TRƯỜNG ĐẠI HỌC CÔNG NGHỆ - ĐHQGHN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN

MÔN CÁC PHƯƠNG PHÁP HÌNH THỨC CHO PHÁT TRIỂN PHẦN MỀM

Mã môn học: ***INT 3106***

Giảng viên: ***TS. Tô Văn Khánh***

Tên đề tài**:** ***HITORI SOLVER***

Nhóm 11:

Ngô Quang Đại (C)	Nam	Lớp: K57CC
Vũ Văn Chiến	Nam	Lớp: K57CLC
Nguyễn Thị Thùy Dung	Nữ	Lớp: K57CB
Lê Khánh Chi	Nữ	Lớp: K57CB
Ngô Đức Dương	Nam	Lớp: K57CC
Chủ Hải Đăng	Nam	Lớp: K57CB

MỞ ĐẦU

Trong logic toán học, SAT (**Satisfiability**) và tính hợp lệ của nó là những khái niệm căn bản. Một công thức logic toán học là SAT nếu nó có thể tìm thấy một lời giải mà làm cho công thức đó có giá trị True. Ngược lại với Satisfiability là Unsatisfiability (**UNSAT**), đó là những công thức logic mà không có lời giải nào làm cho mệnh đề đúng.

Rất nhiều bài toán có thể tự động tìm ra lời giải khi áp dụng công cụ SAT Solver. Trước tiên ta cần chuyển bài toán đó về dạng Logic mệnh đề, sau đó áp dụng SAT Solver vào việc giải các logic mệnh đề đó,cách làm này được gọi là SAT Encoding.

Trong tài liệu này chúng em sẽ thiệu về Hitori, một trò chơi logic có xuất xứ từ Nhật Bản và cách áp dụng SAT Solver vào việc tìm kiếm đáp án cho trò chơi này.

DANH MỤC THUẬT NGỮ

Thuật ngữ	**Ý nghĩa**
SAT (satisfiability)	Thỏa mãn
UNSAT (unsatisfiability)	Không thỏa mãn
SAT Solver	Công cụ chứng minh tính thỏa mãn của các công thức Logic
SAT Encoding	Là phương pháp, cách giải một bài toán bằng việc đưa về bài toán SAT
Mini SAT	Một dạng mã nguồn mở để xây dựng và phát triển mở rộng các SAT Solver
CNF (Conjunctive Normal Form)	Chuẩn tắc hội

MỤC LỤC

DANH MỤC CÁC HÌNH ẢNH

DANH MỤC BẢNG VÀ BIỂU ĐỒ

1. GIỚI THIỆU

1.1. Giới thiệu về trò chơi Hitori

Hitori (ひとりにしてくれ) là một loại trò chơi logic được phát minh bởi công ty Nikoli của Nhật Bản có tên đầy đủ là *Hitori ni shite kure* có nghĩa là "hãy để tôi một mình". Khi chơi Hitori ta cần phải làm cho mỗi số chỉ được xuất hiện không quá một lần trên mỗi hàng và mỗi cột, có một cái gì đó tương tự với sudoku. Tuy nhiên đối với Hitori, người ta không cần điền số vào các ô trống, thay vào đó người chơi sẽ bôi đen các ô có giá trị xuất hiện nhiều hơn một lần trên mỗi hàng và mỗi cột sao cho thỏa mãn các luật của trò chơi.

1.2. Các giàng buộc khi giải trò chơi Hitori

Trên một ma trận có kích thước **NxN** với mỗi ô có giá trị cho trước là các số từ 1 đến n, người chơi sẽ lần lượt bôi đen các ô cho đến khi thỏa mãn 3 điều kiện:

- **Luật 1**: *Trên mỗi hàng và mỗi cột,giá trị trong các ô không được xuất hiện quá một lần.*

6	1	5	4	5	3
1	6	4	3	4	5
2	3	6	1	1	1
2	6	3	5	3	4
3	2	5	6	2	1
5	2	3	1	6	1

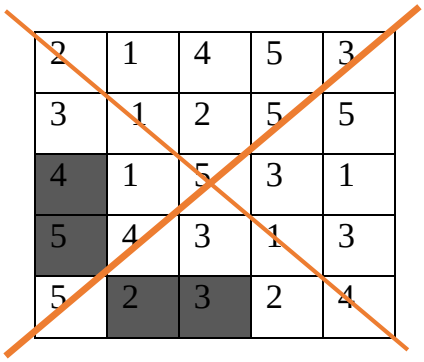
Hình 1.1 Minh họa về luật 1 hitori

Trên hình 1.1, ta thấy ở hàng thứ nhất có hai ô là (1,3) và (1,5) đều có giá trị là 5. Để thỏa mãn luật 1 của Hitori, ta phải bôi đen một trong hai ô đó,khi đó ô giá trị 5 xuất hiện đúng một lần trên hàng thứ nhất. Ta làm tương tự với các hàng và các cột còn lại.

- **Luật 2**: *Các ô được bôi đen không được nằm liền kề nhau trên các hàng hoặc các cột.*

6	1	5	4	5
1	6	4	3	4
2	3	6	1	1
2	6	3	5	3
3	2	5	6	2

Hình 1.2 Minh họa luật 2 hitori



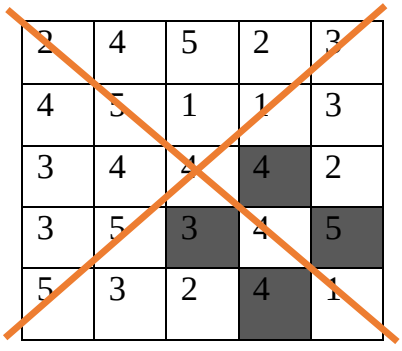
Hình 1.3 Vi phạm luật 2 hitori

Luật 2 có thể giải thích như sau: Nếu ta có ô nằm ở vị trí (i,j) đã được bôi đen thì bốn ô nằm liền kề ô (i,j) là (i,j-1) ,(i,j+1),(i-1,j) và (i+1,j) sẽ không được bôi đen.

- **Luật 3**: *Luôn tồn tại một đường đi giữa các ô trắng đến các ô trắng còn lại (có nghĩa là các ô trắng luôn được kết nối với nhau).*

6	1	5	4	5	3
1	6	4	3	4	5
2	3	6	1	1	1
2	6	3	5	3	4
3	2	5	6	2	1
5	2	3	1	6	1

Hình 1.4: Minh họa luật 3 hitori



Hình 1.5: Minh họa vi phạm luật 3 hitori

Trong hình 1.4 ta có thể thấy, khi bốn ô nằm ở vị trí (3,4), (4,3), (4,5), (5,4) được bôi đen thì ô trắng nằm ở vị trí (5,5) và (4,4) không kết nối được với nhau (không có đường đi giữa hai ô này) và không kết nối được với các ô trắng còn lại. Điều này là vi phạm luật 3 của hitori.

Trên đây là luật chơi của Hitori, dựa vào nó ta có thể đưa ra được các lời giải chính xác cho từng ma trận đầu vào của trò chơi.

Lưu ý: Với một ma trận đầu vào ta có thể tìm ra được nhiều hơn một lời giải. Khi áp dụng SAT Solver, nó sẽ tự động tìm ra hết các kết quả có thể có.

2. PHƯƠNG PHÁP SAT ENCODING

2.1. Các khái niệm cơ bản về SAT Encoding.

2.1.1. Bài toán SAT

Là bài toán nhằm chứng minh tính thỏa mãn hay không thỏa mãn của một mệnh đề logic. Đầu vào của bài toán SAT là một công thức logic mệnh đề biểu diễn dưới dạng CNF (hội của các tuyển sơ cấp).

2.1.2. SAT Solver

Là công cụ chứng minh tự động các công thức logic mệnh đề cho kết quả là thỏa mãn (SAT) hay không thỏa mãn (UNSAT).

Ngày nay, SAT Solver có khả năng giải quyết với số lượng các công thức logic rất lớn, lên đến hàng trăm nghìn mệnh đề, hàng triệu biến.

2.1.3. Mini SAT

Là một dạng mã nguồn mở cho phép các nhà nghiên cứu xây dựng và phát triển mở rộng các SAT Solver dựa trên nó. Là một SAT Solver mạnh.

Một số tính năng chính của **MiniSAT**:

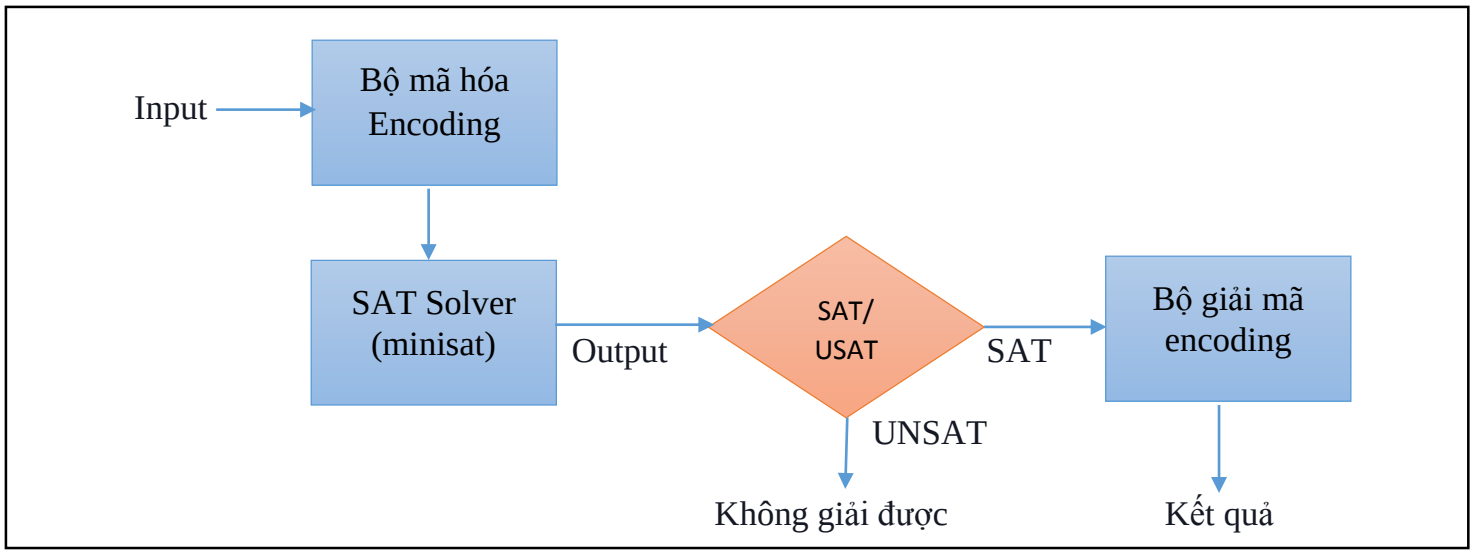
- Dễ dàng chỉnh sửa: **MiniSAT** nhỏ, có tài liệu tốt và được thiết kế tốt, giúp nó trở thành một điểm khởi đầu lý tưởng cho thích ứng kỹ thuật **SAT** dựa vào vấn đề miền cụ thể.
- Hiệu quả cao: Chiến thắng tất cả các loại công nghiệp của cuộc thi **SAT** 2005, **MiniSat** là một điểm khởi đầu tốt đẹp cho cả hai nghiên cứu trong tương lai **SAT**, và cho các ứng dụng sử dụng **SAT**.
- Được thiết kế để tích hợp: **MiniSat** hỗ trợ gia tăng SAT và có cơ chế cho việc thêm các ràng buộc. Nhờ đặc tính dễ thay đổi, đó là một lựa chọn tốt cho việc tích hợp như một phụ trợ công cụ khác, chẳng hạn như một kiểm tra mô hình hoặc một người giải quyết khó khăn chung chung hơn.

2.1.4. SAT Encoding

Là phương pháp, cách giải một bài toán bằng việc đưa về bài toán SAT, biểu diễn các vấn đề ở dạng các công thức logic. Sau đó, áp dụng công cụ SAT Solver để giải.

Phương pháp này thường được sử dụng để giải quyết các trò chơi logic như Hitori, Sudoku, Numberlink, …

- Sơ đồ giải bài toán bằng phương pháp SAT Encoding:



Hình 2.1: Sơ đồ giải Hitori bằng SAT Encoding.

Trong đó:

- **Dữ liệu vào**: Là đầu vào, ma trận bài toán cần giải.
- **Bộ mã hóa**: Nhiệm vụ mã hóa các luật chơi của bài toán thành dạng CNF, sinh ra file đầu vào và đưa vào SAT Solver để giải.
- **Sat Solver**: Là công cụ giải các mệnh đề trong file đầu vào, sinh ra file kết quả, trong bài toán này, chúng tôi sử dụng **sat4j** phiên bản **2.4.3**.
- **Bộ giải mã**: Dịch kết quả trong file kết quả được trả về từ SAT Solver thành lời giải cho bài toán.
- **Kết quả**: Đáp án của bài toán.

2.2. Phương pháp SAT Encoding cho bài toán Hitori

Hai phương pháp **SAT Encoding** được sử dụng để giải quyết trò chơi là:

- Chains and cycles.
- Connectivity Encoding.

Hai phương pháp này đều có cách mã hóa luật 1 và 2 của trò chơi như nhau. Điểm khác nằm ở cách mã hóa cho luật 3.

Một số biến được sử dụng để xây dựng các công thức logic là:

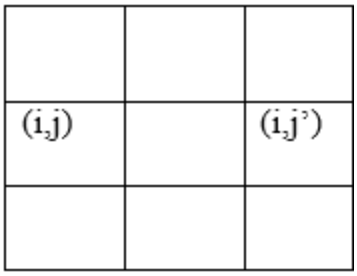
- $\mathbf{x_{ijk}}$: biến này nhận giá trị true khi ô ở hàng i, cột j có giá trị k.
- $\mathbf{b_{ij}}$: biến này nhận giá trị true khi ô ở hàng i, cột j được tô đen.
- $\mathbf{c_{ij,kl}}$: biến nhận giá trị true khi có đường kết nối chéo có hướng từ ô (i,j) đến ô (k,l).

2.2.1. Mã hóa luật 1

Mục đích: Khi xác định được hai hay nhiều ô trên cùng một hàng, hoặc cột có cùng giá trị, ta phải mã hóa để chỉ giữ lại một ô, tất cả các ô còn lại đều phải được tô đen.

Mệnh đề logic cho luật 1:

$$\mathbf{x_{ijk} \wedge x_{ij'k} \rightarrow \neg b_{ij} \vee \neg b_{ij'} \equiv \neg x_{ijk} \vee \neg x_{ij'k} \vee \neg b_{ij} \vee \neg b_{ij'}}$$

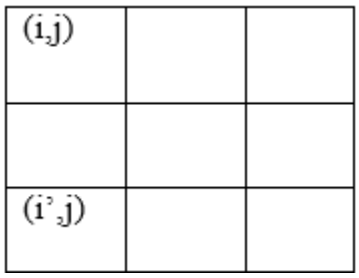


Hình 2.2: Hai ô có cùng giá trị trên 1 hàng

Ý nghĩa: trên hàng i, nếu có 2 ô ở cột j và j' cùng có giá trị k thì hoặc là tô đen một trong 2 ô, hoặc là tô đen cả 2 ô đó.

Tương tự, trên cột j, nếu có 2 ô ở hàng i và i' cùng có giá trị k thì hoặc là tô đen một trong 2 ô, hoặc là tô đen cả 2 ô đó. Ta có mệnh đề:

$$x_{ijk} \land x_{i'jk} \rightarrow \neg b_{ij} \lor \neg b_{i'j} \equiv \neg x_{ijk} \lor \neg x_{i'jk} \lor \neg b_{ij} \lor \neg b_{i'j}$$



Hình 2.3: Hai ô có cùng giá trị trên 1 cột

2.2.2. Mã hóa luật 2

Ta có mệnh đề logic sau:

$$b_{ij} \rightarrow \neg (b_{ij+1} \land b_{ij-1} \land b_{i+1j} \land b_{i-1j}) \equiv (\neg b_{ij} \lor \neg b_{ij+1}) \land (\neg b_{ij} \lor \neg b_{ij-1}) \land (\neg b_{ij} \lor \neg b_{i+1j}) \land (\neg b_{ij} \lor \neg b_{i-1j})$$

	(i-1,j)	
(i,j-1)	(i,j)	(i,j+1)
	(i+1,j)	

Hình 2.4: Các ô lân cận

Ý nghĩa: Nếu 1 ô đã được tô đen thì 4 ô liền kề với nó sẽ không được tô đen.

2.2.3. Mã hóa luật 3

2.2.3.1. Phương pháp Chain & Cycles:

Tìm tất cả các Chain và Cycles trong ma trận Hitori từ đó sinh ra các công thức logic mệnh đề để loại bỏ các Chain và Cycles.

Hạn chế: Khi kích thước ma trận tăng lên số Chain và Cyles là rất lớn dẫn đến công thức CNF sẽ rất phức tạp, SAT Solver giải CNF tốn chi phí lớn.

- **SAT Encoding loại bỏ Chain**

Chain là chuỗi các ô đen kết nối chéo với nhau trong đó ô đầu tiên và ô kết thúc là các ô biên.

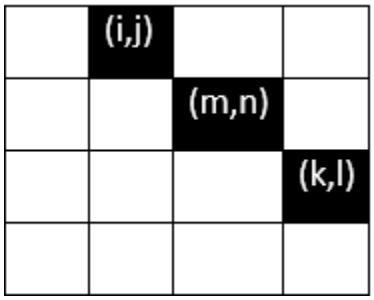
1	4	2	4
3	2	3	1
4	1	3	1
2	4	1	3

Hình 2.5: Một chain với ô bắt đầu và kết thúc là ô biên.

Loại bỏ Chain

Cho một chain C = {(i,j); (m,n); (k,l)} thì mệnh đề lôgic tương ứng để loại bỏ chain C là

$$\mathbf{Cl_{chain} = \neg(b_{ij} \wedge b_{mn} \wedge b_{kl}) \equiv (\neg b_{ij} \vee \neg b_{mn} \vee \neg b_{kl})}$$



Hình 2.6: Minh họa chain {(i,j); (m,n); (k,l)}.

Cl_{chain} đưa ra ràng buộc 3 ô {(i,j); (m,n); (k,l)} sẽ không đồng thời bị tô đen.

- **SAT Encoding loại bỏ Cycles**

Cycle là chuỗi các ô đen được kết nối chéo với nhau tạo thành một đường đi khép kín.

1	2	3	5	4
3	5	3	4	1
5	2	4	2	2
4	1	3	3	5
2	4	5	1	3

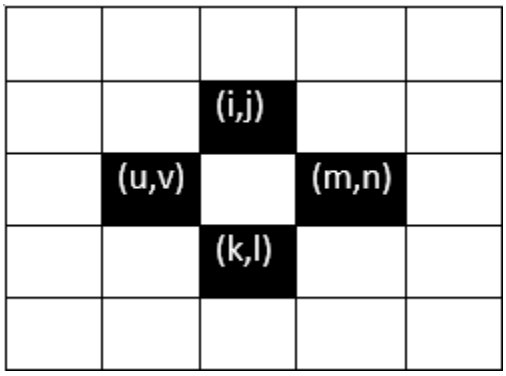
Hình 2.7: Một cycle.

Loại bỏ Cycles

Cho một Cycle Cy = {(i,j); (m,n); (k,l), (u,v)} thì mệnh đề lôgic tương ứng để loại bỏ Cycle Cy là:

$$\mathbf{Cl_{Cycle} = \neg(b_{ij} \wedge b_{mn} \wedge b_{kl} \wedge b_{uv}) \equiv (\neg b_{ij} \vee \neg b_{mn} \vee \neg b_{kl} \wedge \neg b_{uv})}$$

Cl_{Cycle} đưa ra ràng buộc 4 ô {(i,j); (m,n); (k,l), (u,v)} sẽ không đồng thời bị tô đen.



Hình 2.8: Minh họa cycle {(i,j); (m,n); (k,l), (u,v)}

2.2.3.2. Connectivity Encoding

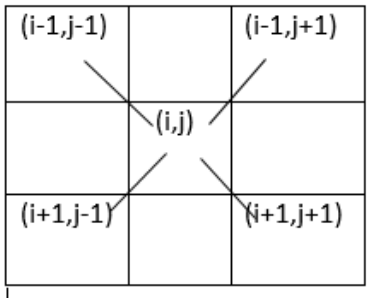
Phương pháp này làm giảm số lượng mệnh đề trong công thức CNF so với phương pháp **Chains &Cycles Encoding**. **Connectivity Encoding** tìm cách loại bỏ đi các **Chains** và **Cycles** trong ma trận, tuy nhiên thay vì tìm kiếm toàn bộ số lượng **Chains** và **Cycles** đó, **Connectivity** xây dựng việc liên kết chéo giữa các ô bị tô đen và đảm bảo việc liên kết chéo giữa các ô đen sẽ không tạo thành kết nối giữa các ô biên(**Chains**) và tạo thành đường khép kín (**Cycles**).

Conectivity sử dụng thêm biến lôgic mệnh đề $c_{ij,kl}$.

- **Kết nối chéo giữa một ô màu đen và 4 ô đen lân cận của nó**

Công thức lôgic mệnh đề tương ứng là:

$$(x_{ijk} \vee x_{i-1j-1k} \vee \neg c_{iji-1j-1} \vee \neg c_{j-1i-1,ij}) \wedge (x_{ijk} \vee x_{i-1j+1k} \vee \neg c_{ij,i-1j+1} \vee \neg c_{i-1j+1,ij}) \wedge (x_{ijk} \vee x_{i+1j-1k} \vee \neg c_{ij,i+1j-1} \vee \neg c_{i+1j-1,ij}) \wedge (x_{ijk} \vee x_{i+1j+1k} \vee \neg c_{ij,i+1j+1} \vee \neg c_{i+1j+1,ij})$$



Hình 2.9: Lân cận của một ô theo kết nối chéo.

Lưu ý: kết nối chéo giữa các ô đen là kết nối có hướng, do đó cij,kl và ckl,ij là hoàn toàn khác nhau.Công thức lôgic mệnh đềcho ràng buộc này là:

$$c_{ij,i+1j+1} \rightarrow \neg c_{i+1j+1,ij} \equiv \neg c_{ij,i+1j+1} \vee \neg c_{i+1j+1,ij}$$

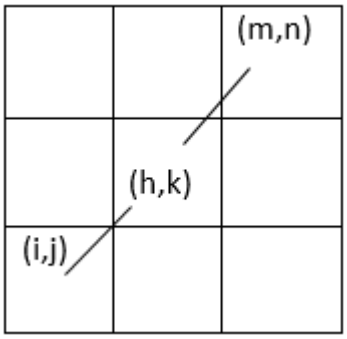
Công thức này quy định kết nối chéo giữa 2 ô đen lân cận chỉ có 1 chiều duy nhất, tức là nếu đã tồn tại kết nối từ ô (i,j) đến ô (i+1,j+1) thì sẽ không có kết nối ngược lại từ ô (i+1,j+1) đến ô (i,j)).

- **Tính chất bắc cầu trong kết nối chéo của các ô đen**

Kết nối chéo có tính chất bắc cầu và mệnh đề lôgic tươngứng là:

$$c_{ij,hk} \wedge c_{hk,mn} \rightarrow c_{ij,mn} \equiv \neg c_{ij,hk} \vee \neg c_{hk,mn} \vee c_{ij,mn}$$

Khi tồn tại một kết nối chéo có hướng từ ô (i,j) tới ô (h,k) và có kết nối chéo giữa ô (h,k) và (m,n) thì sẽ có kết nối chéo từ ô (i,j) tới ô (m,n).

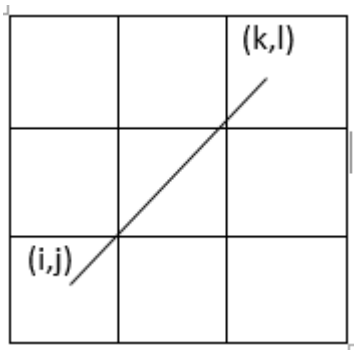


Hình 2.10: Tính chất bắc cầu.

a) Mã hóa loại bỏ Chains:

Xây dựng các mệnh đề để các ô tại biên không được kết nối chéo với nhau. Nếu 2 ô (i,j) và (k,l) là 2 ô biên thì chúng không thể có kết nối chéo với nhau. Mệnh đề lôgic tương ứng là:

$$\neg c_{ij,kl} \wedge \neg c_{kl,ij}$$



Hình 2.11: Kết nối chéo có hướng giữa hai ô

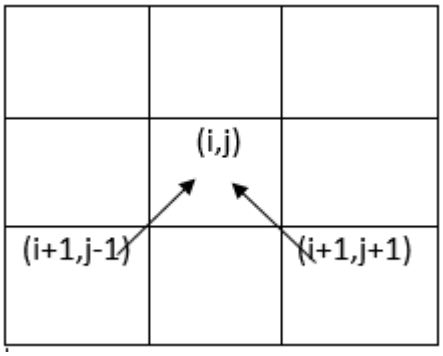
b) Mã hóa loại bỏ Cycles:

Kết nối chéo không được phép hướng vào cùng một ô.

Không cho phép 2 ô bất kỳ có kết nối chéo từ nó đến cùng một ô. Việc đưa ra ràng buộc này quyđinh các đường đi trong một Cycles là đường đi có hướng thống nhất các hướng đi luôn là một chiều) và giúp loại bỏ các khả năng tạo thành Chains.

Mệnh đề lôgic tương ứng:

$$c_{i+1j-1,ij} \rightarrow \neg c_{i+1j+1,ij} \equiv \neg c_{i+1j-1,ij} \vee \neg c_{i+1j+1,ij}$$



Hình 2.12: Hai ô bất kì kết nối tới cùng một ô.

Không tạo kết nối đối với chính nó để không tạo thành Cycles

Ràng buộc này quy định không tồn tại kết nối chéo có hướng khép kín (tạo thành Cycles), nghĩa là không tồn tại một đường đi có hướng từ ô (i,j) đến chính nó.

Mệnh đề lôgic tương ứng: $\neg c_{ij,ij}$

3. THỰC NGHIỆM

Đây là phần trình bày kết quả chạy thực nghiệm 2 phương pháp đã trình bày ở trên. Chúng em tiến hành thực nghiệm trên 60 đầu vào bắt đầu từ ma trận 5x5 và lớn nhất là 30x30.

Cũng trong thực nghiệm, chúng em đánh giá thời gian chạy, số lượng mệnh đề của 2 phương pháp SAT Encoding.

3.1. Môi trường thực nghiệm

- Hệ điều hành: Windows 8.1
- Cấu hình phần cứng: Chip intel core i5 2.5GHz, RAM 4Gb
- Ngôn ngữ lập trình: Java
- Sat solver: sat4j phiên bản 2.4.3

3.2. Kết quả thực nghiệm

Với việc cài đặt và tiến hành thực nghiệm trên cơ sở dữ liệu dựng sẵn, kết quả thu được là sự vượt trội của phướng pháp **Connectivity** so với phương pháp **Chains and cycles**.

Ở phương pháp **Connectivity** chúng em thử nghiệm trên các ma trận từ kích thước **5x5** cho tới kích thước **30x30**. Còn với phương pháp **Chains and cycles,** do chưa thực hiện tối ưu hóa đầu vào nên chỉ có thể thực hiện với kích thước từ **5x5** cho tới **20x20.**

	Chains and cycles (Solver 1)			**Connectivity (Solver 2)**		
Kích thước ma trận	Số biến	Số mệnh đề	Thời gian chạy(ms)	Số biến	Số mệnh đề	Thời gian chạy (ms)
5x5	25	55	13	650	151	25
6x6	36	71	15	1332	327	24
7x7	49	94	16	2450	440	30
8x8	64	519	24	4160	666	62

9x9	81	361	20	6642	900	75
10x10	100	213	21	10100	1023	64
12x12	144	3976	2150	20880	2050	105
15x15	225	28683	8246	50850	2359	120
17x17	289	88299	32424	83810	5900	164
20x20	400	127568	80524	160400	7473	227
21x21	x	x	x	182643	12777	171
23x23	x	x	x	280370	64175	350
25x25	x	x	x	391250	326890	503
30x30	x	x	x	810900	677725	845

Bảng 3.1: So sánh kết quả thực nghiệm của hai phương pháp

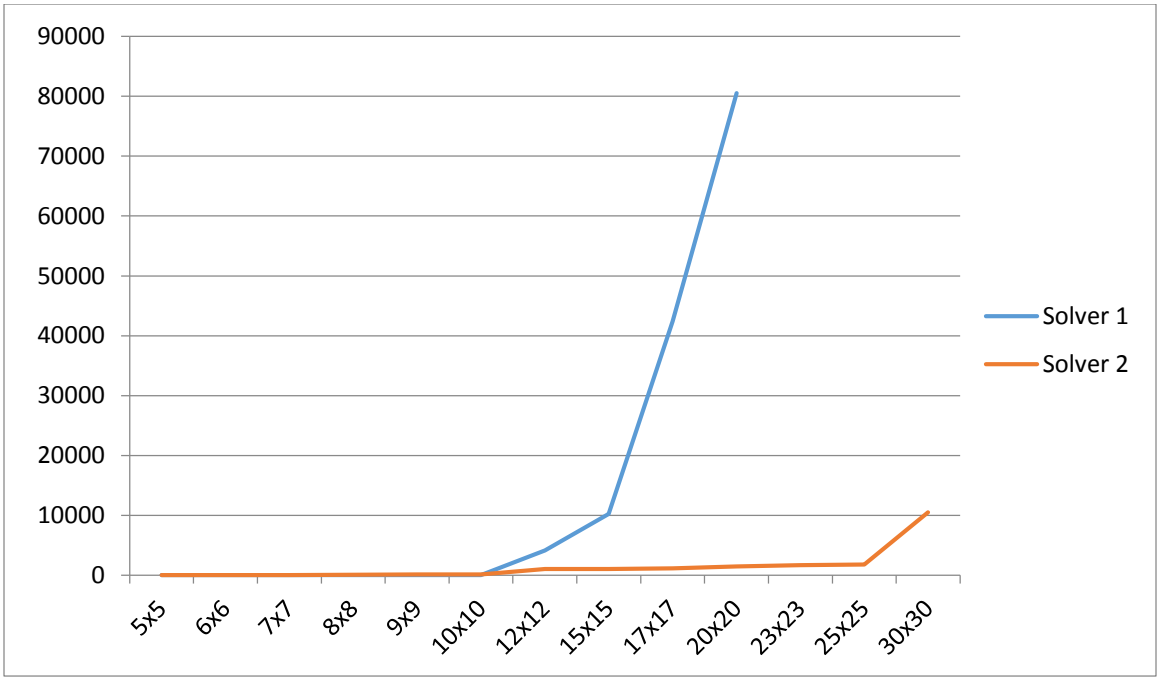
Từ bảng trên có thể thấy, với các ma trận kích thước từ **10x10** trở xuống, sự chênh lệch về thời gian chạy giữa hai phương pháp là không đáng kể, gần như là tương đương nhau. Nhưng từ ma trận **12x12** trở đi, sự khác biệt về thời gian là rất lớn. **Connectivity** vượt trội hơn hẳn so với **Chains and cycles**, ở ma trận **12x12** thời gian chạy của **Solver1** là **2150** ms gấp **20** lần so với **105** ms của **Solver2**. Tới kích thước **20x20** thì **Solver1** mất tới **80524** ms gấp **354** lần so với **227** ms của **Solver2**. Ma trận càng lớn, ưu thế của Solver2 so với Solver1 càng được thể hiện rõ.

Tuy nhiên với **Solver2** thì số biến và số mệnh đề sẽ gấp nhiều lần so với **Solver1**. Điều này có thể giải thích như sau:

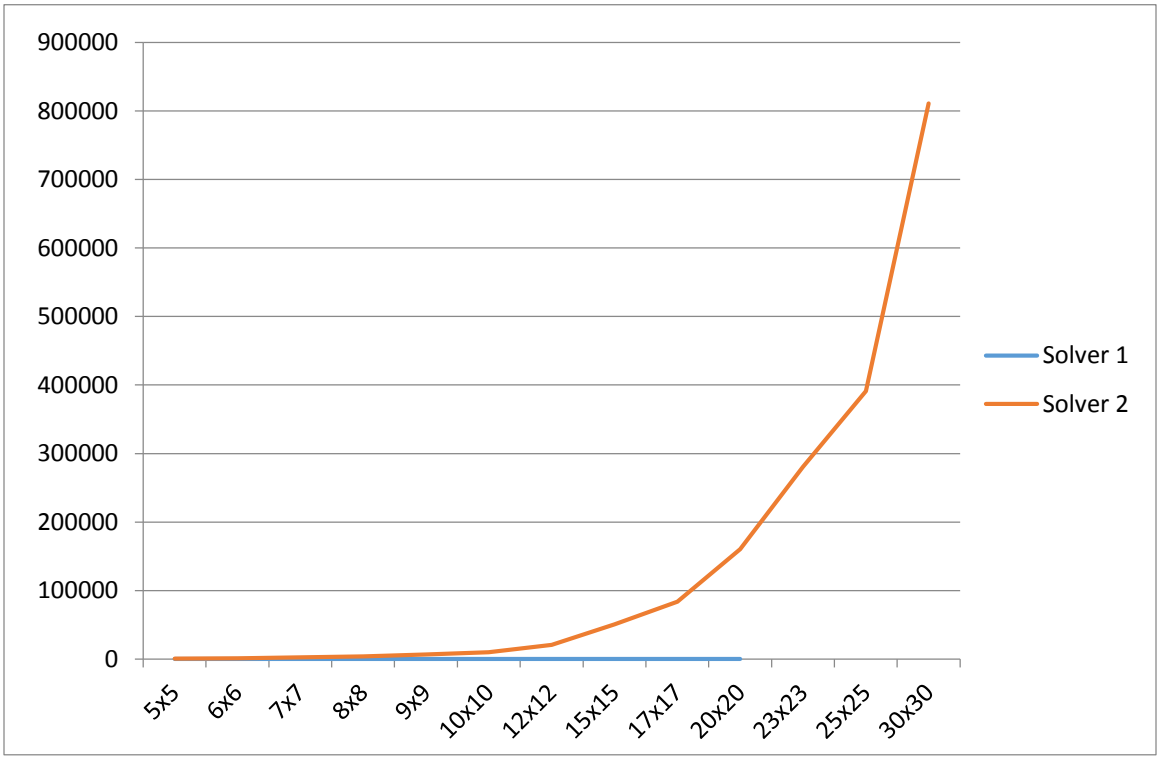
- Với **Solver1** thì chỉ mã hóa mỗi ô là một biến và biểu diễn giá trị, trạng thái của ô đó nên số biến sẽ chỉ là **N*N** biến.
- Với **Solver2**, ngoài việc mã hóa mỗi ô là 1 biến thì còn thực hiện mã hóa hướng đi từ hai ô bất kì. Ví dụ: hướng từ ô thứ **x** tới ô thứ **y** sẽ được mã hóa bởi biến **x * N^2 + y.** Khi đó, toàn bộ ma trận sẽ có số lượng biến là **N^2 + N^4** và lúc này số lượng mệnh đề cũng sẽ tăng lên rất nhiều.

Khi kích thước đầu vào tăng lên thì cho thấy rõ sự tối ưu của phương pháp **conectivity**. Khi kích thước tăng lên đến trên **20x20** thì phướng pháp **Chains and cycles** gần như không thực hiện được.

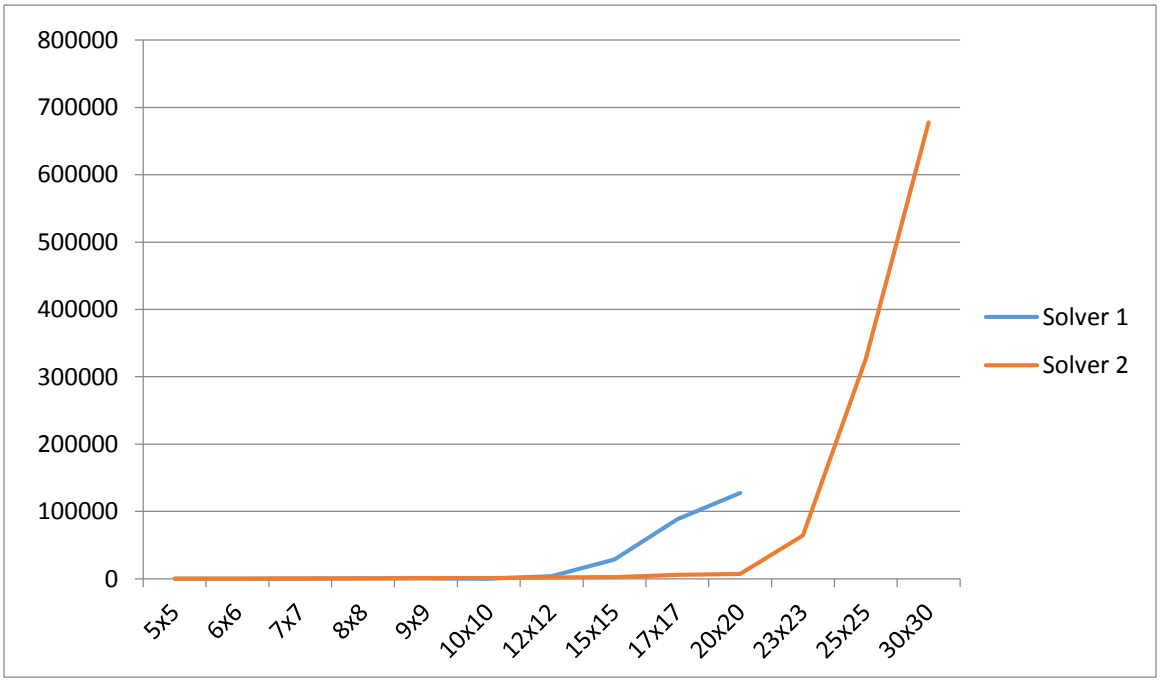
Dưới đây là một số biểu đồ so sánh giữa 2 phương pháp.



Biểu đồ 3.1: So sánh thời gian chạy (ms)



Biểu đồ 3.2: So sánh số biến.



Biểu đồ 3.3: So sánh số mệnh đề.

4. KẾT LUẬN

Bài báo cáo nhóm chúng em đã trình bày hai phương pháp SAT Encoding cũng như cách mã hóa, cài đặt để giải trò chơi Logic Hitori là **Chains and cycles** và **Connectivity** theo hướng dẫn của giảng viên giảng dạy bộ môn.

Kết quả thực nghiệm trên các bài toán cụ thể cho thấy phương pháp **Connectivity** là vượt trội hơn hẳn so với phương pháp **Chains and cycles**, đặc biệt là với những ma trận có kích thước lớn.

Dựa trên ý kiến đóng góp của thầy và kết quả nghiên cứu, một số hạn chế trong cách cài đặt hai thuật toán của nhóm là:

- Chưa thực hiện được việc tối ưu đầu vào.
- Chưa thử nghiệm với hai phương pháp khá hiệu quả các là **2 giải thuật tìm kiếm** đề xuất năm 2006 và năm 2009.

Nhóm sẽ thực hiện việc nghiên cứu bổ sung, cải tiến cách cài đặt để có thể tối ưu hơn nữa các cách giải quyết bài toán Hitori bằng phương pháp SAT Encoding nhằm phục vụ cho các đề tài sau này.

Chúng em xin cảm ơn thầy Tô Văn Khánh đã có những nhận xét, góp ý và hướng dẫn cụ thể, kịp thời để chúng em có thể hoàn thành đề tài này.

5. TÀI LIỆU THAM KHẢO.

[1] SAT4J, www.sat4j.org

[2] Data, www.puzzlebooks.net

[3] Sách tham khảo, Công trình dự thi giải thưởng "Sinh viên nghiên cứu khoa học năm 2015" – Nhóm sinh viên Khoa Công nghệ thông tin, trường ĐH Công nghệ.