بسم الله الرحمن الرحیم

نام و نام خانوادگی : امیرمحمد شفیعی

شماره دانشجویی : 40307314

استاد مربوطه : جناب آقای دکتر منتطری

رنگ‌آمیزی نقشه ایران با استفاده از روش ارضای محدودیت‌ها(CSP)

درس : هوش مصنوعی پیشرفته

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 مجاوری 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 CSPکمک 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 مجاوری رنگ یکسان نداشته باشند. علاوه بر این، تعداد رنگ‌های مجاز باید محدود باشد و امکان استفاده از رنگ‌های نامحدود وجود ندارد.

مدل‌سازی مسئله(CSP)

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

- **متغیرها :**استان‌های ایران
- **دامنه‌ها :**مجموعه‌ای محدود از رنگ‌ها (مثل قرمز، آبی، سبز، زرد)
- **محدودیت‌ها :**اگر دو استان هم‌مرز باشند، نباید رنگ یکسان داشته باشند.

برای مثال، اگر استان‌های "تهران" و "البرز" هم‌مرز باشند، آنگاه نباید هر دو مثلاً رنگ قرمز را داشته باشند. این محدودیت‌ها به‌صورت دوتایی (Binary Constraints) تعریف می‌شوند.

حل نظری مسئله

از نظر تئوری، این مسئله زیرمجموعه‌ای از مسئله معروف رنگ‌آمیزی گراف است. هر استان به‌عنوان یک گره (Node) در گراف در نظر گرفته می‌شود و هر مرز مشترک بین دو استان یک یال (Edge) است.

طبق **قضیه چهار رنگ**، هر نقشه مسطح را می‌توان با حداکثر **چهار رنگ** رنگ‌آمیزی کرد، به‌طوری‌که هیچ دو منطقه مجاوری رنگ یکسان نداشته باشند. بنابراین، از نظر تئوری نیز می‌توان نقشه ایران را با حداکثر چهار رنگ رنگ‌آمیزی کرد.

برای پیاده‌سازی، از الگوریتم **Backtracking**استفاده شده است که یک روش جستجوی بازگشتی برای حل مسائل CSP است. برای بهبود عملکرد، از تکنیک‌های **Forward**

Checking(بررسی رو به جلو) برای حذف زودهنگام رنگ‌های ناسازگار، و **MRV**برای انتخاب هوشمند متغیر استفاده شده است.

حل مسئله به‌صورت دستی (مثال ساده‌شده)

فرض کنیم فقط با چند استان کار داریم:

- Tehranبا Alborz ، Qom، Semnanو Mazandaran هم‌مرز است.
- Alborzبا Qazvin و Mazandaran هم‌مرز است.
- Qomبا Markazi و Isfahan هم‌مرز است.

اکنون می‌خواهیم این استان‌ها را رنگ‌آمیزی کنیم:

1. **Tehran**را قرمز می‌کنیم.
2. **Alborz**نمی‌تواند قرمز باشد ← آبی
3. **Qom**نیز با تهران هم‌مرز است ← سبز
4. **Semnan**هم با تهران هم‌مرز است ← آبی
5. **Mazandaran**با تهران و Alborz هم‌مرز است ← سبز
6. **Qazvin**با Alborz هم‌مرز است ← می‌تواند قرمز باشد
7. **Markazi**با Qom هم‌مرز است ← می‌تواند آبی باشد
8. **Isfahan**با Qom و Markazi هم‌مرز 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 استافده کردیم برای ایجاد استان ها و استاد های 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 دامنه ها و محدودیت رنگ ها 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 Falseبرمی‌گرداند.

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.

پیاده‌سازیforward 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

الگوریتمBacktracking :

تابع backtracking را ایجاد کردیم.

تابع اصلی جستجوی عقب‌گرد است:

1. اگر تمام استان‌ها رنگ گرفته باشند، راه‌حل پیدا شده و بازمی‌گردد.
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 رنگ‌آمیزی‌شده:

در تابع draw_colored_map داریم این موارد را اجرا میکنیم :

1. یک گراف از استان‌ها ساخته می‌شود.
2. با استفاده از networkxو matplotlibگراف ترسیم می‌شود.
3. رنگ هر استان از solutionگرفته می‌شود.
4. نقشه نهایی به صورت فایل PNG ذخیره می‌شود.

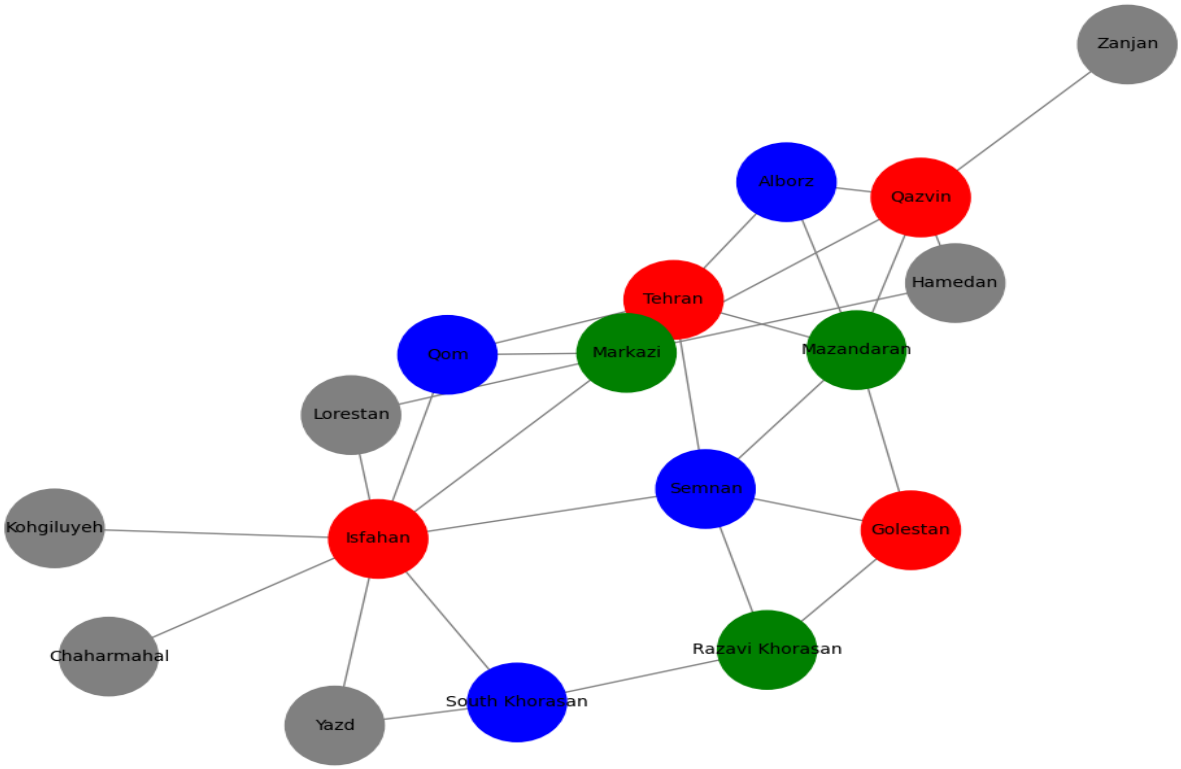
کد ها :

```python
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 راه‌حلی 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 راه‌حلی 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("'iran_map_coloring.png' نقشه رنگ‌آمیزی‌شده ذخیره شد به نام 📌")
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، به‌گونه‌ای که هیچ دو استان هم‌مرزی رنگ یکسان ندارند. این تخصیص رنگ‌ها به‌صورت یک دیکشنری (یا جدول) در خروجی چاپ می‌شود که در آن، کلید نام استان و مقدار مربوط به آن رنگ تخصیص داده‌شده 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 به‌درستی 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 داده‌شده خود نشان داده می‌شود. این تصویر به‌صورت خودکار در مسیر اجرای برنامه با نام: iran_map_coloring.png ذخیره می‌شود.

در این تصویر:

- گره‌هاNode ها نمایانگر استان‌ها هستند.
- یال‌هاEdge ها نشان‌دهنده‌ی ارتباط مرزی بین استان‌ها هستند.

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

این خروجی تصویری به وضوح نشان می‌دهد که هیچ دو گره‌ای که با یک یال به هم متصل‌اند (یعنی هم‌مرز هستند)، رنگ یکسان ندارند. بنابراین، مسئله با موفقیت حل شده و تمام محدودیت‌ها رعایت شده‌اند.

با تشکر فراوان از جناب آقای دکتر منظتری

امیرمحمد شفیعی