**ĐẠI HỌC QUỐC GIA HÀ NỘI**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**



**BÁO CÁO CUỐI KỲ MÔN PHƯƠNG PHÁP HÌNH THỨC**

**CHỦ ĐỀ: SLITHERLINK SOLVER**

Sinh viên thực hiện – Nhóm 3:

Văn Đăng Cường – 19020238

Phạm Việt Hà - 19020273

# MỤC LỤC

[MỤC LỤC 1](#_Toc103203168)

[CHƯƠNG 1. GIỚI THIỆU 2](#_Toc103203169)

[1.1. Giới thiệu slitherlink 2](#_Toc103203170)

[1.2. Bài toán SAT và SAT encoding 3](#_Toc103203171)

[CHƯƠNG 2. ENCODING SLITHERLINK 4](#_Toc103203172)

[2.1. Encoding các đỉnh, ô và cạnh 4](#_Toc103203173)

[2.2. Encoding luật 1 4](#_Toc103203174)

[2.3. Encoding luật 2 5](#_Toc103203175)

[2.4. Encoding luật 3 7](#_Toc103203176)

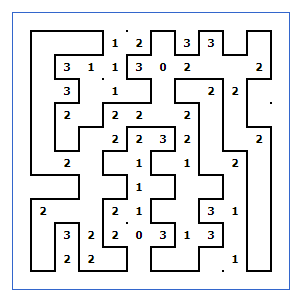
[CHƯƠNG 3. THỰC NGHIỆM 9](#_Toc103203177)

[3.1. Kết quả thực nghiệm 9](#_Toc103203178)

# GIỚI THIỆU

## Giới thiệu slitherlink

Slitherlink là một trò chơi trí tuệ được tung ra bởi Nikoli, công ty Nhật đã phát triển trò chơi Sudoku. Slitherlink được chơi trên một bảng chữ nhật, được chia thành các ô vuông 1x1. Mỗi ô vuông có 1 số nguyên từ 0 đến 3, hoặc là ô trống. Nhiệm vụ của bạn là nối các điểm (là các góc của các hình vuông 1x1) thành 1 đường đi khép kín, sao cho số được ghi trên mỗi ô vuông đúng bằng số cạnh của ô vuông đó mà có đường đi đi qua. (các ô trống có thể có bao nhiêu cạnh thuộc đường đi cũng được.). Một bảng Slitherlink hợp lệ luôn có một cách giải duy nhất.



Hình .: Trò chơi logic Slitherlink

Các luật của slitherlink:

* Luật 1: số cạnh được vẽ của mỗi ô bằng số chứa trong ô vuông đó. Nếu ô vuông đó trống thì có thể có bao nhiêu cạnh được vẽ đều được.

A picture containing chart

Description automatically generated

Hình .: Luật 1 không thỏa mãn

* Luật 2: các đường đi không được phép giao nhau, rẽ nhánh.

A picture containing diagram

Description automatically generated

Hình .: Luật 2 không thỏa mãn

* Luật 3: lời giải phải có chính xác một chu trình.

Shape

Description automatically generated with medium confidence

Hình .: Luật 3 không thỏa mãn

## Bài toán SAT và SAT encoding

Bài toán SAT là bài toán chứng minh tính thỏa mãn hay không thỏa mãn (SAT/UNSAT) của một công thức logic mệnh đề. Nếu tồn tại một phép gán các giá trị true hoặc false cho mỗi biến trong công thức mà công thức ban đầu nhận giá trị đúng thì bài toán là SAT. Ngược lại bài toán là UNSAT.

Công cụ giải công thức logic mệnh đề tự động được gọi là SAT solver.

Một bài toán có thể được giải bằng cách biểu diễn các vấn đề của bài toán dưới dạng các công thức logic mệnh đề. Các công thức này thường được đưa về dạng chuẩn tắc hội (CNF). Kết quả của bài toán được chuyển đổi từ kết quả bài toán SAT. Cách giải như vậy được gọi là SAT encoding và siltherlink là một bài toán như vậy.

# ENCODING SLITHERLINK

## Encoding các đỉnh, ô và cạnh

Đánh số các cạnh bắt đầu từ 1 cho đến hết. Đánh số các cạnh ngang trước từ trái sang phải, từ hàng trên xuống hàng dưới sau đó đánh số các cạnh dọc từ trên xuống dưới, từ cột trái sang cột phải.

Đánh số các đỉnh từ trái sang phải và từ trên xuống dưới bắt đầu từ 0.

Với mỗi đỉnh k ta xác định vị trí như sau :

i = k / (n+1)

j = k % (n+1)

i : dòng thứ i (bắt đầu từ 0)

j : cột thứ j (bắt đầu 0)

Với mỗi đỉnh k các cạnh được xác định như sau:

Cạnh trái: i \* n + j

Cạnh phải: i \* n + j + 1

Cạnh trên: (m + 1) \* n + j \* m + i

Cạnh dưới: (m + 1) \* n + j \* m + i + 1

m : số dòng.

n : số cột

Với mỗi ô (i, j) xác định các cạnh như sau:

Cạnh trên: i \* n + j + 1

Cạnh dưới: (i + 1) \* n + j + 1

Cạnh trái: (m + 1) \* n + j \* m + i + 1

Cạnh phải: (m + 1) \* n + (j + 1) \* m + i + 1

## Encoding luật 1

Luật 1: số cạnh được vẽ của mỗi ô bằng số chứa trong ô vuông đó. Nếu ô vuông đó trống thì có thể có bao nhiêu cạnh được vẽ đều được.

Gọi các cạnh của ô (i, j) có giá trị k là e1, e2, e3, e4 và công thức biểu diễn luật 1 cho ô (i, j) là rule1­(i, j).

Với k = 0: không có cạnh nào được tô vì vậy cả bốn biến nhận giá trị false.

rule1(i, j)= (⌐e1 ˄ ⌐e2 ˄ ⌐e3 ˄ ⌐e4)

Với k = 1: trong bốn cạnh có chính xác một cạnh được tô.

rule1(i, j) = (e1 ˅ e2 ˅ e3 ˅ e4) ˄ (⌐e1 ˅ ⌐e2) ˄ (⌐e1 ˅ ⌐e3) ˄ (⌐e1 ˅ ⌐e4) ˄ (⌐e2 ˅ ⌐e3) ˄ (⌐e2 ˅ ⌐e4) ˄ (⌐e3 ˅ ⌐e4)

Với k = 2: trong bốn cạnh có chính xác hai cạnh được tô.

rule1(i, j) = (e1 ˅ e2 ˅ e3) ˄ (e1 ˅ e2 ˅ e4) ˄ (e1 ˅ e3 ˅ e4) ˄ (e2 ˅ e3 ˅ e4) ˄ (⌐e1 ˅ ⌐e2 ˅ ⌐e3) ˄ (⌐e1 ˅ ⌐e2 ˅ ⌐e4) ˄ (⌐e1 ˅ ⌐e3 ˅ ⌐e4) ˄ (⌐e2 ˅ ⌐e3 ˅ ⌐e4)

Với k = 3: có chính xác một cạnh không được tô.

rule1(i, j) = (⌐e1 ˅ ⌐e2 ˅ ⌐e3 ˅ ⌐e4) ˄ (e1 ˅ e2) ˄ (e1 ˅ e3) ˄ (e1 ˅ e4) ˄ (e2 ˅ e3) ˄ (e2 ˅ e4) ˄ (e3 ˅ e4)

Với k = 4: Cả bốn cạnh đều được tô nên bốn biến đều nhận giá trị true.

rule1(i, j) = (e1 ˄ e2 ˄ e3 ˄ e4)

Công thức logic mệnh đề cho luật này là phép “and” (˄) công thức của tất cả các ô.

Rule1 = rule1(0, 0)˄ … ˄ rule1(m – 1, n - 1)

Với m, n là kích thước ma trận.

## Encoding luật 2

Luật 2: các đường đi không được phép giao nhau, rẽ nhánh.

Ý tưởng: khi một cạnh đi qua đỉnh được tô có chính xác một cạnh trong các cạnh còn lại đi qua đỉnh đó được tô.

Mỗi đỉnh (i, j) có bốn cạnh e1, e2, e3, e4 lần lượt là cạnh trên, cạnh phải, cạnh dưới và cạnh trái. Công thức biểu diễn luật 2 cho đỉnh (i, j) là rule2(i, j).

Có ba trường hợp sau:

* TH1: Đỉnh nằm ở một trong bốn cạnh biên chỉ có ba cạnh.

Chart

Description automatically generated

Hình .: Đỉnh nằm ở biên

* TH2: Đỉnh nằm ở một trong bốn góc chỉ có hai cạnh.

Chart

Description automatically generated

Hình .: Đỉnh nằm ở góc

* TH3: Đỉnh nằm hoàn toàn phía trong có đủ cả bốn cạnh.

Chart, rectangle, box and whisker chart

Description automatically generated

Hình .: Đỉnh nằm trong

Ví dụ trường hợp 3:

rule2(i, j) = [e1 🡪 ExactlyOne(e2, e3, e4)] ˄ [e2 🡪 ExactlyOne(e1, e3, e4)] ˄ [e3 🡪 ExactlyOne(e1, e2, e4)] ˄ [e4 🡪 ExactlyOne(e1, e2, e3)]

Với e1 🡪 ExactlyOne(e2, e3, e4):

e1 🡪 ExactlyOne(e2, e3, e4)

⬄ ⌐e1 ˅ [(e2 ˅ e3 ˅ e4) ˄ (⌐e2 ˅ ⌐e3) ˄ (⌐e2 ˅ ⌐e4) ˄ (⌐e3 ˅ ⌐e4)]

⬄ (⌐e1 ˅ e2 ˅ e3 ˅ e4) ˄ (⌐e1 ˅ ⌐e2 ˅ ⌐e3) ˄ (⌐e1 ˅ ⌐e2 ˅ ⌐e4) ˄ (⌐e1 ˅ ⌐e3 ˅ ⌐e4)

Tương tự với các biểu thức còn lại ta có các khai triển sau:

e2 🡪 ExactlyOne(e1, e3, e4)

⬄ (⌐e2 ˅ e1 ˅ e3 ˅ e4) ˄ (⌐e2 ˅ ⌐e1 ˅ ⌐e3) ˄ (⌐e2 ˅ ⌐e1 ˅ ⌐e4) ˄ (⌐e2 ˅ ⌐e3 ˅ ⌐e4)

e3 🡪 ExactlyOne(e1, e2, e4)

⬄ (⌐e3 ˅ e1 ˅ e2 ˅ e4) ˄ (⌐e3 ˅ ⌐e1 ˅ ⌐e2) ˄ (⌐e3 ˅ ⌐e1 ˅ ⌐e4) ˄ (⌐e3 ˅ ⌐e2 ˅ ⌐e4)

e4 🡪 ExactlyOne(e1, e2, e3)

⬄ (⌐e4 ˅ e1 ˅ e2 ˅ e3) ˄ (⌐e4 ˅ ⌐e1 ˅ ⌐e2) ˄ (⌐e4 ˅ ⌐e1 ˅ ⌐e3) ˄ (⌐e4 ˅ ⌐e2 ˅ ⌐e3)

Kết hợp các khai triển và loại bỏ các biểu thức trùng nhau ta có khai triển cuối cùng như sau:

rule1 = (⌐e1 ˅ e2 ˅ e3 ˅ e4) ˄ (⌐e1 ˅ ⌐e2 ˅ ⌐e3) ˄ (⌐e1 ˅ ⌐e2 ˅ ⌐e4) ˄ (⌐e1 ˅ ⌐e3 ˅ ⌐e4) ˄ (⌐e2 ˅ e1 ˅ e3 ˅ e4) ˄ (⌐e2 ˅ ⌐e3 ˅ ⌐e4) ˄ (⌐e3 ˅ e1 ˅ e2 ˅ e4) ˄ (⌐e4 ˅ e1 ˅ e2 ˅ e3)

Với các trường hợp 1 và 2 ta có các biểu diễn tương tự như sau:

TH1: Đỉnh nằm ở biên:

* Đỉnh nằm ở biên trái tức không có cạnh trái e4:

rule2(i, j) = [e1 🡪 ExactlyOne(e2, e3)] ˄ [e2 🡪 ExactlyOne(e1, e3)] ˄ [e3 🡪 ExactlyOne(e1, e2)]

* Đỉnh nằm ở biên dưới tức không có cạnh dưới e3:

rule2(i, j) = [e1 🡪 ExactlyOne(e2, e4)] ˄ [e2 🡪 ExactlyOne(e1, e4)] ˄ [e4 🡪 ExactlyOne(e1, e2)]

- Đỉnh nằm ở biên phải tức không có cạnh phải e2:

rule2(i, j) = [e1 🡪 ExactlyOne(e3, e4)] ˄ [e3 🡪 ExactlyOne(e1, e4)] ˄ [e4 🡪 ExactlyOne(e1, e3)]

- Đỉnh nằm ở biên trên tức không có cạnh dưới e1:

Rule2(i, j) = [e2 🡪 ExactlyOne(e3, e4)] ˄ [e3 🡪 ExactlyOne(e2, e4)] ˄ [e4 🡪 ExactlyOne(e2, e3)]

TH2: Đỉnh nằm ở góc:

Ta có : ExactlyOne(e) = e

* Đỉnh nằm ở góc trên trái không có cạnh e1 và e4 :

rule2(i, j) = (e2 🡪 e3) ˄ (e3 🡪 e2)

- Đỉnh nằm ở góc trên phải không có cạnh e1 và e2 :

rule2(i, j) = (e3 🡪 e4) ˄ (e4 🡪 e3)

- Đỉnh nằm ở góc dưới phải không có cạnh e2 và e3 :

rule2(i, j) = (e1 🡪 e4) ˄ (e4 🡪 e1)

- Đỉnh nằm ở góc dưới trái không có cạnh e3 và e4 :

rule2(i, j) = (e1 🡪 e2) ˄ (e2 🡪 e1)

Công thức logic cho luật này là phép ‘and’ các công thức của tất cả các đỉnh.

Rule2 = rule2(0, 0) ˄ … ˄ rule2(m – 1, n - 1)

## Encoding luật 3

Luật 3: lời giải phải có chính xác một chu trình.

Ý tưởng: giải công thức kết hợp từ luật 1 và luật 2, rồi xem xét kết quả có thỏa mãn luật 3 hay không. Nếu không thỏa mãn thêm vào công thức ban đầu phủ định của kết quả hiện tại để tìm kiếm một kết quả khác, tiếp tục như vậy đến khi có kết quả thỏa mãn hoặc khi không tìm được thêm kết quả nào khác nữa (UNSAT).

Pseudo code:

Rule1, 2 = Rule1 ˄ Rule2;

sol = Solve(Rule1, 2);

while(type(sol) != String && !oneCycle(sol)){

Rule1, 2 = Rule1, 2 ˄ ⌐sol;

sol = Solve(Rule1, 2);

}

Để kiểm tra kết quả có một chu trình hay không ta làm như sau:

Duyệt từ đỉnh đầu tiên tìm một đỉnh thuộc một cạnh được tô, chọn đỉnh này làm đỉnh bắt đầu. Từ đỉnh bắt đầu đi theo các cạnh được tô đến các đỉnh tiếp theo cho tới khi quay lại đỉnh bắt đầu rồi so sánh số cạnh đã đi qua với tổng số cạnh được tô trong kết quả, nếu bằng nhau kết quả thỏa mãn, nếu không bằng nhau thì kết quả không thỏa mãn.

## Tối ưu luật 3

Ta nhận thấy rằng kết quả được trả về bởi SAT solver rất lớn đối với những ma trận lớn, nếu phủ định toàn bộ các phần tử của nó sẽ dẫn đến mệnh đề được thêm vào cũng có kích thước lớn ảnh hưởng đến tốc độ giải. Vì vậy phần này trình bày phương pháp tối ưu bước phủ định trên để tăng tốc độ của chương trình.

Ý tưởng: Thay phủ định toàn bộ kết quả ta sẽ chỉ phủ định những chu trình nhỏ tồn tại trong kết quả. Chu trình nhỏ là chu trình không đi qua toàn bộ các ô chứa số khác 0. Chu trình lớn là chu trình đi qua tất cả các ô chứa số khác 0.

Ví dụ: Trong hình 2.4 ta thấy có hai chu trình nhỏ màu đỏ chỉ đi qua 2 trong tổng 4 ô. Do đó ta sẽ chỉ thêm vào hai mệnh đề là phủ định của hai chu trình nhỏ này với mỗi mệnh đề chỉ 6 biến thay vì một mệnh đề rất lớn có thể là 60 biến nếu là ma trận 5x5 và sẽ là hàng trăm biến đối với những ma trận lớn hơn nữa.

Shape

Description automatically generated with medium confidence

Hình .: Minh họa chu trình nhỏ

# THỰC NGHIỆM

## Môi trường thực nghiệm

Thực nghiệm được tiến hành trên thiết bị sau:

CPU: Intel core i5 gen 9.

RAM: 8GB

Hệ điều hành: Windows 11 Home Single Language

Ngôn ngữ: python

SAT Solver: pycosat

## Kết quả thực nghiệm

Kết quả thực nghiệm sau đây được tổng hợp từ 11 ma trận đầu vào giải bằng phương pháp đã trình bày ở chương 2. Time out được đặt bằng 900s cho mỗi ma trận. Thời gian có đơn vị là mili giây (ms).

Với phương pháp không tối ưu ta có kết quả như sau:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 5 x 5 | | 7 x 7 | | 10 x 10 | | 15 x 15 | | 20 x 20 | | 30 x 30 |
|  | Normal | Hard | Normal | Hard | Normal | Hard | Normal | Hard | Normal | Hard | Normal |
| Biến | 60 | 60 | 112 | 112 | 220 | 220 | 480 | 480 | 840 | 840 | 1860 |
| Mệnh đề | 303 | 303 | 583 | 574 | 1216 | 1171 | 2710 | 7171 | 8135 | 8392 | 12401 |
| Thời gian (ms) | 15.627 | 0.00 | 0.00 | 0.00 | 218.611 | 343.927 | 499.883 | Time out | Time out | Time out | Time out |
| Lặp | 1 | 2 | 1 | 2 | 59 | 63 | 57 |  |  |  |  |

Bảng .: Kết quả thực nghiệm với phương pháp không tối ưu

Với phương pháp tối ưu:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 5 x 5 | | 7 x 7 | | 10 x 10 | | 15 x 15 | | 20 x 20 | | 30 x 30 |
|  | Normal | Hard | Normal | Hard | Normal | Hard | Normal | Hard | Normal | Hard | Normal |
| Biến | 60 | 60 | 112 | 112 | 220 | 220 | 480 | 480 | 840 | 840 | 1860 |
| Mệnh đề | 303 | 304 | 583 | 575 | 1163 | 1121 | 2659 | 2497 | 4556 | 4614 | 10184 |
| Thời gian (ms) | 0.000 | 0.000 | 0.000 | 15.589 | 0.000 | 15.622 | 0.000 | 46.863 | 62.520 | 124.970 | 562.335 |
| Lặp | 1 | 2 | 1 | 2 | 2 | 4 | 2 | 7 | 5 | 13 | 12 |

Bảng .: Kết quả thực nghiệm với phương pháp tối ưu

Nhận xét:

* Với phương pháp không tối ưu: ta thấy chỉ có thể giải được những ma trận có kích thước nhỏ.
* Với phương pháp tối ưu: có thể giải được những ma trận có kích thước lớn và với những ma trận nhỏ thì tốc độ được tăng lên khá nhiều so với phương pháp không tối ưu. Điều này có thể do kích thước của mệnh đề đã giảm đi khá đáng kể. Ngoài số lần lặp cũng giảm đi không ít điều này cho thấy phương pháp tối ưu còn giúp tìm ra kết quả chính xác nhanh hơn.

## Một vài hình ảnh thực nghiệm

* Graphical user interface

  Description automatically generatedPhương pháp không tối ưu:

Hình .: Kết quả thực nghiệm với ma trận 5 x 5 và phương pháp không tối ưu

Diagram

Description automatically generated

Hình .: Kết quả thực nghiệm với ma trận 10 x 10 và phương pháp không tối ưu

* Với phương pháp tối ưu:

Diagram, schematic

Description automatically generated

Hình .: Kết quả thực nghiệm với ma trận 20 x 20 và phương pháp tối ưu

Diagram, schematic

Description automatically generated

Hình .: Kết quả thực nghiệm với ma trận 30 x 30 và phương pháp tối ưu

# PHÂN CÔNG CÔNG VIỆC

Văn Đăng Cường:

* Thực hiện encoding phương pháp tối ưu và không tối ưu.
* Viết báo cáo.

Phạm Việt Hà:

* Viết giao diện.
* Làm slide thuyết trình.