**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN – CHẤT LƯỢNG CAO**



**REPORT**

**PROJECT 2: COLORING PUZZLE**

CONTENTS

[A. GROUP ROSTER 3](#_Toc79527049)

[B. INTRODUCTION 3](#_Toc79527050)

[C. APPROACH 3](#_Toc79527051)

[D. EXPERIMENT 3](#_Toc79527052)

[E. CONCLUSION 3](#_Toc79527053)

[F. REFERENCES 3](#_Toc79527054)

# GROUP ROSTER

|  |  |  |
| --- | --- | --- |
| **Fullname** | **ID** | **Finished tasks** |
| Nguyễn Huỳnh Khánh Duy | 19127377 |  |

# INTRODUCTION

A coloring puzzle is a problem in which we are given a matrix with each cell containing a value. This value represents the number of green cells adjacent to that cell and including that cell. This value is an integer that belongs to the set {0,1,2,3,4,5,6,7,8,9}. The cells with values ​​that do not belong to the above set will be agreed to have a value of a specific number outside of the above set (here we set it to -1) indicating that we do not have information about the number of green cells here.

The approach to the problem here is that we rely on propositional logic to create CNF clauses. Then use SAT solver to solve, besides SAT solver we can also use A\* algorithm, in addition we will use brute force, backtracking algorithms to compare performance.

# APPROACH

We will apply propositional logic to each cell of the matrix, from normal propositional logic to CNF. We will try on many different cells, from which the general formula can be deduced.

For example:

The first matrix shows the values in the matrix:

|  |  |  |
| --- | --- | --- |
| 3 | 4 | -1 |
| -1 | -1 | -1 |

The second matrix index the order of the cells in the matrix:

|  |  |  |
| --- | --- | --- |
| A | B | C |
| D | E | F |

In the first cell:

A value of 3 means there are 3 adjacent cells including itself which is green.

We will have two conditions:

- There exist 3 green cells (conditions of existence):

After inference:

- If 3 cells are green, the remaining cell is red(conditions of constraints):

After inference and removing duplicates:

After this and more example, we can derive the general formula:

Put:

g: number of blue cells

r: number of red cells

A: set of cells adjacent to the cell, including the cell itself

Conditions of existence:

AND(AND(OR (NEGATIVE-k-COMBINATION(\*)(A))), k = {1,2,...g-1}, OR(S))

Conditions of constraints:

AND(AND(OR (POSITIVE-k-COMBINATION(\*\*)(A))), k = {1,2,...r-1}, OR(S))

Explain:

(\*) NEGATIVE-k-COMBINATION(S): take the k-combination of the set S, negative these elements, the rest will be positive.

(\*\*) POSITIVE-k-COMBINATION(S): take the k-combination of the set S, positive these elements, the rest will be negative.

* About duplication:

Two cells have the same clause only when they have the same set of adjacent cells and the number of blue cells, this only happens when the length and/or width is less than 3.

* Using recursion to create combinations.
* Get all CNF clauses from all cells. Use pysat library to solve these clauses.
* Brute force: Try each model, if the model is satisfied, the solution is found. The number of models is 2n, where n is the number of cells in the matrix.
* Backtracking: Assign values to each cell in turn, check with the assigned value that matches the constraint, if not, assign the next value (here there are only two values, green or red, we can use boolean values). If all cells are assigned a value, the solution is found.
* The program is written in python language.

# EXPERIMENT

Các thí nghiệm anh đã thử: input vào như thế nào, output ra như thế nào, thời gian chạy cho từng thuật toán (có thể sử dụng biểu đồ thể hiện)

* Ít nhất 5 testcase có size khác nhau (10%)
* So sánh giữa các thuật toán với nhau (5%)

|  |  |  |
| --- | --- | --- |
| No | INPUT | OUTPUT |
| 1 | 1 |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |

# CONCLUSION

Các thuật toán có ưu thế trong trường hợp như thế nào, trường hợp tệ nhất như thế nào, trường hợp trung bình là bao nhiêu…

# REFERENCES

Dẫn các nguồn đã tham khảo (bắt buộc)