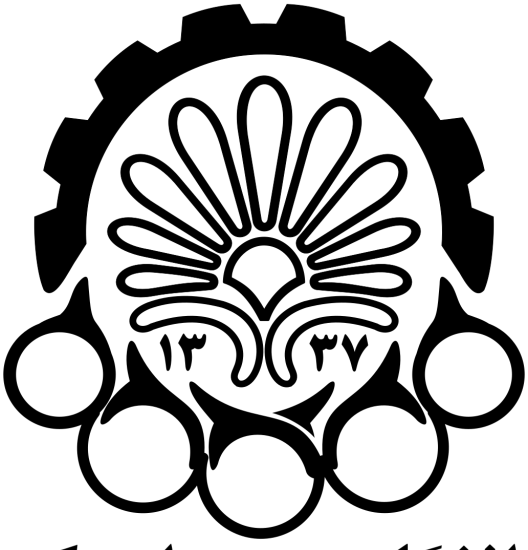


دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

گزارش پروژه‌ی سوم درس مبانی هوش مصنوعی

استاد روشن‌فکر

آرمین ذوالفقاری داریانی ۹۷۳۱۰۸۲ و امیرحسین رجب‌پور ۹۷۳۱۰۸۵

بهار ۱۴۰۰

توضیح کلی توابع و روند الگوریتم:

در ابتدا در فایل Main.py پازل و حالت انتشار محدودیت ورودی گرفته می‌شود و به فرم مورد نظر درمی‌آیند و سپس تابع start_CSP صدا زده می‌شود که آغاز الگوریتم است. در این تابع آرایه‌ی domains_list ساخته می‌شود که دامنه‌ی هر عنصر را در خود دارد و در مراحل بعدی تغییر می کند. سپس تابع CSP_Backtracking صدا زده می‌شود که الگوریتم back tracking را اجرا می‌کند و به صورت recursive در خودش صدا زده می‌شود تا به جواب برسیم و یا اینکه دامنه‌ی تمام عناصر تمام شود و جوابی پیدا نکنیم.

در این تابع ابتدا الگوریتم MRV صدا زده می‌شود و در این الگوریتم متغیری که کمترین تعداد عناصر را در دامنه‌اش دارد انتخاب می‌شود و یکی از مقادیر دامنه‌اش به آن اختصاص داده می‌شود و الگوریتم ادامه پیدا می کند. اگر در این مرحله متغیری که انتخاب کرده بودیم دامنه‌اش خالی شد آنگاه به والدش برمی گردیم و از آن جا با مقدار جدید والد برنامه را ادامه می‌دهیم.

بعد از اجرای MRV یکی از دو الگوریتم forward chaining و یا MAC اجرا می‌شود و اگر در این مرحله state = True برگردانده شد یعنی می‌توان به برنامه ادامه داد پس گره جدید ایجاد می‌شود و برنامه ادامه می‌یابد. در غیر این صورت باید مقدار دیگری برای متغیر انتخاب شده در گره در نظر گرفته شود و دوباره برنامه ادامه پیدا کند. آرایه‌ی domains_list نیز در این مرحله آپدیت می‌شود.

شبه کد الگوریتم **CSP_backtracking:**

```
CSP-BACKTRACKING(A)
1. If assignment A is complete then return A
2. X ← select a variable not in A
3. D ← select an ordering on the domain of X
4. For each value v in D do
    a. Add (X←v) to A
    b. If A is valid then
        i. result ← CSP-BACKTRACKING(A)
        ii. If result ≠ failure then return result
    c. Remove (X←v) from A
5. Return failure
```

توضیح ساختار Node:

برای هر حالت یک Node در نظر گرفته می‌شود که دارای صفحه‌ی بازی در آن حالت (board)، والد آن حالت (parent)، آرایه‌ی دامنه‌ها در آن مرحله (variable_domains)، متغیری که در آن مرحله تعیین می‌کند (assigned_variable) و مقدار آن متغیر (assigned_value).

```
self.board = board
self.parent = parent
self.variables_domain = variables_domain
self.assigned_variable = assigned_variable
self.assigned_value = assigned_value
```

توضیح GameRule:

در این فایل قوانین بازی پیاده‌سازی شده‌اند که همان موارد ذکر شده در دستور پروژه می‌باشند.

۱. هر سطر و هر ستون باید تعداد برابری صفر و یک داشته باشد.

۲. اعداد قرار گرفته در هر سطر و هر ستون باید یک رشته‌ی یکتا تولید کند.

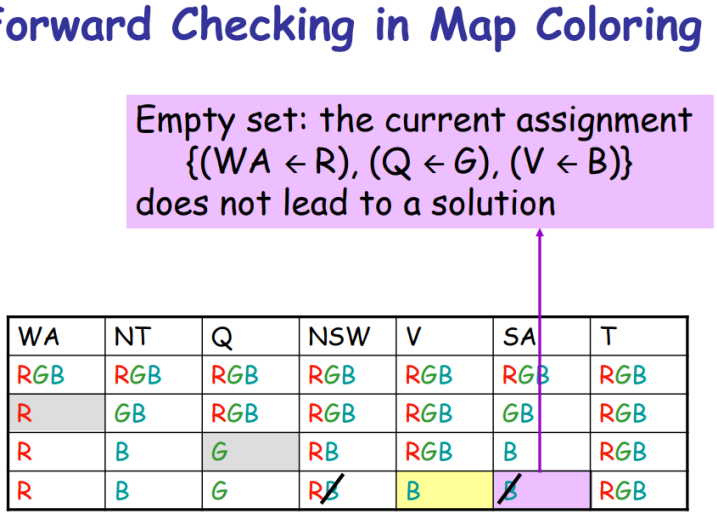
۳. در هر سطر و ستون نباید بیش از ۲ عدد تکراری پشت سر هم قرار بگیرد.

توضیح forward checking:

در این الگوریتم دامنه‌ی متغیرها گرفته می‌شود و سپس الگوریتم forward checking بر روی آن اجرا می‌شود و دامنه‌ها اصلاح می‌شوند و سپس دامنه‌های اصلاح شده به همراه یک فلگ برگردانده می‌شوند که اگر فلگ برابر False باشد به معنای این است که یکی از متغیرها

دامنه‌اش خالی شده‌است و نمی‌تواند مقداری داشته باشد. پس در الگوریتم backtracking در CSP مقدار دیگری برای متغیری که آخرین بار مقدار گرفته بود در نظر گرفته می‌شود و اگر مقداری برای آن متغیر باقی نمانده بود آن گاه به والد آن گره back track می‌کنیم.

در شکل زیر نمونه ی مشابهی از forward checking را می‌بینیم که هنگامی که دامنه‌ی یک متغیر خالی شود دیگر جوابی نداریم که الگوریتم ما در این حالت False برمی‌گرداند.



توضیح MAC:

در الگوریتم MAC نیز ورودی و خروجی‌ها دقیقا به صورت forward checking می‌باشند (جهت هماهنگی کد و استفاده‌ی بهینه‌تر در الگوریتم backtracking).

روال الگوریتم MAC بدین صورت می‌باشد که هر خانه‌ای که مقدارش عوض شد، همسایه‌های آن به یک صف اضافه می‌شوند و دوباره همسایه‌هایشان نیز بررسی می‌شود و در صورت تغییر به صف اضافه می‌شوند.

شرط پایان:

تابع check_all_rule_game نیز بررسی می کند که آیا به حالت پایانی رسیده‌ایم یا خیر (تمام خانه‌های پازل باید مقدار دهی شده باشند و تمام قوانین ذکر شده در پازل رعایت شده باشند)

مقایسه‌ی الگوریتم forward checking و MAC از لحاظ سرعت:

- برای پازل ۴:

```
Time analysis:
for constraint propagation forward-checking took 0.7917 secs long
```

```
Time analysis:
for constraint propagation MAC took 0.7624 secs long
```

- برای پازل شماره ۵:

```
Time analysis:
for constraint propagation forward-checking took 1.0639 secs long
```

```
Time analysis:
for constraint propagation MAC took 0.8901 secs long
```

- پازل ۶:

```
Time analysis:
for constraint propagation forward-checking took 0.0771 secs long
```

```
Time analysis:
for constraint propagation MAC took 0.0171 secs long
```

توضیح:

حال با بررسی پازل ها می‌توان به وضوح مشاهده کرد که الگوریتم forward checking کندتر از الگوریتم MAC عمل می‌کند.