ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ



BÁO CÁO

Slitherlink Solver

|  |  |
| --- | --- |
| Môn học: Công nghệ phần mềm nâng cao | |
|  | Học viên: Trần Quang Linh – 20528459  Võ Hải Bình - 21025032 |
|  |  |
|  |  |

**Hà Nội – 2021**

Mục lục

[Phần I. Mô tả bài toán và kỹ thuật giải 5](#_Toc90055804)

[1. Trò chơi Slitherlink 5](#_Toc90055805)

[2. SAT Solver 5](#_Toc90055806)

[3. Kỹ thuật giải Slitherlink bằng SAT Solver 6](#_Toc90055807)

[4. SAT encoding: Exactly-K 7](#_Toc90055808)

[4.1. Binomial encoding 7](#_Toc90055809)

[Phần II. Cài đặt 8](#_Toc90055810)

[5. Mã hóa dữ liệu Slitherlink 8](#_Toc90055811)

[6. Encoding luật 1 9](#_Toc90055812)

[7. Giải quyết luật 2 10](#_Toc90055813)

[7.1. Encoding luật 2a 10](#_Toc90055814)

[7.2. Xử lý luật 2b bằng phương pháp reloading 11](#_Toc90055815)

[8. Demo kết quả cài đặt với MiniSat.js 12](#_Toc90055816)

[Phần IV. Kết quả thực nghiệm 14](#_Toc90055817)

Phân công công việc

|  |  |
| --- | --- |
| Trần Quang Linh | * Đánh số cạnh * Encoding luật 1, 2a * Cài đặt giao diện |
| Võ Hải Bình | * Xử lý luật 1 chu trình * Chuẩn bị test * Chạy kết quả thực nghiệm |

# 

Tóm tắt

Báo cáo này được thực hiện sau quá trình tìm hiểu về kỹ thuật giải quyết vấn đề bằng SAT và các phương pháp SAT encoding nổi tiếng. Để có được cái nhìn tổng quan về phương pháp cũng như giúp những người nghiên cứu sau dễ dàng tiếp cận, báo cáo này chọn bài toán giải Slitherlink làm ví dụ để trình bày về quy trình giải quyết một bài toán bằng kỹ thuật SAT. Báo cáo gồm các phần sau:

**Phần I. Mô tả bài toán và kỹ thuật cài đặt –** Giới thiệu về trò chơi Slitherlink, bộ giải SAT và phương pháp SAT encoding ***Binormal.***

**Phần II. Cài đặt –** Mô tả quy trình giải quyết bài toán Slitherlink bằng bộ giải SAT.

**Phần III. Kết quả thực nghiệm** – Demo kết quả với phương pháp SAT encoding ***Binormal*** và bộ giải SAT MiniSat.js

# Mô tả bài toán và kỹ thuật giải

## Trò chơi Slitherlink

Slitherlink (hay còn được biết đến với tên **Fences** hoặc **Loop the Loop**) là một trò chơi giải đố với một ma trận số, nhiệm vụ của người chơi là tô màu các cạnh trên ma trận số này đảm bảo 2 luật đơn giản sau:

* **Luật 1**: Số cạnh được tô màu xung quanh 1 ô đơn vị bằng giá trị của ô đó, nếu ô không có giá trị, số cạnh được tô màu là tùy ý.
* **Luật 2**: Tập hợp các cạnh được tô màu tạo thành 1 chu trình khép kín duy nhất.

Hình vẽ dưới đây là một ví dụ minh họa về đề bài (bên trái) và lời giải (bên phải) của Slitherlink:

A picture containing text, clock

Description automatically generated

Hình I‑1. Ví dụ về đề bài và lời giải của một bài toán Slitherlink.

## SAT Solver

Bài toán SAT hay Boolean Satisfiability là một kỹ thuật biểu diễn các bài toán dưới dạng một công thức logic mệnh đề gồm các biến logic nhận 2 giá trị đúng (true) và sai (false). Nếu tồn tại 1 tập các giá trị ứng với các biến thỏa mãn công thức logic mệnh đề được tạo ra thì bài toán là có lời giải (SAT), ngược lại bài toán không có lời giải (UNSAT). Các bộ giải SAThay **SAT Solver** là công cụ giúp kiểm tra một công thức logic mệnh đề là SAT hay UNSAT.

Hiện nay có rất nhiều SAT Solver với sức mạnh giải quyết vấn đề cao, tuy nhiên việc so sánh sức mạnh giữa các SAT Solver không phải là mục tiêu của báo cáo này. Do đó, báo cáo chỉ giới thiệu và sử dụng 1 bộ SAT Solver khá nổi tiếng là **MiniSat** và không đề cập đến các SAT Solver khác.

## Kỹ thuật giải Slitherlink bằng SAT Solver

Quá trình giải Slitherlink bằng SAT solver có thể được chia ra làm 4 bước:

1. Mã hóa dữ liệu Slitherlink.
2. Encoding các luật Slitherlink thành công thức logic dạng CNF.
3. Sử dụng SAT Solver để tìm lời giải.
4. Chuyển lời giải của SAT Solver (nếu kết quả là SAT) thành dữ liệu Slitherlink và hiển thị kết quả.

## SAT encoding: Exactly-K

Trong quy trình giải bài toán Slitherlink với SAT Solver, chúng ta cần mã hóa các luật chơi Slitherlink thành các công thức logic mệnh đề dạng ***Extractly-K-True*** (Chính xác k giá trị trong tập hợp **X** gồm n giá trị cho trước là thỏa mãn 1 giá trị logic (True hoặc False) với 0<=k<=n) dưới dạng công thức CNF (Conjunctive normal form). Phần này sẽ trình bày cách xây dựng công thức Extractly-K và các kỹ thuật liên quan.

Để xây dựng được công thức ***Exactly-K***, chúng ta cần xây dựng 2 công thức CNF thành phần là ***At-Most-K*** và ***At-Least-K :***

Hiện tại có nhiều phương pháp biểu diễn At-Most-K và At-Least-K dưới dạng CNF, tuy nhiên trong phạm vi báo cáo chỉ giới thiệu và sử dụng 2 kỹ thuật ***Binominal*** và ***Sequential Encounter.***

### Binomial encoding

Xét tập hợp X gồm n biến logic ta có

***Công thức At-Least-K (Tối thiểu k biến trong tập X nhận giá trị v):***

Trong đó:

* v = True hoặc v = False
* ()
* ()

là 1 tổ hợp m biến trong X (có tất cả tổ hợp)

***Công thức At-Most-K:***

# Cài đặt

## Mã hóa dữ liệu Slitherlink

* Ta quy ước các ký hiệu sau:
  + **Value(i, j)** thể hiện giá trị được điền vào ô có tọa độ **(i, j).**
  + **Left(i,j)** thể hiện cạnh trái củaô có tọa độ **(i, j).**
  + **Top(i,j)** thể hiện cạnh trên củaô có tọa độ **(i, j).**
  + **Right(i,j)** thể hiện cạnh phải củaô có tọa độ **(i, j).**
  + **Bottom(i,j)** thể hiện cạnh dưới củaô có tọa độ **(i, j).**

Để chuẩn hóa dữ liệu đầu vào cho SAT Solver, ta tiến thành gán các chỉ số (biến logic) cho giá trị cạnh xung quanh ô có tọa độ (i,j). Với một ma trận kích thước MxN ta có:

Index(i,j) = M\*i + j (chỉ số của ô có tọa độ (i,j)

Left(i,j) = Index(i,j)   (với j<M)

Top (i,j) = (M\*N) + Index(i,j)   (với i<N)

Right(i,j) = Left( i, j+1)   (với 0 <i < M  & 0<j<N)

                   hoặc  (M\*N)\*2 + i (với j=N)

Bottom(i,j) = Top(i+1,j)   (với 0 <i < M  & 0<j<N)

  hoặc  (M\*N)\*2 + M + j   (với i=M)

*Diagram, schematic

Description automatically generated*

Hình II‑1. Ví dụ về đánh số cạnh của 1 ma trận Slitherlink kích thước 2x3.

## Encoding luật 1

* Nếu số k được điền vào ô (i, j), có chính xác k cạnh trong 4 cạnh xung quanh ô (i,j) được tô màu:
* Nếu không có số nào được điền vào ô (i,j), nhiều nhất 4 cạnh xung quanh ô (i,j) được tô màu:

## Giải quyết luật 2

Ta chia nhỏ luật 2 (các cạnh được tô màu tạo thành 1 chu trình khép kín duy nhất) thành 2 luật nhỏ hơn:

* **Luật 2a**: Với mỗi cạnh được tô màu, mỗi đầu của cạnh đó sẽ được kết nối với chính xác với 1 cạnh được tô màu khác. (Đảm bảo tạo thành các chu trình khép kín)
* **Luật 2b**: Số lượng chu trình được tạo thành bằng 1.

### Encoding luật 2a

Với mỗi cạnh e và cách cạnh kết nối với nó ta có:

Graphical user interface, text, application

Description automatically generated

Hình II‑2. Encoding luật 2a

### Xử lý luật 2b bằng phương pháp reloading

Sau khi thực hiện encoding các luật 1 và 2a, ta thu được 1 công thức logic theo dạng CNF để đưa vào bộ giải SAT. Trong trường hợp kết quả là SAT, ta sẽ thu được 1 bộ giá trị thỏa mãn các luật đã cho (các giá trị dương là các biến cạnh được tô màu). Tại đây, chúng ta cài đặt một hàm kiểm tra số lượng chu trình được tạo ra có đúng bằng 1 không thông qua phương pháp đếm cạnh của 1 chu trình bất kì trong lời giải, nếu số cạnh của 1 chu trình đúng bằng số biến dương trong kết quả của SAT solver thì đây chính là lời giải hợp lệ của Slitherlink. Ngược lại, ta tiến hành lấy phủ định của lời giải này và kết hợp với đầu vào để tạo ra 1 đầu vào mới. Hình vẽ dưới đây mô tả quá trình hoạt động của phương pháp Reloading này:

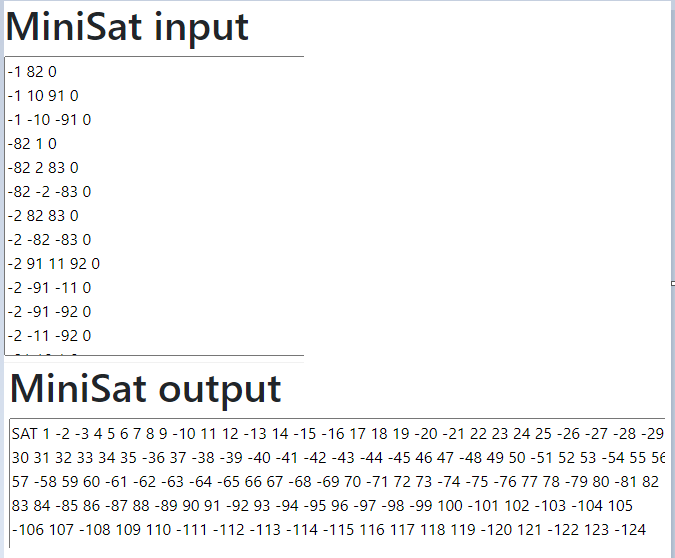
Diagram

Description automatically generated

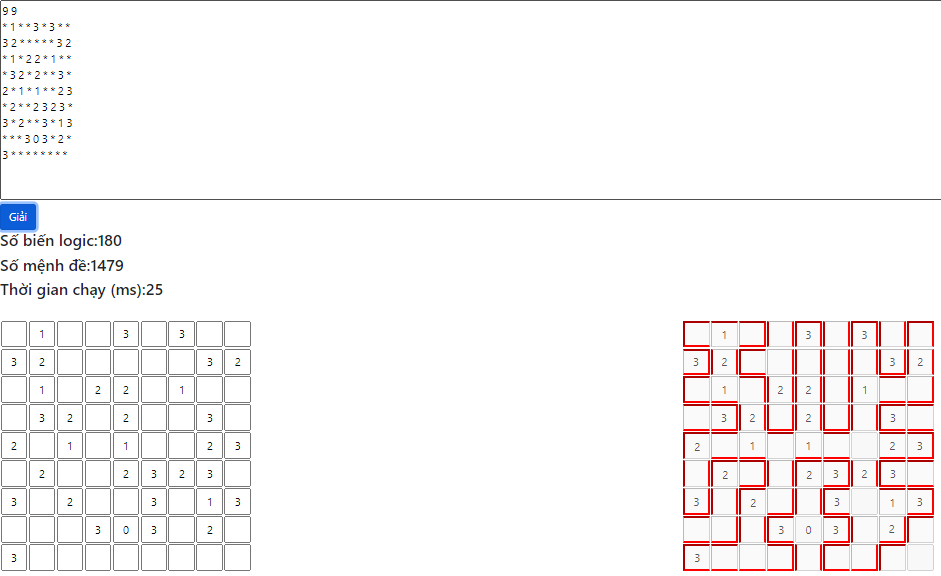
Hình II‑3. Xử lý bài toán 1 chu trình bằng phương pháp reloading

## Demo kết quả cài đặt với MiniSat.js

MiniSat.js là phiên bản được của MiniSat, được biên dịch bằng Emsripten nên nó có thể chạy được trên môi trường trình duyệt web.



*Hình II-2. Ví dụ về đầu vào và đầu ra của MiniSat.js*

Đầu vào của MiniSat.js sẽ là 1 string gồm n mệnh đề viết trên n hàng, mỗi mệnh đề gồm m biến logic phân cách bởi dấu cách và kết thúc bới ký tự ‘0’ (Hình II-2).

*Hình II-3. Giao diện chương trình giải Slitherlink sử dụng MiniSat.js*

# Kết quả thực nghiệm

Tiến hành thực nghiệm với các kích thước ma trận Slitherlink khác nhau và đo lại thời gian chạy (Không tính thời gian xây dựng công thức CNF), số biến logic được sử dụng và số mệnh đề được tạo ra theo từng kỹ thuật encoding:

1. **Môi trường thực nghiệm:**

|  |  |
| --- | --- |
| CPU | Intel I5-9400 |
| GPU | Intel UHD Graphics 630 |
| RAM | 24GB |
| Trình duyệt | Chrome – 96.0 |

*Lưu ý: Do sử dụng MiniSat.js là phiên bản chạy trên trình duyệt web nên chương trình bị giới hạn bộ nhớ RAM cho 1 tab chrome, đối với kỹ thuật binomial encoding, việc xây dựng tổ hợp chập k của n phần tử có thể sẽ gây tràn bộ nhớ hoặc quá thời gian cho phép. Các công thức CNF gồm quá nhiều biến logic hoặc quá nhiều mệnh đề cũng sẽ không thực hiện được do hạn chế về mặt bộ nhớ.*

*A picture containing text

Description automatically generated*

*Hình III-1. Ví dụ về lỗi không đủ bộ nhớ phục vụ việc giải SAT của MiniSat.js*

1. **Kết quả:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kích thước | Độ khó | Số biến | Số mệnh đề | Số lần gọi lại SAT | Thời gian chạy (ms) | | |
| Lần 1 | Lần 2 | Lần 3 |
| 5x5 | Dễ | 60 | 438 | 2 | 5 | 7 | 3 |
| 7x7 | Dễ | 112 | 881 | 2 | 23 | 11 | 9 |
| 9x9 | Trung bình | 180 | 1479 | 2 | 247 | 32 | 25 |
| 10x10 | Rất khó | 220 | 1858 | 1495 | 3028082 | 3835301 | 3545398 |
| 11x11 | Khó | 264 | 2243 | 23 | 523 | 291 | 343 |
| 12x12 | Trung bình | 312 | 2703 | 5 | 311 | 71 | 67 |
| 15x15 | Rất khó | 480 | 4200 | X | Time out | Time out | Timeout |
| 18x18 | Rất khó | 684 | 6158 | X | Time out | Time out | Timeout |
| 20x20 | Rất khó | 840 | 7513 | X | Time out | Time out | Timeout |

*Bảng 1. Kết quả thực nghiệm chương trình giải Slitherlink với MiniSat.js*