BỘ XÂY DỰNG
TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
THÀNH PHỐ HỒ CHÍ MINH

----📖----

Báo cáo

Môn: Trí tuệ nhân tạo

Đề tài: CSP Vietnam Map Coloring

Giảng viên: Nguyễn Văn Diêu
Lớp: CN2305CLCB
Mã học phần: 010412103301
Họ và Tên: Lê Đức Huy
MSSV: 075205009693

TP.Hồ Chí Minh, ngày 12 tháng 6 năm 2025

Mục lục

LỜI CẢM ƠN

Trước tiên, chúng tôi xin bày tỏ lòng biết ơn sâu sắc đến giảng viên Nguyễn Văn Diêu, người đã tận tình hướng dẫn, đóng góp ý kiến và giúp đỡ chúng tôi trong suốt quá trình thực hiện báo cáo này. Sự chỉ dạy quý báu của thầy đã giúp chúng tôi có nhiều tài liệu bổ ích, và kiến thức để có thể làm nên bài báo cáo này.

Chúng xin chân thành cảm ơn Viện đào tạo chất lượng cao đã hỗ trợ cung cấp cho chúng tôi chỗ để học tập và nguồn tài liệu bổ ích để nghiên cứu.

Chúng tôi cũng xin gửi lời cảm ơn đến các tác giả, nhà nghiên cứu và nguồn tài liệu tham khảo đã cung cấp những thông tin hữu ích, giúp chúng tôi có cơ sở để phát triển nội dung báo cáo. Những dữ liệu và kiến thức thu thập được là nền tảng quan trọng để chúng tôi hoàn thiện nghiên cứu này.

Bên cạnh đó, chúng tôi chân thành cảm ơn bạn bè và gia đình, những người đã luôn ủng hộ, động viên chúng tôi trong suốt quá trình thực hiện báo cáo. Sự khích lệ từ mọi người chính là nguồn động lực to lớn giúp chúng tôi vượt qua những khó khăn và hoàn thành công việc.

Cuối cùng, chúng tôi hy vọng rằng báo cáo này sẽ mang lại giá trị thực tiễn, góp phần giúp mọi người hiểu rõ hơn về tầm quan trọng của trí tuệ nhân tạo trong lĩnh vực quản lý tài chính cá nhân.

Chương 1. Mở đầu

I. Lời giới thiệu

Tô màu bản đồ là một bài toán cổ điển trong trí tuệ nhân tạo (AI) và tính toán ràng buộc (Constraint Satisfaction Problem).

Trong thời gian qua nước Việt Nam ta đã có sự tinh chỉnh về địa phận quản lý. Tinh giảm 63 tỉnh thành thành 34 tỉnh thành trên mọi miền của tổ quốc. Bài toán đặt ra là liệu nước ta cần chỉ 3 màu là đủ tô hết mọi tỉnh trên bản đồ hay chỉ cần tối thiểu 4 màu theo định lý bốn màu của Appel và Haken (1976).

II. Mục tiêu báo cáo

Học hỏi và áp dụng các lý thuyết về đồ thị, thỏa mãn ràng buộc và mô hình MiniZinc để giải quyết bài toán. Có kết hợp thư viện networkx, matplotlib để vẽ bản đồ Việt Nam 34 tỉnh thành dưới dạng đồ thị.

Đưa ra lời khẳng định cho bài toàn tô màu bản đồ Việt Nam.

Chương 2. Lý thuyết

I. Lý thuyết đồ thị

1. Khái niệm

Một đồ thị là một cấu trúc rời rạc gồm tập hợp các đỉnh và các cạnh nối giữa các đỉnh đó. Có thể mô tả đồ thị theo cách hình thức như sau:

$$G=(V, E)$$

Trong đó:

G: đồ thị.

V: tập đỉnh.

E: các cạnh, E là tập hợp các cặp (u, v) trong đó u và v là hai đỉnh thuộc V.

Đồ thị G có những phân loại như sau:

- G được gọi là **đơn đồ thị** nếu như giữa hai đỉnh (u,v) của V có nhiều nhất một cạnh trong E nối từ u tới v.
- G được gọi là **đa đồ thị** nếu như giữa hai đỉnh (u,v) của V có nhiều hơn 1 cạnh trong E nối từ u tới v.
- G được gọi là **đồ thị có hướng** nếu như các cạnh trong E là có định hướng, tức là có thể tồn tại cạnh nối từ u tới v nhưng chưa chắc đã tồn tại cạnh nối từ v tới u.
- G được gọi là **đồ thị vô hướng** nếu như các trong E là không định hướng.

2. Cách biểu diễn đồ thị

2.1. Ma trận kề

Đặc điểm:

- Ma trận vuông kích thước n x n (với là n là số đỉnh).
- Phần tử A[i][j] = 1 nếu có cạnh từ đỉnh i đến j, và ngược lại là 0.
- Nếu đồ thị có trọng số, A[i][j] chứa trọng số thay vì 1.

Ví dụ (đồ thị vô hướng, 3 đỉnh, 2 cạnh):

0 1 0

1 0 1

0 1 0

2.2. Danh sách kề

Đặc điểm:

- Mỗi đỉnh lưu danh sách các đỉnh kề với n.
- Nếu có trọng số, mỗi phần từ trong danh sách là cặp (đỉnh kề, trọng số).

Ví dụ (Đồ thị vô hướng, 3 đỉnh, 2 cạnh)

1: 2

2: 1 3

3: 2

2.3. Danh sách cạnh

Đặc điểm:

- Ghi lại tất cả các cạnh của đồ thị dưới dạng danh sách.
- Mỗi cạnh được biểu điễn như một cặp (u, v) hoặc bộ ba (u, v, w) trong đó w là trọng số.

Ví dụ:

(1, 2)

(2, 3)

II. Constraint Satisfaction Problem (CSP)

1. Khái niệm

Một CSP được xác định bởi 3 thành phần:

- **X** = $\{X_1, X_2, ..., X_n\}$ được gọi là tập các biến.
- **D** = $\{D_1, D_2, ..., D_n\}$ được gọi là tập các miền giá trị. Trong đó mỗi D_i , là miền giá trị của X_i.
- **C** = $\{C_1, C_2, ..., C_k\}$: được gọi là các ràng buộc. Trong đó mỗi C_i =(phạm vị, quan hệ).

Ví dụ:

$X = \{X_1, X_2\}$, $D_1, D_2 = \{1, 2, 3\}$

$C_1 = \{ (X_1, X_2), X_1 > X_2 \}$

Một **lời giải** của CSP là một gán giá trị cho mỗi biến sao cho **tất cả các ràng buộc đều được thỏa mãn**.

2. Các kiểu ràng buộc

Biến gồm 2 loại như sau:

- Tập hợp các miền giá trị giới hạn, với n biến và kích thước miền giá trị d thì số lượng các phép gán là $O(d^n)$.
- Tập hợp các miền giá trị vô hạn

Ràng buộc gồm 3 loại như sau:

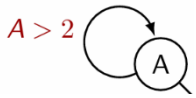
- Unary constraint (ràng buộc đơn) là ràng buộc chỉ áp dụng lên một biến duy nhất.
  ví dụ: C = { (SA), SA != green }
- Binary constranit (ràng buộc nhị phân) là ràng buộc áp dụng lên hai biến cùng lúc.
  ví dụ: C = { (SA, WA), SA != WA}

- Global constraint (ràng buộc toàn cục) là ràng buộc áp dụng cho một tập hợp
  ví dụ: AllDifferent(X_1, ..., X_n)

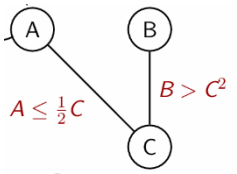
3. Constraint graph

Constraint graph được gọi là đồ thị ràng buộc, trong đó:

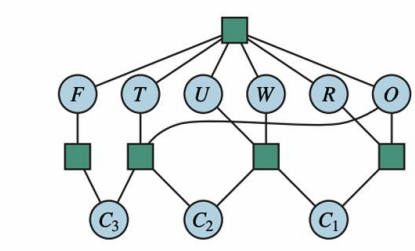
- Tập các biến là tập các đỉnh
- Tập các ràng buộc là tập các cạnh
  - Unary: ràng buộc áp dụng lên một biến duy nhất
  Ví dụ: C = (A, A > 2)



  - Binary: ràng buộc áp dụng lên hai biến
  Ví dụ C_1 = { (C, B), $C^2 > B$}, C_2 = { (C, A), 1/2C < A}



  - n-ary: ràng buộc áp dụng giữa n biến (n>=3)



4. Biểu diễn bài toán

Bài toán được định nghĩa như sau:

- X = { HN, DN, HCM,… LD }: Tổng 34 biến tương ứng với 34 tỉnh thành.
- Di = { đỏ, xanh, váng, hồng }: Giá trị màu của từng tỉnh
- C = { HN != PT, HN != TN,… HCM != DN}: Các ràng giữa các tỉnh chung biển không bị trùng màu.

III. Định lý bốn màu

Định lý bốn màu là một kết quả nổi tiếng trong toán học tổ hợp và lý thuyết đồ thị, phát biểu ràng:

- Bất kỳ bản đồ phẳng nào cũng có thể được tô màu bằng **tối đa 4 màu** sao cho không có hai vùng kề nhau nào có cùng màu.

Chương 3. Thực hành

I. Mô hình MiniZinc

MiniZinc là ngôn ngữ mô hình hóa ràng buộc cấp cao cho phép bạn dễ dàng diễn đạt và giải quyết các vấn đề tối ưu hóa rời rạc.

Bài toán tô màu bản đồ Việt Nam dưới góc nhìn của mô hình MiniZinc như sau (VNColoring.mzn):

```
int: color = 3;

array[1..34] of var 0..color: TinhThanh;

constraint TinhThanh[1] != TinhThanh[19];
constraint TinhThanh[1] != TinhThanh[17];
constraint TinhThanh[1] != TinhThanh[16];
...
constraint TinhThanh[34] != TinhThanh[33];

solve satisfy;
```

Trong Minizinc, ta định nghĩa:

- `int: color = 3;` là khai báo hằng số với số màu là 3.
  - `array[1..34] of var 0..color: TinhThanh;` là mảng khai báo 34 tỉnh thành, trong đó mỗi `TinhThanh[i]` có miền giá trị là `var 0..color` .
  - `constraint TinhThanh[1] != TinhThanh[16];` là ràng buộc giữa tỉnh 1 khác màu với tỉnh 16, tương tự với các dòng ràng buộc khác.
  - `solve satisfy;` là giải bài toán.

Nhấn run để thực hiện giải bài toán:

- =====UNSATISFIABLE===== là bài toàn không có giá trị thỏa mãn.
- Nếu thỏa mãn thì mô hình sẽ xuất những biến có giá trị hợp lý.

Sau khi chạy VNColoring.mzn:
- Ta thấy với giá trị miền số màu là 3 thì mô hình đưa ra kết quả =====UNSATISFIABLE===== nghĩa là không thể tô màu bản đồ Việt Nam với 3 màu.
- Khi thay đổi giá trí miền số màu là 4 thì mô hình đưa ra kết quả TinhThanh = [3, 1, 0, 2, 1, 2, 3, 2, 0, 1, 0, 3, 2, 1, 0, 0, 2, 1, 1, 0, 0, 1, 0, 2, 0, 1, 3, 2, 1, 1, 2, 0, 0, 2]; Nghĩa là có thể tô màu bản đồ Việt Nam với 4 màu.

II. Bài toán dưới góc nhìn lý thuyết đồ thị

1. Tổng quan

Bằng việc phối các mô hình MiniZinc với ngôn lập trình Python kết hợp sử dụng thư viện NetworkX và matplotlib, tôi sẽ vẽ trực quan ra được bản đồ Việt Nam dưới góc lý thuyết đô thị theo kết quả tô màu của MiniZinc.

Mã giả giải và vẽ bài toán như sau:

```
Gọi hàm giải CSP từ file “VNColoring.mzn”, lưu vào VNColor

If VNColor thỏa mãn:
     -Gọi hàm input_graph để đọc dữ liệu đồ thị Việt Nam
từ file “map.csv”, lưu vào biến graph
     -Gọi hàm draw để vẽ đồ thị
Else (không thỏa mãn):
     Print(“Không thể tô màu bản đồ 34 tỉnh thành Việt
Nam”)

Hàm GiaiCSP(namefile):
     -Tải mô hình MiniZinc từ file namefile
     -Gọi phương thức solve để giải bài toán
     -Trả về kết quả
```

Hàm NhapDoThi(namefile):
    -Mở file CSV ở chế độ đọc dữ liệu từ namefile
    -Tạo đồ thị g rỗng
    -Với mỗi dòng trong file:
      -Thêm cạnh theo danh sách kề
    -Trả về đồ thị g

Hàm VeDoThi(csp, graph):
    -csp: lời giải từ MiniZinc (chứa danh sách các màu ứng với mỗi tỉnh)
    -graph: đồ thị các tỉnh Việt Nam
    -Định nghĩa danh sách màu

    -Với mỗi đỉnh trong đồ thị:
       - Lấy giá trị màu từ lời giải CSP
       - Gán màu cho đỉnh tương ứng

    -Vẽ đồ thị bằng networkx và matplotlib với các màu đã gán

2. Chi tiết

```
import csv
import networkx as nx
import matplotlib.pyplot as plt
import minizinc

def solve_csp(namefile:str):
    model = minizinc.Model(namefile)
    gecode = minizinc.Solver.lookup("gecode")
    instance = minizinc.Instance(gecode, model)
    result = instance.solve()
    return result



def input_graph(namefile:str):
    with open(namefile, 'r', encoding='utf-8') as f:
        reader = csv.reader(f)
        g = nx.Graph()

        for row in reader:
            edge = row[0].split()
            node_a = int(edge[0])
            node_b = int(edge[1])
            g.add_edge(node_a, node_b)
        return g

def draw(csp, graph):
    color_list = ['red', 'blue', 'green', 'yellow']
    node_colors = [color_list[csp["TinhThanh"][i - 1]] for i in graph.nodes()]
    nx.draw(graph, with_labels=True, node_color=node_colors, node_size=200, font_size=10)
    plt.show()

VNColor = solve_csp("VNColoring.mzn")

if str(VNColor) != "None":
    Graph = input_graph('map.csv')
    draw(VNColor, Graph)
else:
    print("Không thể vẽ với bài toán chưa thỏa mãn")
```

Giải thích đoạn mã:

```
import csv
import networkx as nx
import matplotlib.pyplot as plt
import minizinc
```

- csv: đọc file csv.
- networkx: thư viện xử lý và biểu diễn đồ thị.
- matplotlib.pyplot: vẽ đồ thị.
- minizinc: gọi mô hình minizinc để giải bài toán ràng buộc.

```
def solve_csp(namefile:str):
    model = minizinc.Model(namefile)
    gecode = minizinc.Solver.lookup("gecode")
    instance = minizinc.Instance(gecode, model)
    result = instance.solve()
    return result
```

Chi tiết:

```
model = minizinc.Model("VNColoring.mzn")
```

- Tạo đối tượng Model từ thư viện minizinc với file khởi chạy là “VNColoring.mzn”.

```
gecode = minizinc.Solver.lookup("gecode")
```

- Sử dụng trình giải 'gecode'.

```
instance = minizinc.Instance(gecode, model)
```

- Instance tạo phiên bản cụ thể để kết nối model và solver.

```
result = instance.solve()
```

- Gọi hàm solve để giải bài toán và trả về kết quả bài toán dưới dạng list.

Hàm input_graph dùng để nhập dữ liệu từ file “map.csv”:

```
def input_graph(namefile:str):
    with open(namefile, 'r', encoding='utf-8') as f:
        reader = csv.reader(f)
        g = nx.Graph()

        for row in reader:
            edge = row[0].split()
            node_a = int(edge[0])
            node_b = int(edge[1])
            g.add_edge(node_a, node_b)
        return g
```

- Biến reader là kiểu iterators với mỗi index chứa nội dung của một dòng, thực hàm split() để loại bỏ khoảng trống. File đọc có dạng danh sách kề nên ta thêm cạnh lần lượt bỏ vào và trả về đồ thị g.

Hàm draw(csp, graph) thực hiện trực quan hóa dữ liệu:

```
def draw(csp, graph):
    color_list = ['red', 'blue', 'green', 'yellow']
    node_colors = [color_list[csp["TinhThanh"][i - 1]] for i in graph.nodes()]
    nx.draw(graph, with_labels=True, node_color=node_colors, node_size=200, font_size=10)
    plt.show()
```

- csp là danh sách số nguyên màu của từng tỉnh.
- graph là chứa dữ liệu các đỉnh và cạnh.

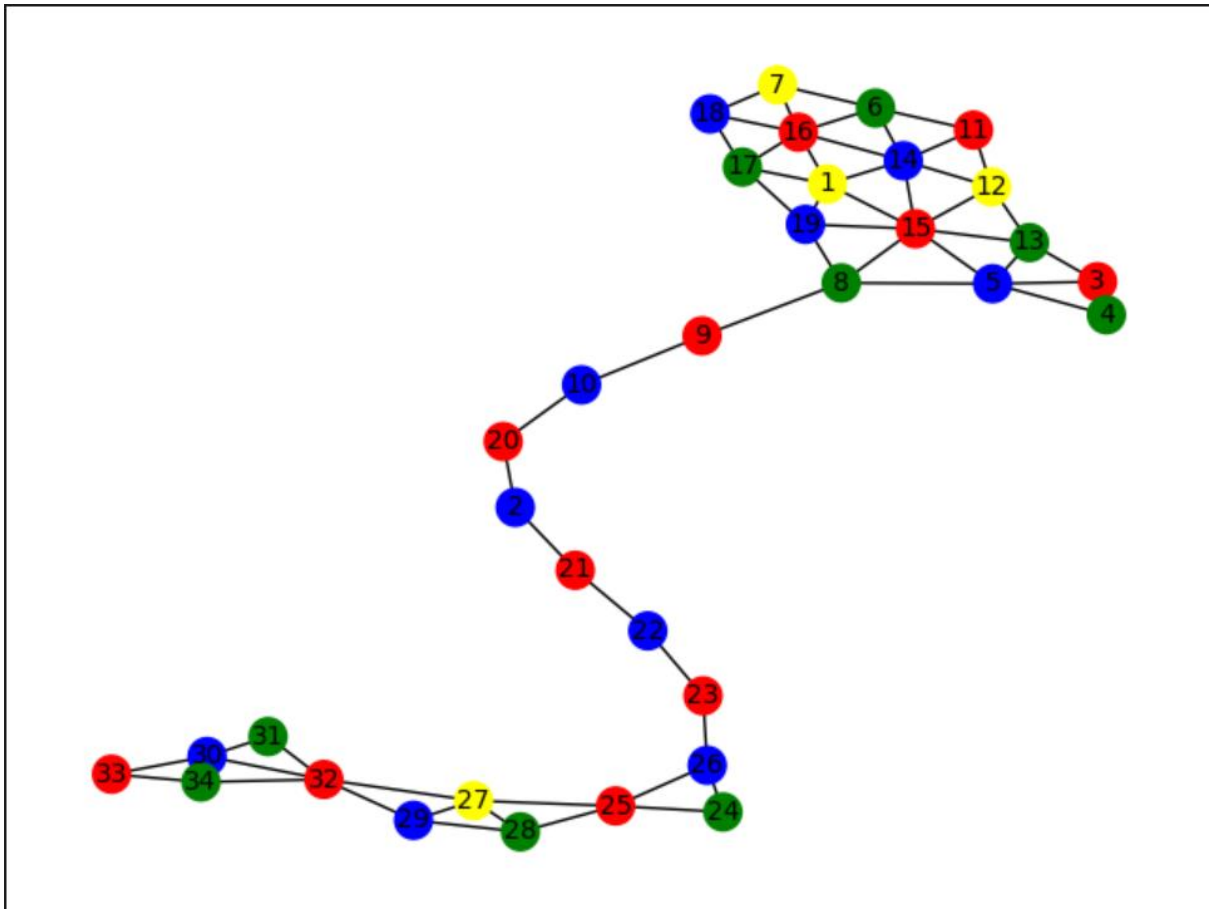
```
    node_colors = [color_list[csp["TinhThanh"][i - 1]] for i in graph.nodes()]
```

- Lập danh sách string màu tương ứng với từng đỉnh với bảng màu color_list.

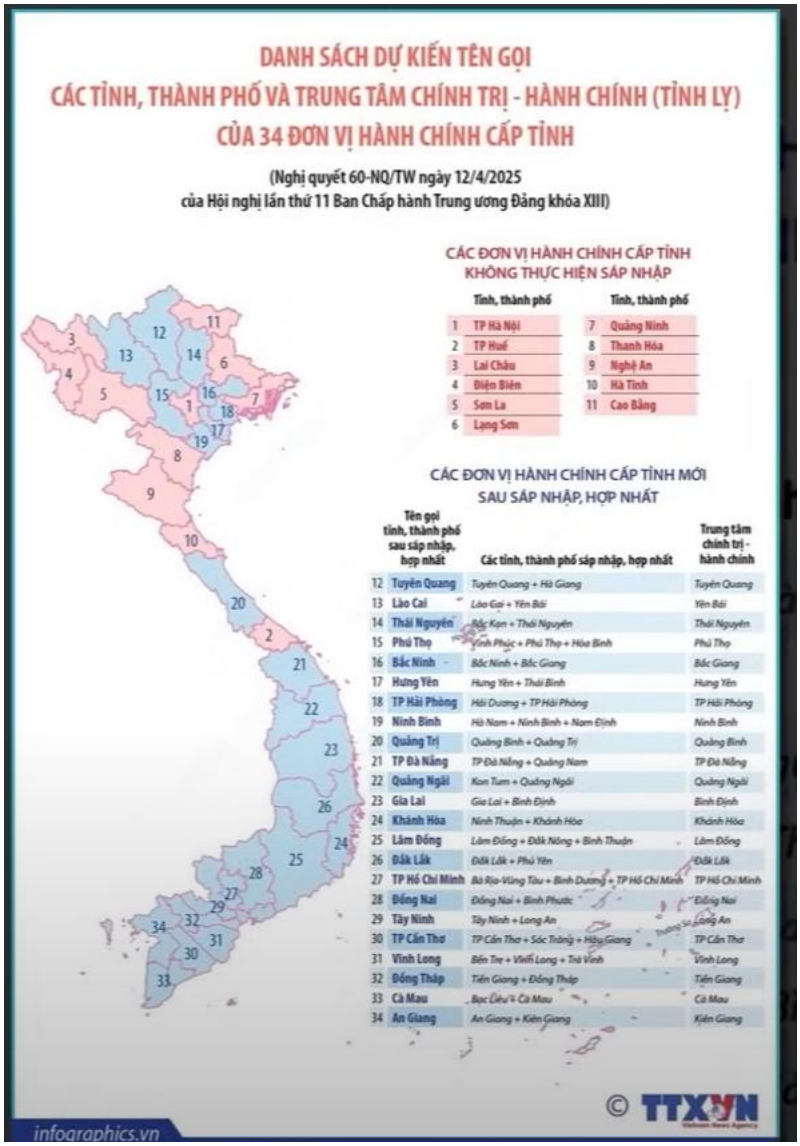
```
nx.draw(graph, with_labels=True, node_color=node_colors, node_size=200, font_size=10)
```

- Thực hiện thêm màu cho đồ thị graph, có ghi tên, node_color=node_colors chứa danh sách màu kiểu string, cỡ đỉnh là 200, cỡ chữ là 10. Rồi thực hiện thị trên ra màn hình bằng plt.show().

Khi chạy chương trình trên với 4 màu thỏa mã ràng buộc và sẽ có hình vẽ như sau:



Với mỗi số tương ứng tên tỉnh thành như sau:



III. Kết luận

Bài toán tô màu bản đồ là một ứng dụng điển hình của CSP. Việc áp dụng MiniZinc giúp giải quyết nhanh chóng, hiệu quả. Nghiên cứu cho thấy với 34 tỉnh thành đó, ta không thể tô màu bản đồ 34 tỉnh thành với 3 màu.

IV. Tài liệu tham khảo

1. Tài liệu bài giảng Viện Công nghệ thông tin và Truyền thông Trường Đại Học Bách Khoa Hà Nội của giảng viên Lê Thanh Hương
2. Tài liệu bài giảng Đại học Giao thông Vận tải thành phố Hồ Chí Minh của giảng viên Nguyễn Văn Diêu
3. https://www.atoha.com/blogs/kien-thuc/ly-thuyet-ve-cac-rang-buoc-theory-of-constraints
4. https://www.minizinc.org/