

بسم الله الرحمن الرحيم



نام و نام خانوادگی : امیر محمد شفیعی

استاد مربوطه : جناب آقای دکتر منتطری

رنگ آمیزی نقشه ایران با استفاده از روش ارضای محدودیت‌ها (CSP)

درس : هوش مصنوعی پیشرفته

بهار 1404

چکیده

در این پروژه، مسئله رنگ‌آمیزی نقشه ایران با استفاده از روش‌های حل مسئله در هوش مصنوعی مورد بررسی قرار گرفته است. هدف این است که به هر استان رنگی اختصاص داده شود به‌طوری که هیچ دو استان مجاور رنگ یکسان نداشته باشند. این مسئله در قالب مسئله ارضای محدودیت‌ها (CSP) مدل‌سازی شده و با استفاده از الگوریتم بازگشتی Backtracking به همراه بهینه‌سازی‌هایی مانند بررسی رو به جلو (Forward Checking) و انتخاب متغیر بر اساس کمترین مقادیر باقی‌مانده (MRV) حل شده است. همچنین اثبات نظری امکان‌پذیری حل این مسئله با چهار رنگ با استناد به قضیه چهار رنگ ارائه شده است.

هدف پروژه

هدف از این پروژه، پیاده‌سازی یک مسئله کلاسیک در زمینه هوش مصنوعی است که در آن با استفاده از مفاهیم مربوط به ارضای محدودیت‌ها (Constraint Satisfaction Problem)، روشی برای حل یک مسئله واقعی یعنی رنگ‌آمیزی نقشه ارائه می‌شود. این پروژه علاوه بر پیاده‌سازی عملی، به درک بهتر مفاهیمی مانند دامنه، محدودیت، جستجو و بهینه‌سازی در مسائل CSP کمک می‌کند.

تعریف مسئله

نقشه ایران شامل تعدادی استان است که برخی از آن‌ها با یکدیگر مرز مشترک دارند. در این پروژه هدف این است که برای هر استان یک رنگ انتخاب شود به‌گونه‌ای که هیچ دو استان مجاور رنگ یکسان نداشته باشند. علاوه بر این، تعداد رنگ‌های مجاز باید محدود باشد و امکان استفاده از رنگ‌های نامحدود وجود ندارد.

مدل سازی مسئله (CSP)

این مسئله به صورت یک مسئله ارضای محدودیت (CSP) مدل سازی شده است. عناصر اصلی آن به این صورت هستند:

- **متغیرها:** استان های ایران
 - **دامنه ها:** مجموعه ای محدود از رنگ ها (مثل قرمز، آبی، سبز، زرد)
 - **محدودیت ها:** اگر دو استان هم مرز باشند، نباید رنگ یکسان داشته باشند.
- برای مثال، اگر استان های "تهران" و "البرز" هم مرز باشند، آنگاه نباید هر دو مثلاً رنگ قرمز را داشته باشند. این محدودیت ها به صورت دوتایی (Binary Constraints) تعریف می شوند.

حل نظری مسئله

از نظر تئوری، این مسئله زیرمجموعه ای از مسئله معروف رنگ آمیزی گراف است. هر استان به عنوان یک گره (Node) در گراف در نظر گرفته می شود و هر مرز مشترک بین دو استان یک یال (Edge) است.

طبق قضیه چهار رنگ، هر نقشه مسطح را می توان با حداکثر چهار رنگ رنگ آمیزی کرد، به طوری که هیچ دو منطقه مجاور رنگ یکسان نداشته باشند. بنابراین، از نظر تئوری نیز می توان نقشه ایران را با حداکثر چهار رنگ رنگ آمیزی کرد.

برای پیاده سازی، از الگوریتم **Backtracking** استفاده شده است که یک روش جستجوی

بازگشتی برای حل مسائل CSP است. برای بهبود عملکرد، از تکنیک های **Forward**

Checking (بررسی رو به جلو) برای حذف زود هنگام رنگ های ناسازگار، و **MRV** برای انتخاب هوشمند متغیر استفاده شده است.

حل مسئله به صورت دستی (مثال ساده شده)

فرض کنیم فقط با چند استان کار داریم:

• **Tehran** با **Alborz**، **Qom**، **Semnan** و **Mazandaran** هم مرز است.

• **Alborz** با **Qazvin** و **Mazandaran** هم مرز است.

• **Qom** با **Markazi** و **Isfahan** هم مرز است.

اکنون می خواهیم این استان ها را رنگ آمیزی کنیم:

1. **Tehran** را قرمز می کنیم.

2. **Alborz** نمی تواند قرمز باشد → آبی

3. **Qom** نیز با تهران هم مرز است → سبز

4. **Semnan** هم با تهران هم مرز است → آبی

5. **Mazandaran** با تهران و **Alborz** هم مرز است → سبز

6. **Qazvin** با **Alborz** هم مرز است → می تواند قرمز باشد

7. **Markazi** با **Qom** هم مرز است → می تواند آبی باشد

8. **Isfahan** با **Qom** و **Markazi** هم مرز است → سبز

و به همین ترتیب، با کمی دقت و دنبال کردن محدودیت‌ها می‌توان استان‌ها را رنگ‌آمیزی کرد بدون اینکه رنگ‌های تکراری در همسایه‌ها دیده شود.

کد نویسی با پایتون و خروجی آن :

در ابتدای کد، از سه کتابخانه اصلی استفاده شده:

- **matplotlib.pyplot** برای ترسیم نمودار و نقشه رنگ‌آمیزی شده.
- **networkx** برای ساخت گراف استان‌ها و ارتباطات بین آن‌ها.
- **matplotlib.use("TkAgg")** مشخص می‌کند که از **backend TkAgg** برای نمایش گراف استفاده شود (برای هماهنگی با محیط گرافیکی سیستم).

تعریف گراف استان‌های ایران:

در قسمت **iran_map** ما از یک دیکشنری استفاده کردیم برای ایجاد استان‌ها و استاد‌های مجاورشان. در این بخش، گرافی تعریف شده که هر کلید نمایانگر یک استان است و مقدار آن لیستی از استان‌های هم‌مرز با آن است. این ساختار نشان‌دهنده‌ی روابط مکانی بین استان‌ها است. تعریف رنگ‌ها و دامنه‌ها:

در قسمت بعد نیاز است که دامنه‌ها و محدودیت رنگ‌ها را مشخص کنیم.

- مجموعه‌ای از رنگ‌ها تعریف شده که قرار است به استان‌ها اختصاص داده شوند.
- مجموعه کامل استان‌ها استخراج شده است، چون بعضی استان‌ها ممکن است فقط در لیست همسایگان آمده باشند.

- دامنه‌ی هر استان (یعنی رنگ‌های مجاز برای آن) برابر با لیست کامل رنگ‌ها در ابتدا در نظر گرفته شده.

بررسی معتبر بودن رنگ:

در تابع `is_valid_coloring` بررسی می‌کند که آیا می‌توان رنگ مورد نظر را به استان داده‌شده نسبت داد یا نه. اگر یکی از استان‌های هم‌مرز همین رنگ را داشته باشد، تابع `False` برمی‌گرداند. انتخاب متغیر بعدی:

در تابع `select_unassigned_variable` از **Heuristic انتخاب MRV (Minimum Remaining Values)** استفاده شده. یعنی استان‌هایی که تعداد گزینه‌های رنگی کمتری دارند، زودتر انتخاب می‌شوند.

پیاده‌سازی `forward checking`:

بعد از انتخاب یک رنگ برای یک استان، این تابع رنگ انتخاب‌شده را از دامنه‌ی استان‌های هم‌مرز حذف می‌کند. همچنین برای بازگردانی دامنه‌ها در مراحل بعدی، وضعیت قبلی را در یک دیکشنری ذخیره می‌کند.

بازگرداندن دامنه‌ها:

برای مواقعی که مسیر حل به بن‌بست می‌رسد و باید به مرحله‌ی قبل برگردیم، از این تابع استفاده می‌شود تا دامنه‌ی استان‌ها به حالت اولیه بازگردد. تابع `restore_domain` برای این حالت نوشته شده

الگوریتم Backtracking :

تابع `backtracking` را ایجاد کردیم.

تابع اصلی جستجوی عقب‌گرد است:

1. اگر تمام استان‌ها رنگ گرفته باشند، راه‌حل پیدا شده و باز می‌گردد.
2. یک استان انتخاب می‌شود با استفاده از `MRV`.
3. روی رنگ‌های مجاز آن استان پیمایش می‌شود.
4. اگر رنگ معتبر بود، آن را موقتاً به استان اختصاص می‌دهیم.
5. با حذف رنگ از دامنه‌ی همسایگان (`forward checking`) مسیر را ادامه می‌دهیم.
6. در صورت عدم موفقیت، به حالت قبلی برمی‌گردیم و رنگ دیگری را امتحان می‌کنیم.

ترسیم نقشه رنگ‌آمیزی شده:

در تابع `draw_colored_map` داریم این موارد را اجرا می‌کنیم :

1. یک گراف از استان‌ها ساخته می‌شود.
2. با استفاده از `networkx` و `matplotlib` گراف ترسیم می‌شود.
3. رنگ هر استان از `solution` گرفته می‌شود.
4. نقشه نهایی به صورت فایل `PNG` ذخیره می‌شود.

کد ها :

```
import matplotlib.pyplot as plt
import networkx as nx
import matplotlib
matplotlib.use("TkAgg")

iran_map = {
    "Tehran": ["Alborz", "Qom", "Semnan", "Mazandaran", "Markazi"],
    "Alborz": ["Tehran", "Mazandaran", "Qazvin"],
    "Qom": ["Tehran", "Markazi", "Isfahan"],
    "Semnan": ["Tehran", "Mazandaran", "Golestan", "Razavi Khorasan", "Isfahan"],
    "Mazandaran": ["Tehran", "Alborz", "Semnan", "Golestan"],
    "Markazi": ["Tehran", "Qom", "Isfahan", "Lorestan", "Hamedan", "Qazvin"],
    "Qazvin": ["Alborz", "Mazandaran", "Hamedan", "Zanjan", "Markazi"],
    "Isfahan": ["Qom", "Semnan", "Yazd", "South Khorasan", "Chaharmahal", "Kohgiluyeh", "Lorestan", "Markazi"],
    "Golestan": ["Mazandaran", "Semnan", "Razavi Khorasan"],
    "Razavi Khorasan": ["Golestan", "Semnan", "South Khorasan"],
    "South Khorasan": ["Razavi Khorasan", "Yazd", "Isfahan"],
}

colors = ["red", "blue", "green", "yellow"]
all_provinces = set(iran_map.keys()) | {neighbor for neighbors in iran_map.values() for neighbor in neighbors}
domain = {province: list(colors) for province in all_provinces}

def is_valid_coloring(province, color, assignment):
    for neighbor in iran_map.get(province, []):
        if neighbor in assignment and assignment[neighbor] == color:
            return False
    return True

def select_unassigned_variable(assignment):
    unassigned = [v for v in iran_map if v not in assignment]
    return min(unassigned, key=lambda var: len(domain[var]))

def forward_checking(province, color):
```



```

def forward_checking(province, color):
    temp = {}
    for neighbor in iran_map.get(province, []):
        if color in domain[neighbor]:
            temp[neighbor] = domain[neighbor][:]
            domain[neighbor].remove(color)
    return temp

def restore_domain(temp):
    for neighbor, values in temp.items():
        domain[neighbor] = values

def backtracking(assignment):
    if len(assignment) == len(iran_map):
        return assignment
    province = select_unassigned_variable(assignment)
    for color in domain[province]:
        if is_valid_coloring(province, color, assignment):
            assignment[province] = color
            temp = forward_checking(province, color)
            result = backtracking(assignment)
            if result:
                return result
            restore_domain(temp)
            del assignment[province]
    return None

solution = backtracking({})
if solution:
    print("حل پیدا شد:", solution)
else:
    print("هیچ راه حلی یافت نشد!")

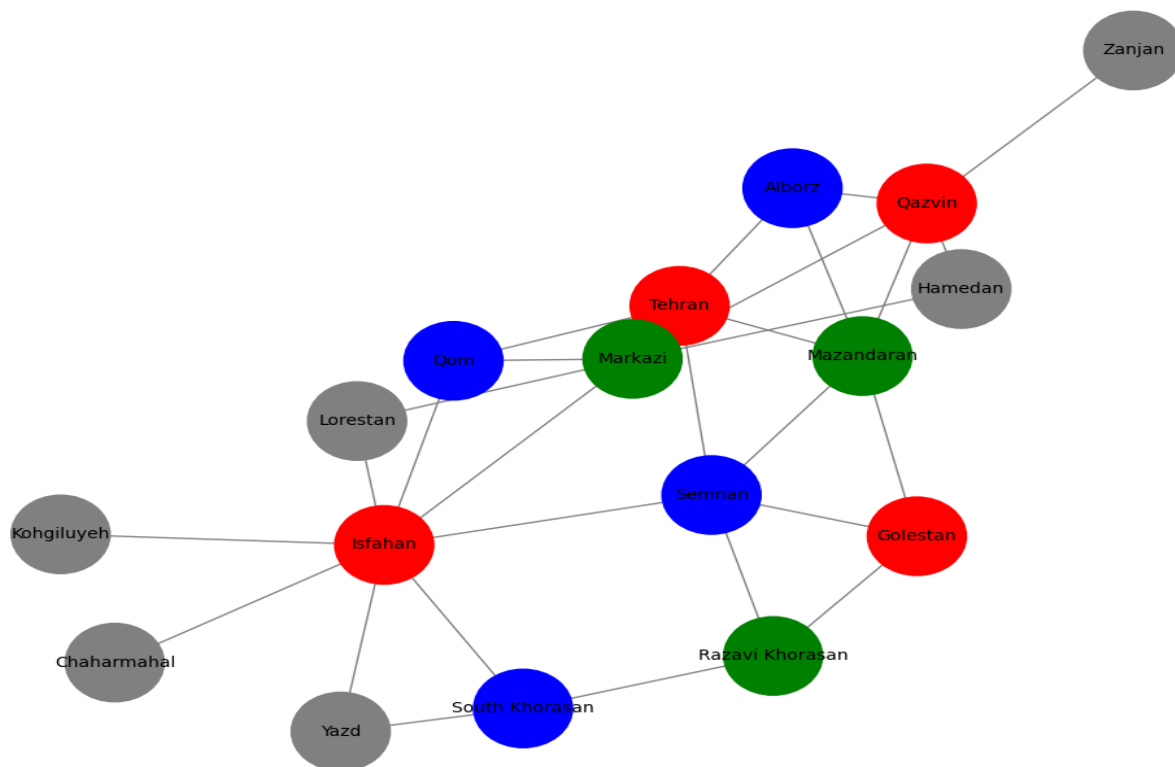
```

```

        result = backtracking(assignment)
        if result:
            return result
        restore_domain(temp)
        del assignment[province]
    return None
solution = backtracking({})
if solution:
    print("حل پیدا شد:", solution)
else:
    print("هیچ راه حلی یافت نشد!")
def draw_colored_map(solution):
    G = nx.Graph()
    for province, neighbors in iran_map.items():
        for neighbor in neighbors:
            G.add_edge(province, neighbor)
    plt.figure(figsize=(10, 8))
    pos = nx.spring_layout(G, seed=42)
    colors_map = [solution.get(node, "gray") for node in G.nodes()]
    nx.draw(G, pos, with_labels=True, node_color=colors_map, node_size=3000, edge_color="gray", font_size=10)
    plt.savefig("iran_map_coloring.png")
    print("📍 نقشه رنگ آمیزی شده ذخیره شد به نام 'iran_map_coloring.png'")
draw_colored_map(solution)

```

توضیح خروجی برنامه:



پس از اجرای برنامه، الگوریتم جستجوی عقب‌گرد همراه با بررسی پیش‌رو (**Forward Checking**) با موفقیت رنگ‌هایی را به استان‌های مختلف نسبت می‌دهد، به گونه‌ای که هیچ دو استان هم‌مرزی رنگ یکسان ندارند. این تخصیص رنگ‌ها به صورت یک دیکشنری (یا جدول) در خروجی چاپ می‌شود که در آن، کلید نام استان و مقدار مربوط به آن رنگ تخصیص داده شده است.

برای مثال، خروجی متنی برنامه ممکن است به صورت زیر باشد:

حل پیدا شد: }

, 'Tehran': 'red'

, 'Alborz': 'blue'

, 'Qom': 'green'

, 'Semnan': 'yellow'

...

{

این خروجی نشان می‌دهد که الگوریتم توانسته با استفاده از فقط چهار رنگ، مسئله رنگ‌آمیزی را به درستی و بدون تضاد حل کند.

همچنین، یک خروجی **گرافیکی** نیز تولید می‌شود که در آن نقشه‌ی گرافی استان‌ها نمایش داده شده و هر استان با رنگ اختصاص داده شده خود نشان داده می‌شود. این تصویر به صورت خودکار در مسیر اجرای برنامه با نام: `iran_map_coloring.png` ذخیره می‌شود.

در این تصویر:

- گره‌ها **Node** ها نمایانگر استان‌ها هستند.
- یال‌ها **Edge** ها نشان‌دهنده‌ی ارتباط مرزی بین استان‌ها هستند.

- رنگ هر گره همان رنگ تخصیص داده شده به استان توسط الگوریتم است.

این خروجی تصویری به وضوح نشان می دهد که هیچ دو گره ای که با یک یال به هم متصل اند (یعنی هم مرز هستند)، رنگ یکسان ندارند. بنابراین، مسئله با موفقیت حل شده و تمام محدودیت ها رعایت شده اند.

با تشکر فراوان از جناب آقای دکتر منظتری

امیر محمد شفیعی