ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Đỗ Văn Duy

KẾT HỢP MẪU TÌM KIẾM VÀ PHƯƠNG PHÁP SAT ENCODING GIẢI TRÒ CHƠI HITORI

KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Công Nghệ Thông Tin

HÀ NỘI - 2016

ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Đỗ Văn Duy

KẾT HỢP MẪU TÌM KIẾM VÀ PHƯƠNG PHÁP SAT ENCODING GIẢI TRÒ CHƠI HITORI

KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Công nghệ thông tin

Cán bộ hướng dẫn: TS. Tô Văn Khánh

Hà Nội – 2016

TÓM TẮT

Tóm tắt: Hitori là một trò chơi logic nổi tiếng ở Nhật Bản. Có nhiều cách để giải quyết trò chơi logic này, trong đó việc sử dụng SAT Encoding là một hướng giải quyết rất hiệu quả. Khóa luận này sẽ trình bày những khái niệm cơ bản về SAT, SAT Encoding và áp dụng của chúng. Sau khi giải thích các quy tắc của Hitori, khóa luận trình bày cách mã hóa các quy tắc đó bằng các công thức logic mệnh đề theo nhiều cách khác nhau. Sau đó, sử dụng SAT Solver để giải quyết các công thức logic mệnh đề đó. Bên cạnh đó khóa luận cũng giới thiệu một cách sử dụng SAT Encoding mới để giải bài toán logic Hitori. Tôi tiến hành thực nghiệm trên hai bộ dữ liệu (một bộ lấy từ internet, một bộ tôi tạo ra), sau đó so sánh hiệu quả giải quyết bài toán giữa bốn phương pháp SAT Encoding và một phương pháp tìm kiếm kết hợp quay lui và so sánh trên từng bộ dữ liệu mà tôi có, cuối cùng rút ra kết luận.

Từ khóa: *Hitori Solver, SAT Solver, SAT Encoding*

Lời cam đoan

Tôi xin cam đoan những kết quả nghiên cứu được đưa ra trong khóa luận này là do công sức học hỏi, nghiên cứu của riêng tôi, không sao chép bất kỳ kết quả nghiên cứu của tác giả nào. Tất cả những nội dung sử dụng thông tin, tài liệu từ các nguồn báo cáo, tạp chí, website đều được liệt kê trong danh mục tài liệu tham khảo.

Hà Nội, tháng 4 năm 2016

Sinh viên thực hiện

Đỗ Văn Duy

Lời cảm ơn

Trong suốt thời gian tôi làm khóa luận này, bên cạnh những nỗ lực của riêng tôi, tôi cũng nhận được sự giúp đỡ rất nhiều từ trường Đại học, giảng viên hướng dẫn, cũng như bạn bè và người thân của tôi.

Đầu tiên, tôi xin gửi lời cảm ơn chân thành và sâu sắc đến Tiến sĩ Tô Văn Khánh – giảng viên tại bộ môn Công nghệ phần mềm – người đã nhiệt tình hướng dẫn, giúp đỡ tôi trong suốt thời gian nghiên cứu cũng như hoàn thành khóa luận. Tiếp đến, tôi muốn cảm ơn các thầy cô trong trường Đại Học Công Nghệ, đặc biệt là các thầy cô trong khoa Công nghệ thông tin đã cung cấp cho tôi những kiến thức quý báu để tôi hoàn thành tốt khóa luận này.

Cuối cùng, xin cảm ơn những người thân của tôi, những người luôn ủng hộ, động viên tôi về mặt tinh thần để tôi tập trung hoàn thành khóa luận này. Một lần nữa, xin cảmbơn tất cả.

Hà Nội, tháng 4 năm 2016

Sinh viên thực hiện

Đỗ Văn Duy

Mục lục

DANH MỤC CÁC HÌNH ẢNH

DANH MỤC CÁC BẢNG BIỂU VÀ BIỂU ĐỒ

DANH MỤC CÁC THUẬT NGỮ VIẾT TẮT

Thuật ngữ viết tắt	Viết đầy đủ
SMT	SAT Modulo Theories
SAT	Satisfiablility
UNSAT	Unsatisfiability
CNF	Conjunctive Normal Form
DPLL	Davis-Putnam-Loveland-Logemann
CC	Chains and Cycles Encoding
C1st	Connectivity form 1st Cell
CE	Connectivity Encoding
CE^+	Connectivity Encoding Plus
SWP	Search with patterns
CsNF	Cells not fix
NCs	Neighbouring Cells

Lời mở đầu

Bài toán SAT (Satisfiability) là bài toán chứng minh tính thỏa mãn hay không thỏa mãn (SAT/UNSAT) của một hay nhiều công thức Lôgic mệnh đề. Các công cụ chứng minh tự động cho bài toán này được gọi là SAT Solver, chúng có nhiều ứng dụng trong trí tuệ nhân tạo hay trong các bài toán kiểm chứng, kiểm thử phần mềm. Các SAT Solver đóng vai trò như các công cụ nền *(backend engine)* cho các SMT (SAT Modulo Theories) Solver, là các công cụ chứng minh tính thỏa mãn của các công thức Lôgic xây dựng trên lý thuyết Lôgic vị từ cấp I (First Order Lôgic - FOL).

Hitori là một trò chơi lôgic của người Nhật. Đã có rất nhiều phương pháp giải quyết bài toán Hitori từ cổ điển như "Tìm kiếm theo mẫu" hay các phương pháp sử dụng SAT Encoding để giải quyết.

Khóa luận này sẽ trình bày tất cả những phương pháp giải bài toán logic Hitori đã có như thuật toán tìm kiếm theo mẫu kết hợp quay lui (*Search With Patterns)*, nhiều cách giải sử dụng SAT Encoding như *Chains and Cycles Encoding*, *Connectivity Encoding*, cũng như những phương pháp mới được tìm thấy, qua đó so sánh đánh giá hiệu năng của chúng. Một cách SAT Encoding mới tìm thấy cũng được đề cập và so sánh đó là cách dùng SAT Encoding giải bài toán Hitori của PGS Pedro Cabalar. Khóa luận cũng đề xuất một phương pháp giải bài toán logic Hitori sử dụng SAT Encoding hoàn toàn mới mang tên *Connectivity Encoding Plus* dựa trên thuật toán tìm kiếm theo mẫu kết hợp quay lui và thuật toán *Connectivity Encoding*. Qua thực nghiệm sẽ cho thấy kết quả thực sự tốt của *Connectivity Encoding Plus* (CE^+) so với các phương pháp còn lại.

Tính cấp thiết của đề tài

Việc áp dụng SAT Encoding vào giải quyết các bài toán logic có thể mang lại hiệu quả vượt trội so với nhiều phương pháp khác. Bởi lẽ đó, SAT Encoding cần được áp dụng trong nhiều lĩnh vực khác nhau. Bên cạnh đó, việc áp dụng SAT Encoding cũng cần được chú trọng và quan tâm hơn nữa trong các trường đại học, các viện nghiên cứu để có thể khai thác toàn bộ lợi ích mà nó mang lại.

Mục tiêu nghiên cứu

Trình bày những điểm cơ bản về bài toán SAT, SAT Encoding, ứng dụng SAT Encoding để giải quyết một vấn đề.

Áp dụng SAT Encoding để giải trò chơi logic Hitori.

Cải tiến phương pháp SAT Encoding nhằm mục đích cải thiện hiệu quả giải quyết trò chơi logic Hitori.

Ý nghĩa khoa học

Đề xuất và cải tiến phương pháp SAT Encoding mới (CE^+) để giải trò chơi logic Hitori, qua đó định hướng cho tương lai nhằm cải thiện hơn nữa hiệu quả của chương trình.

Ý nghĩa thực tiễn

Xây dựng chương trình tự động giải trò chơi logic Hitori. Qua đó làm nền tảng cho việc xây dựng hệ thống trò chơi logic này cho con người tự chơi.

Nội dung khóa luận

Chương 1: Giới thiệu về trò chơi logic Hitori, xuất xứ và các luật chơi của nó. Giới thiệu tổng quan về SAT Encoding, quy trình giải một bài toán logic bằng SAT Encoding, các công cụ giải bài toán SAT.

Chương 2: Giới thiệu những phương pháp giải quyết trò chơi logic Hitori trước đó và đề xuất một phương pháp SAT Encoding mới có tên Connectivity Encoding Plus để giải bài toán này.

Chương 3: Trình bày thực nghiệm để đánh giá giữa phương pháp được đề xuất và các phương pháp đề xuất trước đó.

Chương 4: Chương trình tự động giải trò chơi logic Hitori, modul tự động thực nghiệm các phương pháp giải trò chơi logic Hitori và modul tự động tạo ra ma trận đầu vào cho bài toán logic Hitori.

Kết luận: Trình bày kết quả đạt được.

Chương 1. Giới thiệu

Trong chương này, khóa luận sẽ giới thiệu về trò chơi lôgic Hitori, phương pháp SAT Encoding một phương pháp giải bài toán bằng cách biểu diễu dữ liệu bài toán về dạng những mệnh đề logic và giải chúng để tìm ra kết quả.

1.1 Trò chơi Hitori

Hitori [11] (ひとりにしてくれ *Hitori ni shite kure*; "leave me alone" hay "hãy để tôi một mình") là một câu đố logic của người Nhật xuất bản bởi Nikoli.

4	8	1	6	3	2	5	7
3	6	7	2	1	6	5	4
2	3	4	8	2	8	6	1
4	1	6	5	7	7	3	5
7	2	3	1	8	5	1	2
3	5	6	7	3	1	8	4
6	4	2	3	5	4	7	8
8	7	1	4	2	3	5	6

Hình 1.1: Đầu vào của trò chơi lôgic Hitori

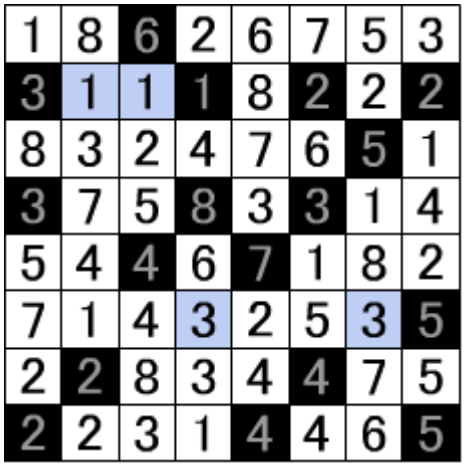
Hitori được chơi với một ma trận số. Mục tiêu của trò chơi là loại bỏ số bằng cách điền vào các ô hay bôi đen (xóa bỏ số) để thỏa mãn 3 luật được đề ra.

Luật chơi Hitori rất đơn giản chỉ gồm 3 quy tắc. Ta quy ước nói bôi đen một ô nào đó có nghĩa là số trông ô đó sẽ được xóa đi.

Luật 1: *Giá trị của số trong mỗi ô không được xuất hiện nhiều hơn một lần ở mỗi hàng, mỗi cột*

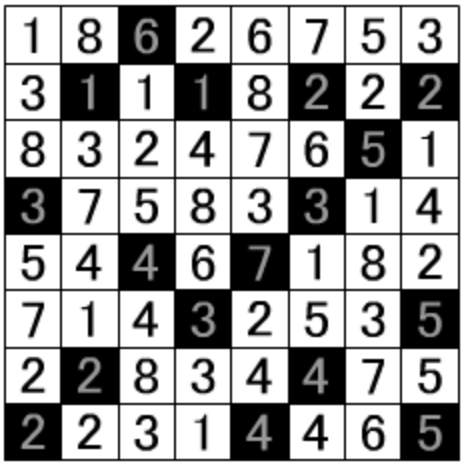
Luật 2: *Các ô bôi đen không được nằm liền kề nhau trên hàng hoặc cột*

Luật 3: *Mọi ô không bôi đen (ô trắng) phải kết nối được với nhau. Nói cách khác luôn tồn tại một đường đi từ mọi ô trắng tới các ô trắng còn lại.*



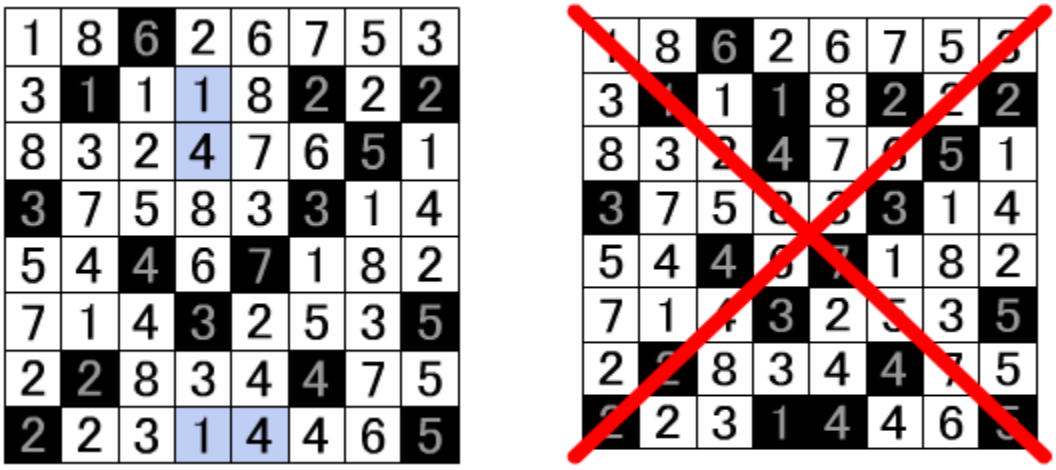
Hình 1.2: Kết quả sai của luật 1

Một kết quả sai (*Hình 1.2*) khi hai ô có cùng giá trị trên cùng hàng cùng tồn tại (các ô được tô màu xám).



Hình 1.3: Kết quả đúng của luật 1

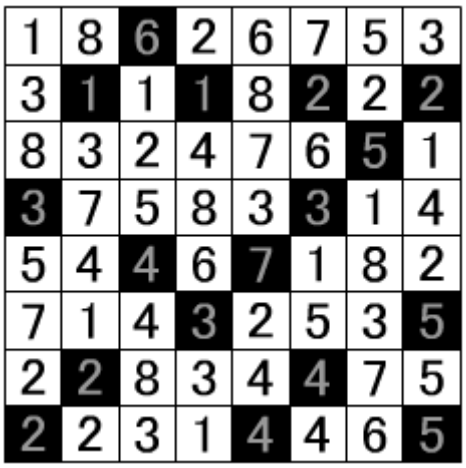
Kết quả đúng (*Hình 1.3*) khi ở hàng thứ 2 và thứ 6 các cặp ô có cùng giá trị thì một trong 2 ô cùng giá trị đó đã bị bôi đen (xóa bỏ).



a) *b)*

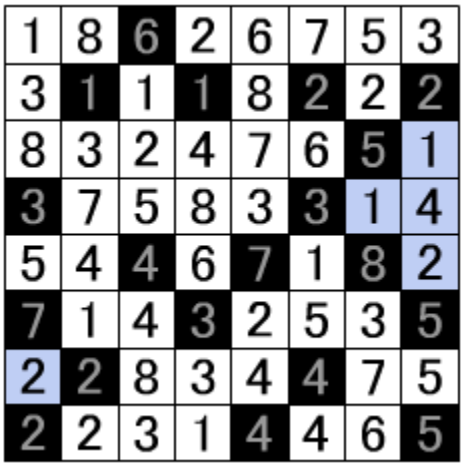
Hình 1.4: Kết quả sai của luật 2

Hình 1.4 cho thấy một kết quả sai, các ô được tô màu xám (Hình 1.4a) bị bôi đen liền kề nhau theo chiều ngang hoặc dọc.



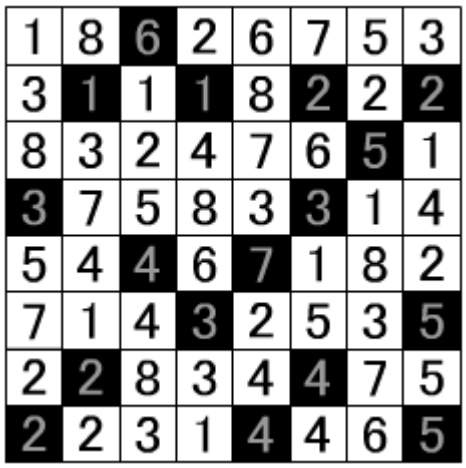
Hình 1.5: Kết quả đúng của luật 2

Hình 1.5 là kết quả chính xác



Hình 1.6: Kết quả sai của luật 3

Hình 1.6 thể hiện kết quả là sai khi các ô trắng được đánh dấu màu xám không thể có đường đi với các ô trắng ở bên ngoài đường bao màu đen xung quanh chúng.



Hình 1.7: Kết quả đúng của luật 3

Kết quả đúng phải là như Hình 1.7

Chú ý: Một kết quả đúng là một kết quả đồng thời thỏa mãn cả 3 luật của một trò chơi. Một câu đố Hitori có thể có nhiều kết quả khác nhau.

1.2 Phương pháp SAT Encoding

1.2.1 SAT Solver

SAT Solver là công cụ chứng minh tự động cho bài toán SAT, tức là cho những công thức logic mệnh đề, SAT Solver thường cài đặt nhiều thuật toán và các phương pháp khác nhau nhằm tối ưu quá trình tìm lời giải. Kết quả của SAT Solver là SAT nếu bài toán có lời giải hoặc UNSAT nếu bài toán không tồn tại lời giải.

Input cho SAT Solver thường theo dạng chuẩn nối tiếp CNF (Conjunctive normal form) có dạng: $(x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee \neg x_5)$ trong đó x_1, x_2, x_3, x_4, x_5 là các biến logic (chỉ nhận 2 giá trị true hoặc false), mỗi biểu thức dấu ngoặc tròn gọi là 1 mệnh đề (clause). Công việc của Sat solver là tìm 1 bộ giá trị cho X_1, X_2, X_3, X_4, X_5 để cho kết quả của biểu thức là true.

❖ Cách thức hoạt động của SAT Solver

Hầu hết các Sat solver hiện nay đều dựa trên thuật toán DPLL [3] (Davis-Putnam-Loveland-Logemann). Đây là thuật toán được giới thiệu đầu tiên vào năm 1962 bởi Martin Davis và cộng sự.

Thuật toán DPLL dựa trên thuật toán quay lui, vét cạn tìm tất cả các bộ giá trị có thể bằng cách gán các giá trị có thể cho tất cả các biến logic, sau đó kiểm tra kết quả để đưa ra kết luận của bài toán.

Trong khi thực hiện, thuật toán thường được tối ưu dựa vào 2 quy tắc:

- Nếu mệnh đề chỉ gồm 1 biến thì nó được gán giá trị true, không cần thử với giá trị false.
- Nếu 1 biến chỉ có dạng khẳng định hoặc phủ định thì khi gán cho nó giá trị true, tất cả các mệnh đề chứa biến đó trở thành true và có thể được loại ra.

Ví dụ: với công thức $(x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee \neg x_5)$

- Với quy tắc 1 thì ta thấy biến x_2 chỉ có thể nhận giá trị true
- Với quy tắc 2 ta thấy khi gán x_2 = true thì 2 mệnh đề đầu tiên trở thành true và ta không cần xét đến nó nữa, có thể loại ra để tăng tốc quá trình giải.

❖ Một số SAT Solver đáng chú ý

- **Minisat:**

Minisat [4] là 1 Sat solver được phát triển đầu tiên vào năm 2003 dưới giấy phép MIT mã nguồn mở. Minisat được viết bằng C/C++ nên có đặc điểm nhỏ gọn, nhanh, hiệu quả cùng với tài liệu sử dụng rất chi tiết. SAT Solver này chạy được trên 2 hệ điều hành Linux và Windows nên có thể dễ dàng được tích hợp vào các ứng dụng khác nhau.

Một số tính năng chính của minisat:

- *Dễ dàng chỉnh sửa*: minisat nhỏ gọn và có tài liệu tốt, và có thể nói là thiết kế tốt, làm cho nó có một điểm khởi đầu lý tưởng cho thích ứng kỹ thuật SAT dựa vào vấn đề tên miền cụ thể.
- *Hiệu quả cao*: Chiến thắng tất cả các công nghệ của cuộc thi SAT 2005, minisat là điểm khởi đầu tốt cho cả hai nghiên cứu trong tương lai là SAT và các ứng dụng sử dụng SAT.
- *Thiết kế cho hội nhập*: Minisat hỗ trợ gia tăng SAT và có cơ chế cho việc thêm các mệnh đề không hề khó khăn. Nhờ được dễ dàng để sửa đổi, đó là một lựa chọn tốt cho việc tích hợp như một phụ trợ công cụ khác, chẳng hạn như kiểm tra một mô hình hoặc nhiều hơn một trình giải quyết hạn chế chung chung.

Cách sử dụng cơ bản của Minisat

Sau đây là cách sử dụng MiniSat, các Sat solver khác cách sử dụng tương tự Input của MiniSat có dạng : p cnf NUMBER_OF_VARIABLES NUMBER_OF_CLAUSES sau đó là các mệnh đề.

Ví dụ : p cnf 5 3

Có nghĩa là input được cho dưới dạng CNF, công thức gồm có 5 biến và 3 mệnh đề, các mệnh đề có dạng:

1 -5 3 0

-1 3 4 5 0

-3 4 0

Dòng đầu tiên của công thức : X1 ∨ X3 ∨ ¬X5 tương tự cho các dòng tiếp theo

Output của MiniSat có dạng tương tự

Ví dụ với input trên thì MiniSat sẽ cho kết quả dạng:

SAT 1 2 -3 4 5 0

có nghĩa là : X_1 = true, X_2 = true, X_3 = false, X_4 = true, X_5 = true.

SAT4j là một thư viện được viết bằng Java, rất thân thiện ,nhiều chức năng, dễ tích hợp có thể giải được nhiều bài toán như : SAT, MAXSAT, Pseudo-Boolean, Minimally…

Viết bằng java nên nó không phải là nhanh nhất để giải quyết những vấn đề (một trình giải quyết SAT trong java chậm hơn so với bản sao của mình trong C++ khoảng 3.25 lần) nhưng nó có đầy đủ tính năng, mạnh mẽ, người dùng thân thiện và làm theo hướng dẫn thiết kế java và quy ước mã (kiểm tra sử dụng phân tích tĩnh của mã nguồn). Thư viện này được thiết kế cho tính linh hoạt. Hơn nữa Sat4j là mã nguồn mở và phân phối dưới giấy phép Eclipse, LGPL, GNU.

- **Glueminisat:**

Glueminisat [6] là một trình giải quyết SAT xây dựng dựa trên khoảng cách các khỗi chữ (LBD) đề xuất bởi Audemard và Simon là một tiêu chí đánh giá chất lượng các mệnh đề trong các trình giải quyết CDCL. Hiệu quả của LBD được thể hiện trong đường giải quyết SAT của họ tại cuộc thi mới nhất SAT. Glueminisat dùng một khái niệm hạn chết nhỏ của LBD, gọi là LBD nghiêm ngặt.

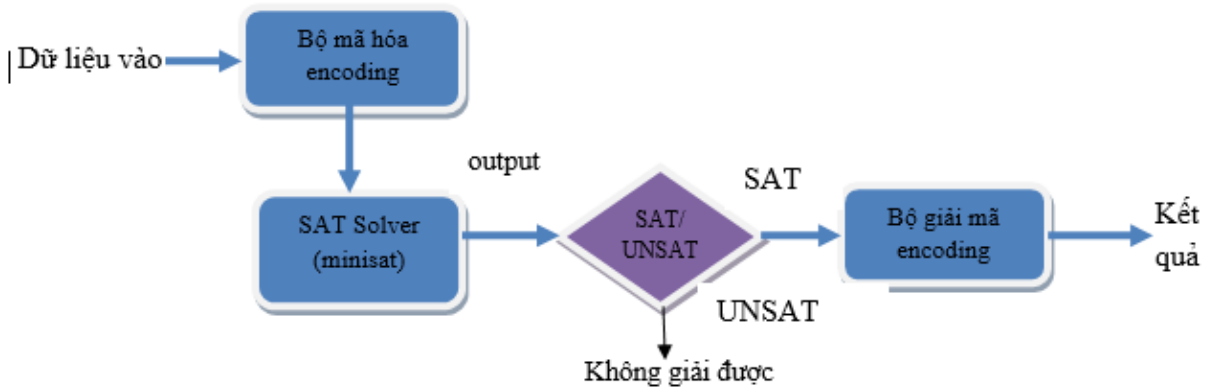
1.2.2 SAT Encoding

SAT Encoding là phương pháp biểu diễn vấn đề bằng công thức logic mệnh đề và áp dụng SAT Solver vào giải các công thức mệnh đề đó. Từ kết quả trả về của SAT Solver phương pháp đưa ra lời giải của bài toán (khi SAT Solver thông báo kết quả của bài toán là Satisfiable), hoặc thông báo không tìm được lời giải (Unsatisfiable). SAT Encoding đã được ứng dụng để giải quyết một số trò chơi Logic như Sudoku, Slitherlink, Hitori, Numberlink.

Một bài toán sử dụng SAT Encoding để giải sẽ thực hiện theo thứ tự sau:

Sơ đồ hình 1.8 gồm các thành phần:

- ***Dữ liệu vào***: Là bài toán cần giải, ví dụ đầu vào là một ma trận *Hitori*, nó gồm giá trị các số ở mỗi ô của ma trận.
- ***Bộ mã hóa***: Mã hóa các luật chơi của dữ liệu vào thành các mệnh đề CNF (*conjunctive normal form*) theo chuẩn DIMACS để đưa vào SAT solver. Khi mã hóa sẽ sinh ra file đầu vào rồi đưa tới SAT Solver.
- ***SAT Solver***: Là công cụ sẽ giải các mệnh đề trong file đầu vào rồi sinh ra file kết quả.
- ***Bộ giải mã***: Dịch kết quả trong file kết quả trả về từ SAT Solver, đưa ra lời giải cho bài toán đầu vào.



Hình 1.8: Biểu đồ thứ tự thực hiện bài toán SAT

- ***Kết quả***: Là đáp án cho dữ liệu vào, nếu đầu vào là một ma trận Hitori thì kết quả là đáp án của ma trận Hitori đó.

1.2.3 Một số ứng dụng của SAT Encoding

Trong phần này, khóa luận trình bày một số ứng dụng của phương pháp SAT Encoding để giải một số trò chơi logic phổ biến ở Nhật Bản. Trong đó, khóa luận trình bày cụ thể về cách Encoding cho các luật của trò chơi Sudoku. Ngoài ra cũng trình bày về các luật của trò chơi Slitherlink và Numberlink, qua đó chúng ta sẽ tìm được hướng tiếp cận để Encoding cho hai trò chơi này.

1.2.3.1 SAT Encoding giải bài toán Sudoku

Sudoku là một trò chơi logic rất phổ biến trên toàn thế giới. Nó được chơi trên một ma trận kích cỡ NxN, trong đó người chơi phải điền các con số từ 1 -> N vào các ô sao cho mỗi hàng, mỗi cột hay mỗi hộp kích cỡ $\sqrt{N}$ x$\sqrt{N}$ đều chứa các ô từ 1->N thỏa mãn các luật sau:

- Mỗi ô chỉ chứa duy nhất một số từ 1->N.
- Mỗi số chỉ xuất hiện duy nhất một lần trong mỗi hàng, mỗi cột.
- Mỗi số chỉ xuất hiện duy nhất một lần trong mỗi hộp

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Hình 1.9: Trò chơi logic Sudoku

Mã hóa biến

Ở đây chỉ sử dụng duy nhất biến X_{ijk} để mã hóa cho ràng buộc về giá trị của mỗi ô trong ma trận Sudoku.

X_{ijk}: Biểu diễn giá trị k có xuất hiện hay không tại ô có hàng i, cột j.

X_{ijk} = true: Giá trị k được điền vào ô (i,j).

X_{ijk} = false: Giá trị k không được điền vào ô (i,j).

Mã hóa các luật của Sudoku

Luật 1: *mỗi ô chỉ nhận một giá trị duy nhất.*

Với luật này, chúng ta có thể chia thành hai luật nhỏ sau:

1.1: Mỗi ô nhận ít nhất một giá trị từ 1->N.

Công thức CNF: $X_{ij1} \vee X_{ij2} \vee X_{ij3} \vee \dots \vee X_{ijN}$

Ví dụ: $X_{111} \vee X_{112} \vee X_{113} \vee \dots \vee X_{119}$

Điều đó có nghĩa là với ma trận Sudoku kích cỡ 9x9 tại ô (1,1) chỉ nhận tối thiểu một giá trị từ 1->N.

1.2: Mỗi ô nhận nhiều nhất một giá trị từ 1->N.

Công thức CNF: $(\neg X_{ij1} \vee \neg X_{ij2}) \wedge (\neg X_{ij1} \vee \neg X_{ij3}) \wedge \dots \wedge (\neg X_{ij1} \vee \neg X_{ijN})$

Ví dụ: $\neg X_{111} \vee \neg X_{112}$

Điều này có nghĩa là: Tại ô (1,1) nó chỉ nhận được một trong hai giá trị là 1 hoặc 2 chứ không thể nhận được cả hai giá trị đó.

Luật 2: *Mỗi số chỉ xuất hiện duy nhất một lần trong mỗi hàng.*

Tương tự với SAT Encoding cho mỗi ô, chúng ta sẽ chia luật này làm hai luật sau:

2.1: Mỗi số xuất hiện ít nhất một lần trong mỗi hàng.

Công thức CNF: $X_{i1k} \vee X_{i2k} \vee X_{i3k} \vee \dots \vee X_{iNk}$

Ví dụ: với ma trận 9x9, tại hàng thứ nhất (hàng 1), luật này được áp dụng như sau:

$X_{111} \vee X_{i21} \vee X_{131} \vee \dots \vee X_{191}$

Điều đó có nghĩa là tại hàng thứ nhất giá trị 1 xuất hiện tối thiểu một lần.

2.2: Mỗi số xuất hiện nhiều nhất một lần trong mỗi hàng.

Công thức CNF: $(\neg X_{i1k} \vee \neg X_{i2k}) \wedge (\neg X_{i1k} \vee \neg X_{i3k}) \wedge \dots \wedge (\neg X_{i1k} \vee \neg X_{iNk})$

Ví dụ: $\neg X_{111} \vee \neg X_{121}$

Điều này có nghĩa là tại hàng 1 của ma trận Sudoku, giá trị 1 không thể xuất hiện cùng lúc tại cột 1 và cột 2.

Luật 3: *Mỗi số chỉ xuất hiện duy nhất một lần trong mỗi cột.*

Việc SAT Encoding cho luật này tương tự như luật 2 trình bày ở trên, tuy nhiên điểm khác ở đây là thay vì cố định các hàng để mã hóa, chúng ta sẽ mã hóa cố định trong mỗi cột, như vậy hai công thức mệnh đề CNF tương ứng sẽ là:

3.1: Mỗi số xuất hiện ít nhất một lần trong mỗi cột.

Công thức CNF: $X_{1jk} \vee X_{2jk} \vee X_{3jk} \vee \ldots \vee X_{Njk}$

Ví dụ với ma trận 9x9, tại cột thứ nhất, luật này được áp dụng như sau:

$X_{111} \vee X_{211} \vee X_{311} \vee \ldots \vee X_{911}$

Điều này có nghĩa là tại cột thứ nhất giá trị 1 xuất hiện tối thiểu một lần.

3.2: Mỗi số xuất hiện nhiều nhất một lần trong mỗi cột.

Công thức CNF: $(\neg X_{1jk} \vee \neg X_{2jk}) \wedge (\neg X_{1jk} \vee \neg X_{3jk}) \wedge \ldots \wedge (\neg X_{1jk} \vee \neg X_{Njk})$

Ví dụ: $\neg X_{111} \vee \neg X_{211}$

Điều này có nghĩa tại cột 1 của ma trận Sudoku giá trị 1 không thể xuất hiện cùng lúc tại hàng 1 và hàng 2.

Luật 4: *Mỗi số chỉ xuất hiện duy nhất một lần trong mỗi hộp.*

Ở đây sẽ lấy ví dụ với ma trận Sudoku kích cỡ 9x9, khi đó mỗi hộp sẽ có kích cỡ là 3x3.

4.1: Mỗi số xuất hiện ít nhất một lần trong mỗi hộp.

Công thức CNF: $X_{11k} \vee X_{12k} \vee X_{13k} \vee \ldots \vee X_{33k}$

4.2: Mỗi só xuất hiện nhiều nhất một lần trong mỗi hộp.

Công thức CNF: $(\neg X_{11k} \vee \neg X_{12k}) \wedge (\neg X_{11k} \vee \neg X_{13k}) \wedge \ldots \wedge (\neg X_{11k} \vee \neg X_{33k})$

Ngoài việc SAT Encoding cho 4 luật chơi của Sudoku nêu trên, chúng ta sẽ phải Encoding cho những ô số đã được điền sẵn trong ma trận đầu vào Sudoku đó. Khi đó công thức mệnh đề CNF cho việc SAT Encoding này sẽ là:

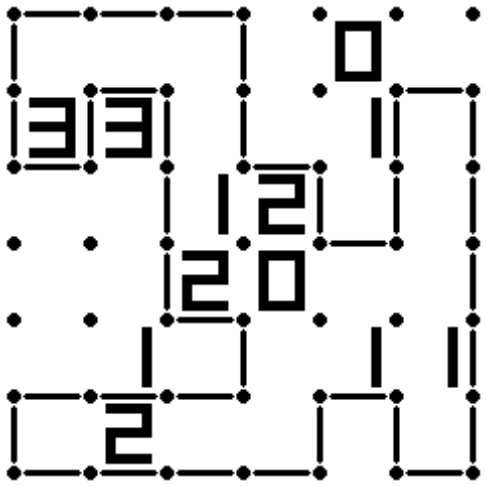
$$X_{ijk}$$

Trong đó ô (i,j) đã có giá trị được điền vào sẵn trước đó là k.

Bằng việc áp dụng SAT Encoding cho từng luật chơi của Sudoku nêu trên, chúng ta sẽ dễ dàng giải quyết bài toán Sudoku đưa ra.

1.2.3.2 Một số ứng dụng khác:

Slitherlink [19]



Hình 1.10: Trò chơi logic Slitherlink

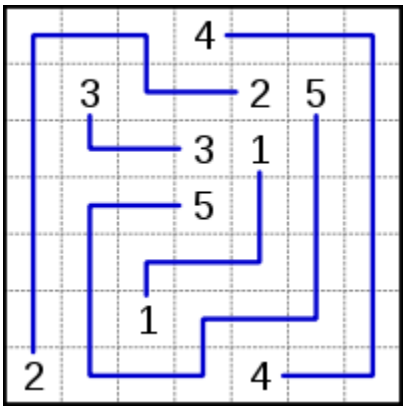
Slitherlink là một trò chơi logic mà ở đó người chơi phải tìm kiếm một đường đi khép kín duy nhất sao cho thỏa mãn bốn luật chơi sau đây:

- Với mỗi ô có trọng số là 0: Đường đi không được đi qua cạnh nào chưa ô có trọng số là 0.
- Với mỗi ô có trọng số là 1: Đường đi chỉ được đi qua duy nhất một cạnh của ô có trọng số là 1.
- Với mỗi ô có trọng số là 2: Đường đi phải đi qua 2 cạnh của ô có trọng số là 2.
- Với mỗi ô có trọng số là 3: Đường đi phải đi qua 3 cạnh của ô có trọng số là 3.

Với 4 luật chơi nêu trên, bằng việc áp dụng SAT Encoding chúng ta sẽ dễ dàng tìm ra đáp án cho trò chơi logic này.

Nhận xét: Đối với slitherlink, việc Encoding khó nhất chính là phải Encoding sao cho đường đi tìm được là đường đi đơn, duy nhất khép kín.

Numberlink [12]



Hình 1.11: Trò chơi logic Numberlink

Numberlink cũng là một trò chơi logic phổ biến ở Nhật Bản, nó bao gồm M x N các ô vuông đơn vị, trong đó luôn tồn tại các cặp ô có chứa các số bằng nhau. Numberlink có ba luật chơi, cụ thể là:

- Các ô chứa các số bằng nhau phải được kết nối với nhau và liên tục.
- Đường kết nối giữa 2 ô phải đi qua tâm của các ô số, theo chiều ngang, chiều dọc, hoặc thay đổi hướng và không bao giờ đi qua hai lần thông qua các ô tương tự.
- Các đường đi không được cắt nhau hoặc đi ngang qua các ô chứa số.

Bằng việc giải thích các luật của Numberlink như trên, chúng ta hoàn toàn dễ dàng áp dụng phương pháp SAT Encoding vào để giải quyết trò chơi này.

Trên đây là những ví dụ điển hình cho việc áp dụng phương pháp SAT Encoding để giải quyết những bài toán, trò chơi, câu đố có tính logic. Tuy nhiên để giải quyết những vấn đề đó bằng SAT Encoding, chúng ta sẽ phải thực hiện theo quy trình.

Chương 2. Các phương pháp giải trò chơi logic Hitori

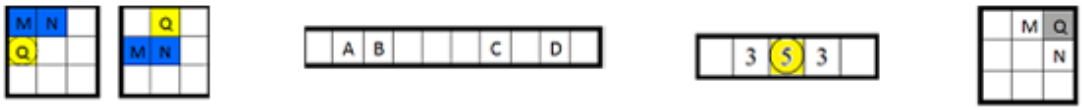
Phần này khóa luận sẽ trình bày lại những phương pháp giải trò chơi lôgic Hitori từ tìm kiếm theo mẫu kết hợp quay lui tới những phương pháp giải bằng SAT Encoding. Đặc biệt còn đề xuất một phương pháp mới (CE^+) sử dụng kết hợp các mẫu tìm kiếm và *Connectivity Encoding* để giải.

2.1 Thuật toán tìm kiếm

Thuật toán tìm kiếm theo mẫu hay *Search With Patterns* – SWP với 8 mẫu này được giới thiệu vào năm 2009 bởi Shi-Jim Yen và cộng sự [15].

Thuật toán gồm 3 bước sau:

- **Bước 1**: Thực hiện tìm kiếm các mẫu cơ bản, các trường hợp tìm ra các ô chắc chắn bị xóa đi hay giữ lại

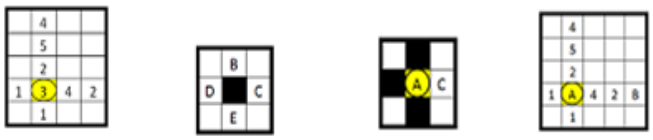


Hình 2.1: Một số mẫu cơ bản

Với các mẫu trên:

- Mẫu thứ nhất: *Nếu M=N thì Q phải được để trắng (không được tô đen)*.
- Mẫu thứ hai**:** *Nếu A=B và A, B nằm cạnh nhau thì các số giống với A, B khác phải bị xóa (C, D bị tô đen)*
- Mẫu thứ ba: *Nếu có một số nằm giữa hai số bằng nhau thì số đó phải được để trắng.*
- Mẫu thứ tư: *Nếu M=N=Q và chúng có vị trí như hình, thì Q phải bị tô đen*

- **Bước 2**: Khi các mẫu ở bước 1 không đủ để giải quyết Hitori thì sang bước hai, thuật toán đưa ra một số các mẫu phức tạp hơn.



Hình 2.2: Các mẫu nâng cao

Cụ thể áp dụng như sau:

- *Nếu một số là duy nhất trong cả hàng và cột thì nó phải được để trắng.*
- *Nếu một ô đã bị bôi đen thì các ô liền kề cạnh với nó phải được để trắng (để đảm bảo luật 2).*
- *Nếu một ô nằm kề cạnh với ba ô đã bị xóa thì nó và ô kề cạnh còn lại phải để trắng.*
- *Nếu trong hàng hoặc cột đã có một ô được xác định là để trắng thì những ô khác bằng nó trong hàng hoặc cột phải bị tô đen.*

- **Bước 3**: Trong những ô còn lại, thực hiện đệ quy. Ở mỗi lần đệ quy, ta giả sử một ô chưa xác định trạng thái là được giữ lại, sau đó sử dụng các mẫu nâng cao để suy ra trạng thái của các ô khác, kiểm tra xem trạng thái được suy ra có vi phạm điều kiện về liên thông và liền kề hay không, nếu có ta quay lui và đánh dấu lại trạng thái của ô vừa được giả sử được giữ lại là bị xóa đi, nếu không ta tiếp tục với ô tiếp theo. Cứ như thế cho đến khi tất cả các ô đều được xác định trạng thái.

2.2 Các thuật toán sử dụng SAT Encoding

Phương pháp SAT Encoding giải quyết những bài toán dạng này, thường được thực hiện theo thứ tự các bước trong phần giới thiệu về SAT Encoding.

Phần này khóa luận sẽ trình bày thực hiện bước thứ 2 (phân tích bài toán về lôgic mệnh đề và chuyển về dạng CNF) với các cách khác nhau, phần này là phần chính quyết định đến chất lượng của một trình giải quyết Hitori bằng SAT Encoding khi cài đặt và cũng là phần khó khăn nhất.

Các biến được sử dụng:

Biến W_{ij} có giá trị "true" quy định ô (i,j) không bị tô đen, ngược lại ô (i,j) bị tô đen.

Chú ý:

- Khi viết W_{ij}^{k} là đang nói ô (i,j) có giá trị k và không được tô đen.
- Nói đường đi giữa các ô theo kiểu của ô trắng, tức là ô này đi tới ô tiếp theo theo chiều dọc hoặc ngang với ô ngay kề nó, nói đường đi giữa các ô theo kiểu của ô đen tức là ô này đi tới ô tiếp theo theo chiều chéo nhau với ô chéo sát góc với nó.

Luật 1: *Giá trị của số trong mỗi ô không được xuất hiện nhiều hơn một lần ở mỗi hàng, mỗi cột*

Trong một hàng hoặc một cột, nếu thấy hai ô có cùng giá trị thì ít nhất một trong hai ô đó phải được xóa đi:

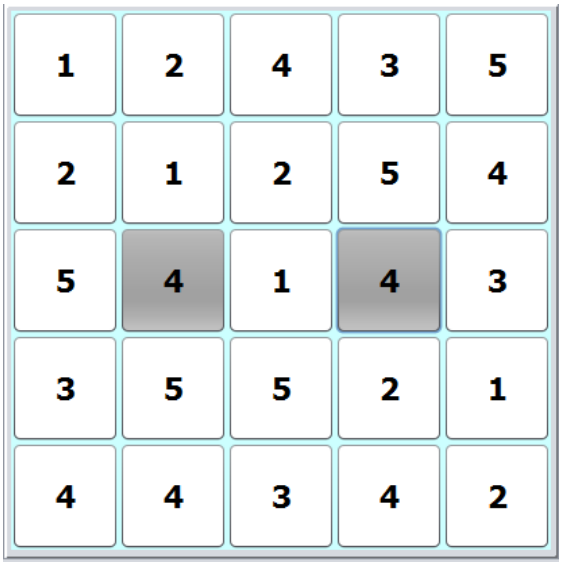
Cùng hàng i: $\neg W_{ij}^{k} \vee \neg W_{i',j}^{k}$

Cùng hàng j: $\neg W_{ij}^{k} \vee \neg W_{ij',}^{k}$

Ví dụ:

Chỉ tính riêng hai ô có cùng giá trị được tô màu xám ta có mệnh đề

$$\neg W_{21} \vee \neg W_{23}$$



Hình 2.3: Hai ô có cùng giá trị trong cùng một hàng

Tương tự trong một hàng hoặc cột có nhiều ô có cùng giá trị thì chỉ được một ô duy nhất trong các ô đó không bị tô đen, còn lại tô đen hết.

Ví dụ ba ô cùng hạt hoặc cột W_1, W_2, W_3 (không ghi chỉ số hàng cột, thay vào đó đánh số):

$$W_1^{k} \wedge W_2^{k} \wedge W_3^{k} \rightarrow (W_1^{k} \wedge \neg W_2^{k} \wedge \neg W_3^{k}) \vee (\neg W_1^{k} \wedge W_2^{k} \wedge \neg W_3^{k}) \vee (\neg W_1^{k} \wedge \neg W_2^{k} \wedge W_3^{k})$$

Ngoài ra tình huống nhiều ô trong hàng hoặc cột có cùng giá trị cũng có thể xử lý bằng cách từng đôi một hai ô có cùng giá trị không thể cùng trắng

$$W_1^k \wedge W_2^k \wedge W_3^k \rightarrow (\neg W_1^k \vee \neg W_2^k)$$
$$\wedge (\neg W_2^k \vee \neg W_3^k)$$
$$\wedge (\neg W_1^k \vee \neg W_3^k)$$

Tương tự với nhiều ô cùng hàng hoặc cột có cùng giá trị hơn.

Ví dụ:

4	1	6	2	5	7	7
5	3	7	2	6	4	1
3	7	7	1	7	2	6
7	3	1	4	7	3	2
2	6	1	7	6	1	2
1	6	3	6	2	7	4
7	4	6	5	2	6	5

Hình 2.4: Ba ô có cùng giá trị trong cùng một hàng

Chỉ xét riêng nhưng ô có cùng giá trị là 7 và được tô màu xám thì có mệnh đề sau:

$$(W_{21} \wedge \neg W_{22} \wedge \neg W_{24})$$
$$\vee (\neg W_{21} \wedge W_{22} \wedge \neg W_{24})$$
$$\vee (\neg W_{21} \wedge \neg W_{22} \wedge W_{24})$$

Hoặc có thể biểu diễn như sau:

$$(\neg W_{21} \vee \neg W_{22})$$
$$\wedge (\neg W_{22} \vee \neg W_{24})$$
$$\wedge (\neg W_{21} \vee \neg W_{24})$$

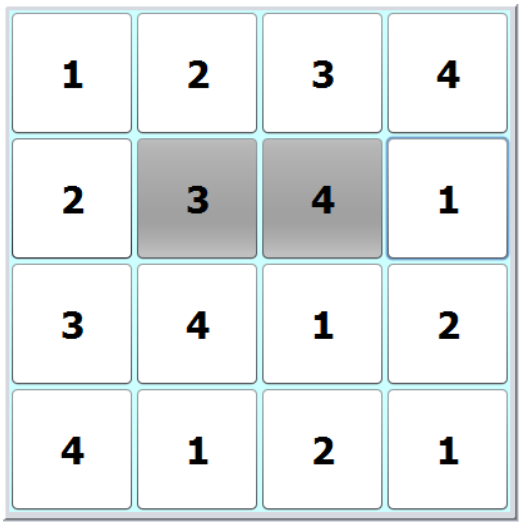
Luật 2: *Các ô bôi đen không được nằm liền kề nhau trên hàng hoặc cột*

Tức là hai ô kề nhau theo hàng hay cột thì ít nhất phải có một ô trắng:

Trên cùng cột: $W_{ij} \vee W_{i+1j}$

Trên cùng hàng: $W_{ij} \vee W_{ij+1}$

Ví dụ:

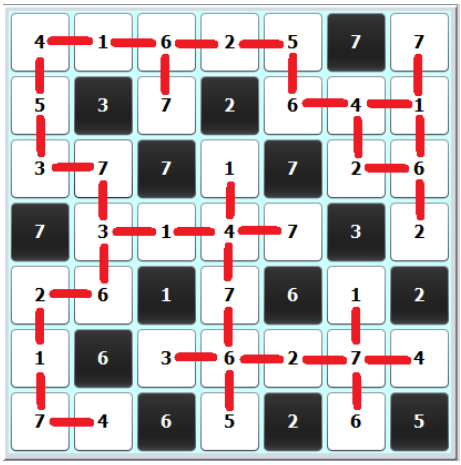


Hình 2.5: Hai ô bất kì cạnh nhau không được cùng đen

Xét riêng với hai ô sát nhau được tô màu xám như trên hình thì có mệnh đề:

$$W_{11} \vee W_{12}$$

Luật 3: *Mọi ô không bôi đen (ô trắng) phải kết nối được với nhau. Nói cách khác luôn tồn tại một đường đi từ mọi ô trắng tới các ô trắng còn lại.*



Hình 2.6: Các ô trắng trong ma trận phải có đường đi với nhau

Với luật 3 này lại có nhiều cách cài đặt. Qua thực nghiệm thì luật 3 này sẽ tiêu tốn nhiều tài nguyên nhất khi giải quyết một bài toán Hitori, vì thế với mỗi cách encoding cho luật 3 lại cho một hiệu năng khác nhau. Sau đây là tổng hợp các phương pháp từ trước tới nay:

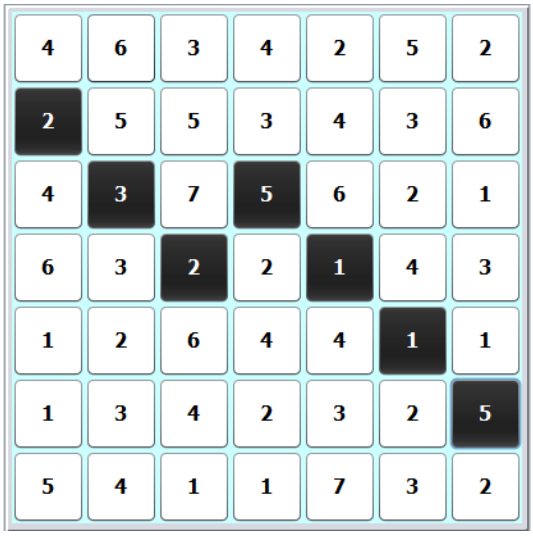
2.2.1 Chains and Cycles Encoding - CC

Phương pháp này được giới thiệu vào năm 2006 [5], đây là cách dễ hiểu nhất, cơ bản nhất nhưng lại tạo ra nhiều mệnh đề nhất.

Mọi ô trắng đều có thể kết nối với nhau cũng có thể hiểu không có hai ô trắng nào không tồn tại đường đi giữa chúng (đường đi ở đây là đường nối giữa những ô theo chiều dọc hoặc ngang). Nghĩa là sẽ không tồn tại vùng một hoặc các ô trắng liền kề nhau bị bao quanh bởi những ô đen. Có hai loại đường bao quanh cụm ô trắng đó gọi là Chain và Cycle đúng như tên của phương pháp này.

Chain là đường nối những ô đen (theo chiều chéo nhau) được bắt đầu bởi một ô kề với cạnh của ma trận đề bài và kết thúc cũng bằng một ô khác kề với ma trận đề bài, nó chia ma trận ra làm 2 vùng khiến cho các ô trắng từ vùng này không thể có đường đi (theo kiểu ngang hoặc dọc của ô trắng) từ vùng này sang vùng khác.

Ví dụ một Chain gồm n ô đen liền nhau W_1 W_2 W_3 … W_n

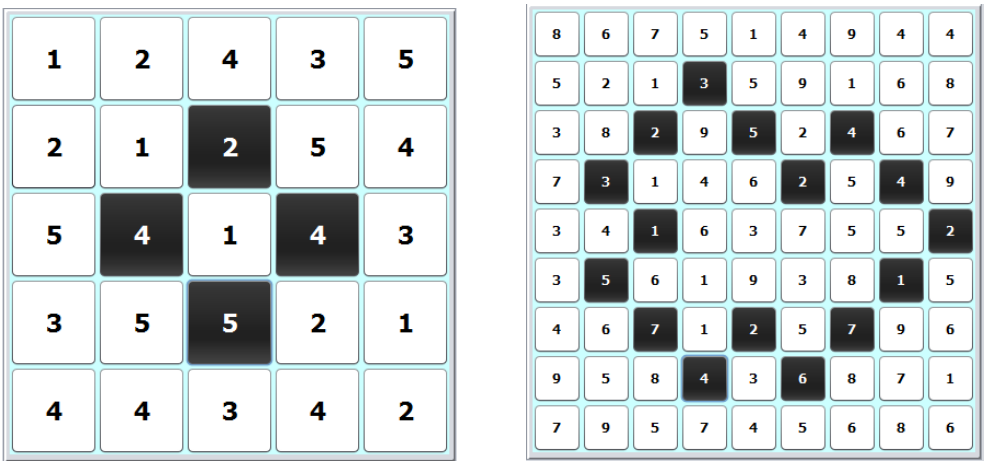


Hình 2.7: Một Chain cơ bản

Cycle là một vòng được tạo bởi những ô đen liền nhau (theo chiều chéo nhau) vòng này cũng chia ma trận làm hai vùng, một vùng bên trong và một vùng bên ngoài vòng, và

các ô trắng bên trong với các ô trắng bên ngoài vòng thì không có đường đi nối chúng với nhau (đường đi theo kiểu của các ô trắng).

Ví dụ một Cycle gồm có n ô đen W_1 W_2 W_3 … W_n



Hình 2.8: Thí dụ về Cycle

Chains và Cycles nếu tồn tại sẽ làm cho kết quả bài toán không chuẩn nữa. Vậy loại hết các Chain và Cycle có thể tồn tại, bài toán sẽ đúng.

Một Chain hoặc Cycle bất kì có n ô W_1 W_2 W_3 … W_n sẽ tồn tại ít nhất một ô trắng

$$W_1 \vee W_2 \vee W_3 \vee \ldots \vee W_n$$

Mệnh đề tương tự như vậy cho tất cả các Chain và Cycle khác có thể tồn tại.

Số lượng các Chain và Cycle cực kì lớn và lại càng lớn hơn rất nhiều khi kích cỡ ma trận tăng, kéo theo số lượng mệnh đề sẽ rất lớn. Điều này sẽ thấy rõ hơn trong phần thực nghiệm.

2.2.2 Connect from 1st Cell – C1st

Tác giả của phương pháp này là Pedro Cabalar Phó giáo sư Khoa học máy tính Đại học Coruna, Galicia, Tây Ban Nha đăng tải Hitori.c trên trang cá nhân của trường ĐH vào năm 2014 [13].

Phương pháp này sử dụng thêm một biến mới $P_{ij,k}$ có giá trị "true" thể hiện ô trắng (i,j) có đường đi tới ô trắng đầu tiên (theo kiểu của ô trắng). Giá trị của k cực đại theo lý thuyết sẽ là m x n – 1 nếu tính như k bắt đầu từ 0 và ma trận có kích cỡ m x n. Ta gọi maxk = m x n -1.

Từ luật 2, hai ô liền nhau luôn có ít nhất một ô trắng, vì vậy ô trắng đầu tiên trong ma trận sẽ là ô (0,0) hoặc ô (0,1)

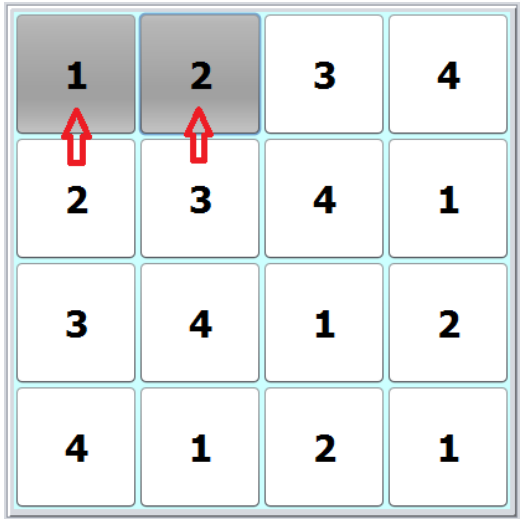
$$W_{00} \rightarrow P_{00,0}$$

$$W_{01} \rightarrow P_{01,1}$$

Tại sao mệnh đề thứ 2 bên trên không phải là $W_{01} \rightarrow P_{01,0}$ .Nếu cả hai ô (0,0) và (0,1) đều là ô trắng thì ta sẽ có $P_{00,0}$ và $P_{01,0}$, điều này bất hợp lý vì rõ ràng độ dài đường đi từ ô (0,1) tới ô (0,0) sẽ là 1 vì ô (0,0) mới là ô trắng đầu tiên.

Nếu một ô là trắng thì chắc chắn tồn tại đường đi từ ô đó tới ô trắng đầu tiên:

$$W_{ij} \rightarrow P_{ij,1} \vee P_{ij,2} \vee \ldots \vee P_{ij,maxk}$$



Hình 2.9: Hai ô đầu tiên chắc chắn có ít nhất một ô là trắng

Nếu một ô có đường đi tới ô trắng đầu tiên độ dài là k (tất nhiên ô này là ô trắng) thì tồn tại một ô liền kề với nó theo chiều dọc hoặc ngang cũng là ô trắng và có đường đi tới ô trắng đầu tiên và đường đi có độ dài k-1:

$$P_{ij,k} \leftrightarrow P_{i-1j,k-1} \vee P_{i+1j,k-1} \vee P_{ij-1,k-1} \vee P_{ij+1,k-1}$$

Điều ngược lại cũng đúng. Điều này giúp không có ô nào vừa có đường đi lại không có đường đi tới ô trắng đầu tiên, nhưng một ô trắng có thể có nhiều đường đi tới ô trắng đầu tiên, hay tồn tại nhiều giá trị của k mà $P_{ij,k}$, không sao cả, với những mệnh đề trên luật 3 vẫn được thỏa mãn.

2.2.3 Connectivity Encoding - CE

Mục đích của phương pháp này là làm giảm số lượng mệnh đề trong công thức CNF so với phương pháp *Chains and Cycle***.** Tư tưởng chính của *Connectivity Encoding* vẫn là tìm cách để loại bỏ đi các *Chain* và *Cycle* trong ma trận, tuy nhiên thay vì tìm kiếm toàn bộ số lượng *Chain và Cycle* đó, phương pháp xây dựng việc *liên kết chéo* giữa các ô được tô đen và đảm bảo việc liên kết chéo giữa các ô đen sẽ không tạo thành *kết nối giữa các ô biên* (*Chain*) và *tạo thành đường khép kín* (Cycle) [2].

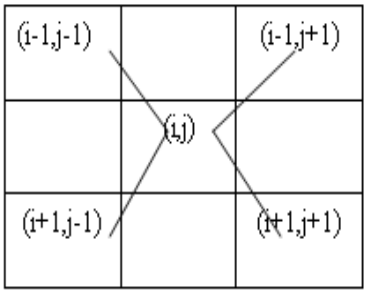
Để xây dựng kết nối chéo trong các ô đen chúng ta sử dụng thêm biến Lôgic mệnh đề $C_{ij,ab}$, nếu biến này bằng *true*, có nghĩa là 2 ô đen *(i,j)* và *(a,b)* có kết nối chéo với nhau hay nói cách khác là có đường đi chéo từ ô *(i,j)* đến ô *(a,b)*.

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

Hình 2.10: Liên kết

Trong hình 2.10, liên kết chéo xuất hiện, $C_{33,35}$ = *true* (hai ô đen (3,3) và (3,5) trong ma trận có kết nối chéo với nhau).

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



Hình 2.11: Lân cận của một ô

Như trong hình 2.11, ô *(i,j)* đã được tô đen và nó có thể tạo kết nối chéo tới 4 ô lân cận *(i-1, j-1) (i+1, j-1), (i-1, j+1),* và *(i+1, j+1).*

Công thức Lôgic mệnh đề tương ứng là:

$$(W_{ij}^k \wedge W_{i-1j-1}^k \rightarrow \neg C_{ij,i-1j-1} \vee \neg C_{i-1j-1,ij})$$

$$\wedge (W_{ij}^k \wedge W_{i-1j+1}^k \rightarrow \neg C_{ij,i-1j+1} \vee \neg C_{i-1j+1,ij})$$

$$\wedge (W_{ij}^k \wedge W_{i+1j-1}^k \rightarrow \neg C_{ij,i+1j-1} \vee \neg C_{i+1j-1,ij})$$

$$\wedge (W_{ij}^k \wedge W_{i+1j+1}^k \rightarrow \neg C_{ij,i+1j+1} \vee \neg C_{i+1j+1,ij})$$

Lưu ý: Quy định kết nối chéo giữa các ô đen là kết nối có hướng, do đó $c_{ij,kl}$ và $c_{kl,ij}$ là hoàn toàn khác nhau *(mục đích là để loại bỏ các Cycle - được trình bày ở phần tiếp theo)*. Công thức Lôgic mệnh đề cho 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)*. Tương tự với các ô lân cận còn lại.

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

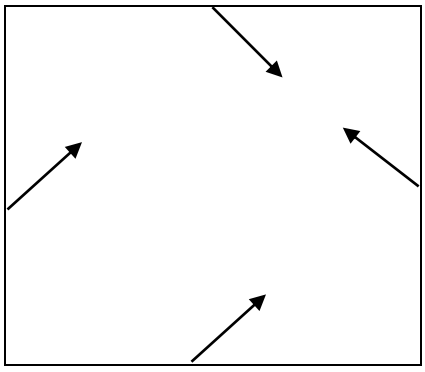
Kết nối chéo có tính chất bắc cầu và mệnh đề Lôgic tương ứng là:

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

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 suy ra có kết nối chéo từ ô *(i,j)* tới ô *(m,n).*

Mã hóa loại bỏ Chains với Connectivity Encoding

Phương pháp xây dựng các mệnh đề để các ô tại biên không được kết nối chéo với nhau.



Hình 2.12: Các ô biên đen luôn có chiều kết nối vào trong ma trận

Nếu 2 ô *(i,j)* và *(k,l)* là 2 ô biên thì chúng không thể có kết nối chéo với nhau và mệnh đề Lôgic tương ứng là: $\neg C_{ij,kl} \wedge \neg C_{kl,ij}$

Một điểm nữa là hướng kết nối của các ô đen ở biên phải luôn hướng về bên trong ma trận (Hình 2.12).

Mã hóa loại bỏ Cycles với Connectivity Encoding

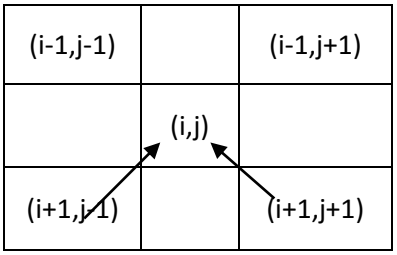
Như đã trình bày ở trên kết nối chéo giữa các ô đen phải có hướng để xây dựng các mệnh đề loại bỏ *Cycles*.

- **Kết nối chéo không được phép hướng vào cùng một ô**

Hình 21 thể hiện ràng buộc được xây dựng. Ràng buộc này quy định không cho phép 2 ô bất kỳ có kết nối chéo từ nó đến cùng một ô. Việc đưa ra ràng buộc này sẽ giúp quy định các đường đi trong một Cycle là đường đi ***có hướng thống nhất*** *(các hướng đi luôn là một chiều)* và giúp loại bỏ các khả năng tạo thành *Chain*. Mệnh đề Lôgic tương ứng sẽ là:

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

Áp dụng công thức tương tự cho các ô lân cận còn lại.



Hình 2.13: Không cho phép kết nối có hướng theo cách này

- **Không tạo kết nối đối với chính nó để không tạo thành Cycle**

Ràng buộc này quy định không tồn tại kết nối chéo có hướng khép kín *(tạo thành Ciyle)*, nghĩa là không tồn tại một đường đi có hướng từ ô *(i,j)* đến chính nó. Mệnh đề Lôgic tương ứng sẽ là: $\neg C_{ij,ij}$

2.2.4 Connectivity Encoding Plus - CE^+

Với mỗi phương pháp nêu trên có một số ưu và nhược điểm riêng. Như thuật toán tìm kiếm (Tìm kiếm theo mẫu – Search With Pattern) các mẫu cơ bản được xác định rất nhanh nhưng những ô không thể xác định (trắng hoặc đen) sẽ phải thực hiện đệ quy nhiều lần, các thuật toán sử dụng SAT Encoding sinh ra rất nhều mệnh đề cho toàn bộ ma trận đầu vào, nhưng tính ổn định cao và với những cách cài đặt tốt như *Connectivity Encoding* số lượng mệnh đề đã được cải thiện đáng kể. Kết hợp các phương pháp lại để sử dụng những ưu điểm của từng phương pháp đã cho ra ý tưởng về một phương pháp mới đó là *Connectivity Encoding Plus*.

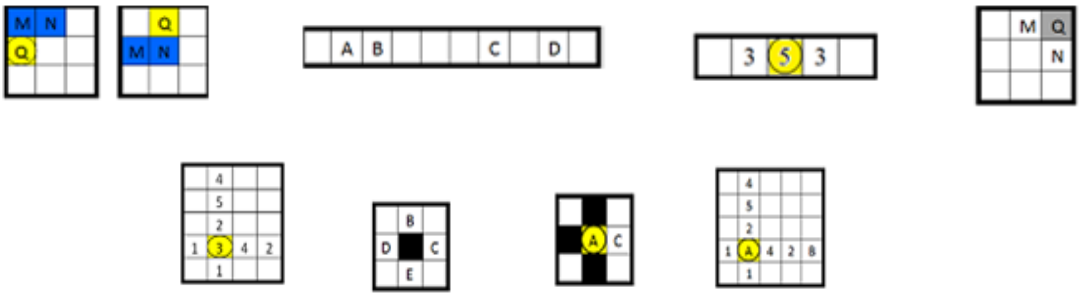
Phần này khóa luận sẽ trình bày về phương pháp được đề xuất mang tên *Connectivity Encoding Plus* (CE^+) với việc dùng *Connectivity Encoding* làm nòng cốt chính, thêm vào đó tận dụng ưu điểm của phương pháp tìm kiếm giúp tối ưu dữ liệu đầu vào (chỉ tính riêng dữ liệu đầu vào cho luật 3 vì luật 1 và 2 hầu như các phương pháp đều giống nhau) theo các mẫu ngoài ra phương pháp còn được tối ưu hơn nữa để giảm đi số lượng biến và mệnh đề. Các tối ưu này sẽ được trình bày rõ trong phần sau.

Đối với những bài toán sử dụng SAT Encoding độ hiệu quả của một phương pháp phụ thuộc phần lớn vào số lượng biến sử dụng và số mệnh đề được sinh ra, do đó để cải thiện tốc độ cho một phương pháp nào đó sử dụng SAT Encoding ta cần phải giảm được số lượng biến và mệnh đề sử dụng cho phương pháp. Phần này trình bày cải tiến phương pháp Connectivity Encoding dựa trên hai ý tưởng sau:

- *Tối ưu dữ liệu đầu vào bằng mẫu tìm kiếm*

Với phương pháp tìm kiếm kết hợp quay lui dễ thấy nó gồm hai thành phần xử lý chính là tìm kiếm với các mẫu và quay lui, quá trình tìm kiếm diễn ra rất nhanh chóng cho thấy hiệu quả tốt của việc áp dụng các mẫu tìm kiếm để xác định nhanh một số ô (là trắng hay đen). Thực nghiệm cho thấy quá trình quay lui lại không thể đạt được tốc độ tốt như vậy. Đối với phương pháp SAT Encoding, cụ thể là với Connectivity Encoding (phương pháp SAT Encoding cho hiệu quả tốt nhất nếu chưa tính phương pháp được đề xuất này) toàn bộ ma trận sẽ được tiền xử lý, quá trình này tạo ra những mệnh đề logic. Sử dụng các mẫu tìm kiếm để tìm kiếm nhanh những ô là trắng hay đen sẽ giúp khâu tiền xử lý không phải thao tác trên toàn bộ ma trận đầu vào nữa. Khâu đệ quy quay lui trước kia tốn nhiều thời gian nay sẽ được thay thế bằng cách sử dụng SAT Encoding cho những ô trong ma trận đầu vào không thể xác định là trắng hoặc đen.

Với việc sử dụng 8 mẫu tìm kiếm như trong thuật toán tìm kiếm (phần 2.1, 8 mẫu tìm kiếm như hình 2.14) sẽ cho ta biết chắc chắn rằng một số ô là trắng hay đen. Những ô mà ta biết trắng hay đen này được tìm kiếm rất nhanh bằng cách lặp đi lặp lại việc rà soát các mẫu trong ma trận đầu vào, nhưng những ô còn lại mà các mẫu tìm kiếm vẫn chưa xác định được chúng chắc chắn phải tô màu trắng hay đen, ta gọi là những ô chưa biết (CsNF), cách duy nhất để xác định chúng theo như thuật toán tìm kiếm đó là đệ quy và thử từng ô là trắng hoặc đen, cũng kết hợp tìm kiếm bằng những mẫu trên. Sẽ không còn đơn giản nếu như một ma trận nào đó sau khi tìm kiếm cho số ô chưa biết là lớn, thuật toán đệ quy sẽ rất lâu và tốn tài nguyên trên nhiều dữ liệu cần xử lý, hơn nữa mỗi lần tìm ra kết quả của đệ quy, lại phải làm một thao tác kiểm tra, việc này lại làm tăng thời gian để chúng ta tìm ra kết quả.



Hình 2.14: Các mẫu tìm kiếm được sử dụng

Khóa luận sẽ trình bày việc thay khâu đệ quy bằng SAT Encoding để đạt hiệu quả hơn, cũng có thể nói dữ liệu đầu vào trước khi sử dụng Connectivity Encoding sẽ tìm kiếm trước để loại đi những ô đã biết trước là trắng hay đen.

- *Tối ưu dựa vào đặc tính liên kết của những ô đen (hay tối ưu theo vùng - tối ưu theo zone)*

Số lượng biến và mệnh đề được tạo ra để phục vụ cho mệnh đề bắc cầu là rất lớn trong phương pháp Connectivity Encoding (mệnh đề bên dưới).

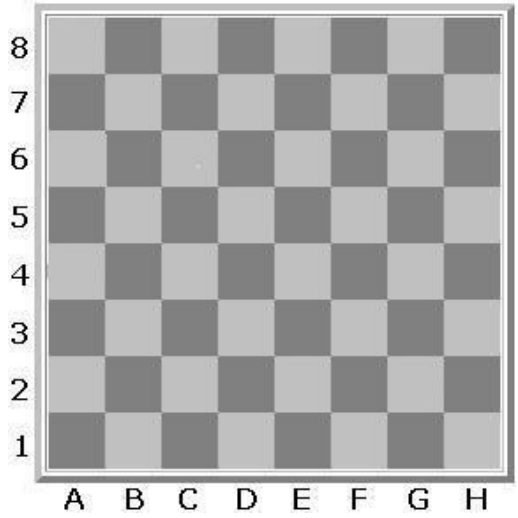
$$C_{ij,hk} \wedge C_{hk,mn} \rightarrow C_{ij,mn} \equiv \neg C_{ij,hk} \vee \neg C_{hk,mn} \vee C_{ij,mn}$$

Do mỗi ô trong ma trận đầu vào đều có một cơi hội có kết nối với mỗi ô khác trong ma trận đầu vào đó (kết nối theo chiều chéo nhau giữa các ô đen) do đó số kết nối là rất

lớn, nhiều kết nối được tạo ra kéo theo việc nhiều mệnh đề được tạo ra để xử lý bắc cầu. Để giảm được số biến và số mệnh đề được tạo ra khóa luận trình bày việc chia ma trận đầu vào thành nhiều vùng xử lý riêng với nhau, trong đó ở mỗi vùng các ô mới có thể có cơ hội tạo liên kết với nhau. Các vùng này là riêng biệt và không thể có cơ hội tạo liên kết giữa một ô trong vùng này với một ô trong vùng kia.

Những vùng này được xác định bởi một tính chất đã được nhắc đến trong phần đầu của chương đối với những ma trận đầu vào đó là những ô multi (những ô có giá trị xuất hiện thêm ít nhất một lần nữa trong cùng hàng hoặc cột) mới có cơ hội tô đen. Những ô multi có thể là không nằm liền sát nhau vì thế liên kết chéo với những ô đen có thể là không liên tục. Việc xác định những ô multi cộng thêm chút tính toán sẽ giúp ta phân chia ma trận đầu vào thành nhiều vùng kết nối riêng biệt, qua đó giảm được số lượng biến và mệnh đề một cách đáng kể.

Nếu đã từng chơi cờ vua, hãy nhớ rằng mỗi bên người chơi luôn có hai quân tịnh, và mỗi quân đều đi theo làn của riêng mình, chúng không bao giờ chéo nhau được. Hay khi nhìn bàn cờ vua có những ô trắng và những ô đen đó, quân tịnh ở ô trắng không bao giờ đi được tới ô đen và ngược lại. Ở đây cũng vậy, chúng ta cũng có ít nhất 2 vùng được tạo ra theo kiểu đi của ô đen trong ma trận Hitori, từ đó việc phân vùng ma trận đầu vào được tăng cường hơn giúp số biến và mệnh đề được giảm đi rất nhiều.



Hình 2.15: Hình ảnh một bàn cờ vua mà 2 vùng liên kết chéo khác biệt nhau

Quá trình tối ưu này được diễn ra như sau:

i) Sử dụng các mẫu tìm kiếm

Xác định nhiều nhất có thể những ô chắc chắn phải được tô trắng hoặc đen trong ma trận đầu vào bằng việc tìm kiếm theo 8 mẫu như hình 2.14. Những ô chưa biết là trắng hay đen còn lại tạo thành những vùng nhỏ bên trong ma trận, vì thế không còn có thể sử dụng được hết những logic mệnh đề của Connectivity Encoding để xử lý những ô này được, chúng có thể không cạnh những ô biên, chúng có liên quan tới những ô đã xác định trắng hay đen bởi những đường mà vẫn quen gọi là Chain và Cycle. Do đó việc dùng từng mệnh đề của Connectivity Encoding đều cần xử lý khéo léo kết hợp với những ô đã xác định là đen.

Xử lý những ô nằm ngay liền kề với vùng mà những ô chưa biết trắng đen tạo thành. Gọi những ô đó là NCs (Neighbouring Cells). Những thông tin cần biết với NCs là từng ô đó có liên kết (theo kiểu của ô đen) với một ô nào khác (cùng gần 1 vùng mà các ô chưa biết) trong NCs không, từng ô trong NCs có đường liên kết (theo kiểu của ô đen) với một ô nào đó gần biên của ma trận không.

ii) Sử dụng tối ưu theo vùng

Những ô chưa được xác định là trắng hay đen bằng các mẫu tìm kiếm sẽ được kiểm tra và xác định xem ô nào là ô multi. Sau đó phân chia những ô multi này thành các vùng riêng biệt.

iii) Sử dụng những mệnh đề của Connectivity Encoding để xử lý những ô multi theo từng vùng riêng biệt.

$$(W_{ij} \wedge W_{i-1j-1} \rightarrow \neg C_{ij,i-1j-1} \vee \neg C_{i-1j-1,ij})$$

$$\wedge (W_{ij} \wedge W_{i-1j+1} \rightarrow \neg C_{ij,i-1j+1} \vee \neg C_{i-1j+1,ij})$$

$$\wedge (W_{ij} \wedge W_{i+1j-1} \rightarrow \neg C_{ij,i+1j-1} \vee \neg C_{i+1j-1,ij})$$

$$\wedge (W_{ij} \wedge W_{i+1j+1} \rightarrow \neg C_{ij,i+1j+1} \vee \neg C_{i+1j+1,ij})$$

Những đường kết nối kiểu này là một chiều:

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

Mệnh đề thể hiện tính chất bắc cầu vẫn như vậy

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

Ở đây sẽ xét cả những ô CsNF và những ô NCs, bởi sự liên kết của những ô CsNF còn phụ thuộc vào những ô NCs gần với chúng có liên kết đã xác định chưa, hoặc có liên kết theo kiểu của ô đen với những ô đen ở biên không. Nếu 2 ô *(i,j)* và *(k,l)* là 2 ô NCs và chúng cùng có đường đi tới một ô biên của ma trận hoặc chúng có kết nối theo kiểu ô đen bởi một đường những ô đen ta đã xác định từ trước thì chúng không thể có kết nối chéo với nhau và mệnh đề Lôgic tương ứng là: $\neg C_{ij,kl} \wedge \neg C_{kl,ij}$.

Ràng buộc này quy định không tồn tại kết nối chéo có hướng khép kín *(tạo thành Cycle)*, nghĩa là không tồn tại một đường đi có hướng từ ô *(i,j)* đến chính nó. Mệnh đề Lôgic tương ứng sẽ là: $\neg C_{ij,ij}$, mệnh đề này vẫn vậy.

Phương pháp này cho hiệu quả rất rõ rệt và sẽ được trình bày trong phần thực nghiệm phía dưới.

Chương 3. Thực nghiệm đánh giá

Thông qua thực nghiệm này, có thể thấy rõ được hiệu quả của phương pháp được đề xuất CE^+ so với các phương pháp khác.

3.1 Môi trường thực nghiệm

3.1.1 Dữ liệu thực nghiệm

Chúng tôi tiến hành thực nghiệm trên 2 bộ cơ sở dữ liệu đầu vào:

- Bộ dữ liệu **BI** (Benchmarks from Internet): là bộ dữ liệu gồm 190 ma trận đầu vào với kích thước ma trận lớn nhất là *30x30*, được lấy từ [7] và các trang web phổ biến Hitori hiện nay.
- Bộ dữ liệu **CB** (Creational Benchmarks): là bộ dữ liệu gồm 110 ma trận đầu vào với kích thước ma trận lên tới *50x50*. Mục đích tạo ra bộ dữ liệu thực nghiệm CB là để thực nghiệm trên những ma trận có kích thước lớn hơn để đánh giá tính hiệu quả của các phương pháp giải Hitori hiện có so với phương pháp CE^+.

3.1.2 Cấu hình phần cứng

Thực nghiệm được tiến hành trên máy tính (laptop) với cấu hình như sau:

- **CPU**: Core i7 5500U 2.40GHz (4CPUs)
- **Bộ nhớ Ram**: 4GB DDR3 1600MHz
- **HDD**: 5400 RPM Serial ATA
- **Hệ điều hành**: Linux Mint 17.2 (Cinnamon 64-bit, *Linux kernel*: 3.16.0-38-generic)
- **Ngôn ngữ lập trình**: java (*java version*: 1.8.0_66)
- **SAT Solver**: minisat 2.2.1

3.2 Kết quả thực nghiệm

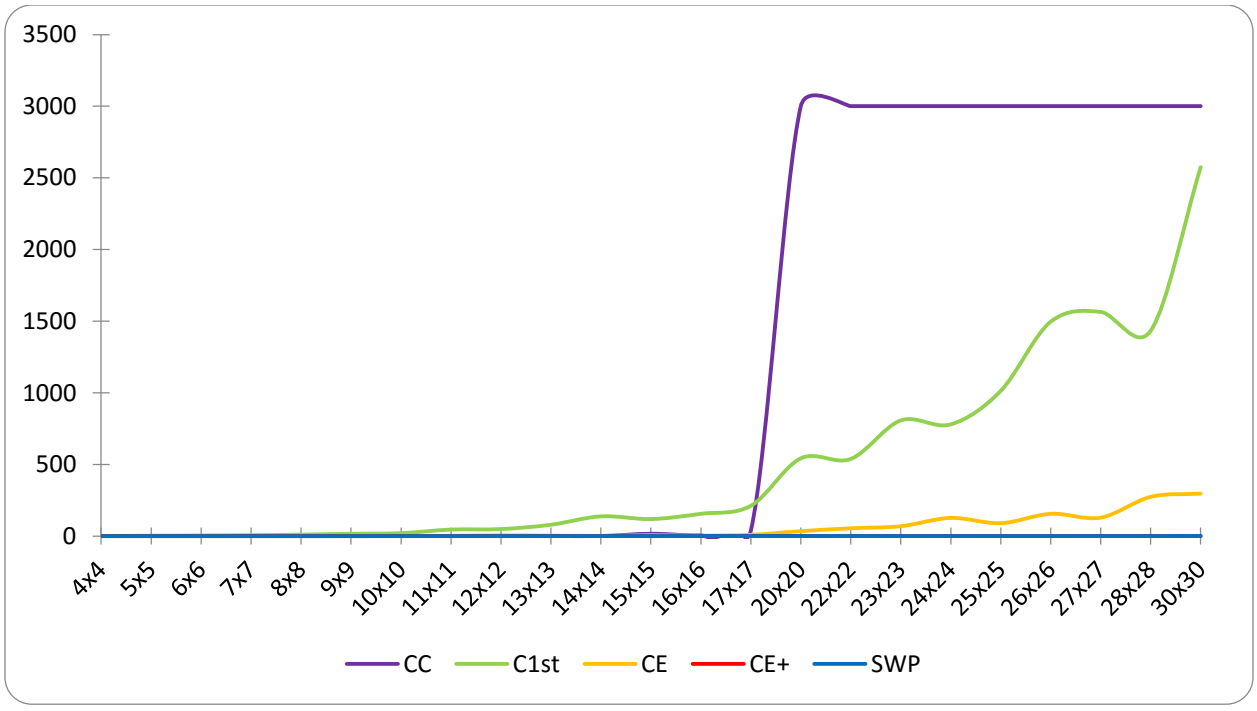
Chúng tôi sẽ tiến hành so sánh ***thời gian chạy*** *(được tính bằng mili giây - ms)****, số lượng biến lôgic và số mệnh đề*** trong từng phương pháp SAT Encoding (đối với phương pháp tìm kiếm thì chỉ đánh giá thời gian chạy). Số lượng biến lôgic và số mệnh đề là những yếu tố chi phối tới tốc độ của phương pháp SAT Encoding. Trong mỗi thực nghiệm chúng tôi tiến hành chạy 3 lần cho một ma trận đầu vào và lấy kết quả trung bình của 3 lần chạy. ***Timeout*** được thiết lập là ***3000 mili giây*** trên toàn bộ thực nghiệm.

3.2.1 Bộ dữ liệu BI

- **Thời gian chạy**

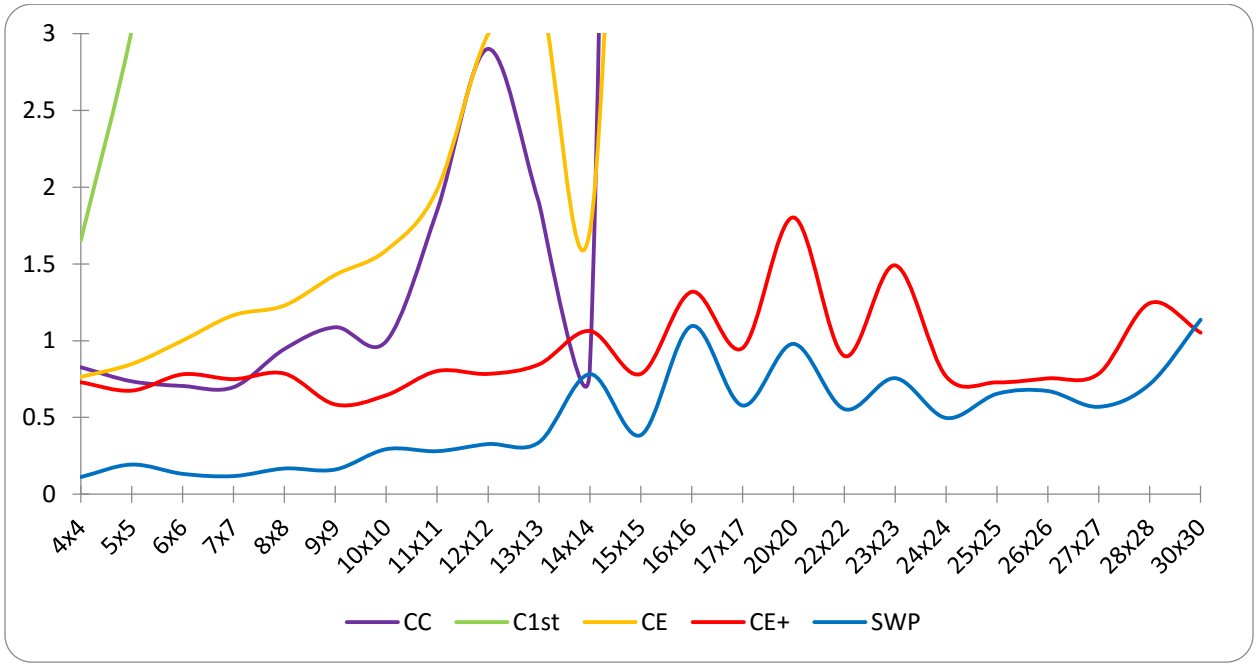
Size	CC	C1st	CE	CE$^+$	SWP
4x4	0.83	1.65	0.76	0.73	0.11
5x5	0.74	3.02	0.85	0.67	0.19
6x6	0.70	4.81	1.00	0.78	0.13
7x7	0.70	6.88	1.17	0.75	0.12
8x8	0.95	10.80	1.23	0.79	0.17
9x9	1.09	16.78	1.43	0.58	0.16
10x10	1.00	21.13	1.59	0.64	0.29
11x11	1.85	45.89	1.98	0.80	0.28
12x12	2.90	49.40	3.00	0.78	0.33
13x13	1.90	79.66	3.26	0.85	0.34
14x14	0.82	138.31	1.71	1.06	0.78
15x15	16.94	119.47	7.07	0.78	0.38
16x16	5.97	155.86	3.97	1.32	1.10
17x17	38.65	211.93	9.38	0.95	0.58
20x20	Timeout	543.26	35.02	1.80	0.98
22x22	Timeout	539.01	54.96	0.90	0.55
23x23	Timeout	807.69	69.51	1.49	0.76
24x24	Timeout	779.97	127.22	0.77	0.50
25x25	Timeout	1013.87	90.55	0.73	0.65
26x26	Timeout	1496.70	155.90	0.75	0.67
27x27	Timeout	1564.92	129.38	0.79	0.57
28x28	Timeout	1433.27	274.15	1.24	0.72
30x30	Timeout	2574.41	297.04	1.05	1.14

Bảng 3.1: So sánh thời gian chạy giữa các phương pháp trên bộ dữ liệu BI



Biểu đồ 3.1: So sánh thời gian chạy tổng quát giữa các phương pháp trên bộ dữ liệu BI

Biểu đồ 3.1 trên đường CE^+ (màu đỏ) bị che lấp bởi đường SWP (đường xanh da trời) do hai đường này nằm rất sát với nhau. Biểu đồ sau sẽ chi tiết hơn cho ta thấy.



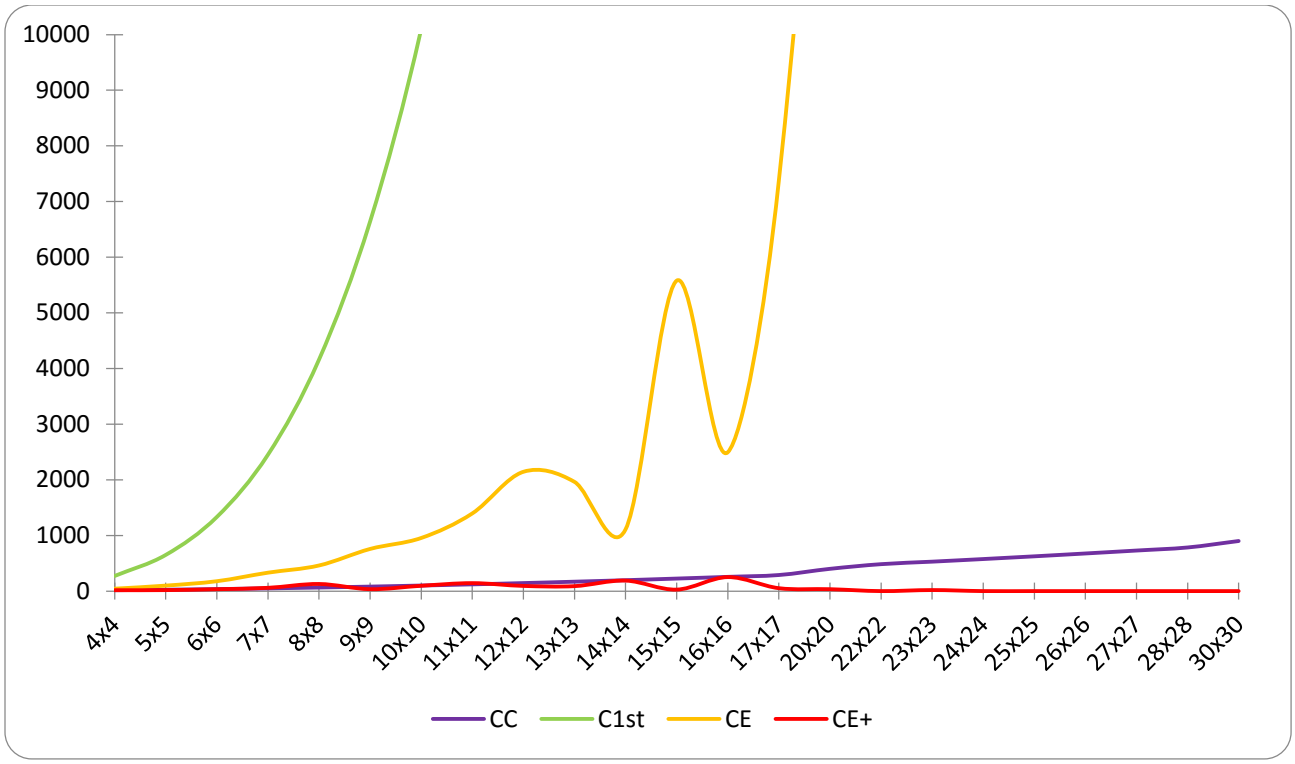
Biểu đồ 3.2: So sánh thời gian chạy chi tiết giữa các phương pháp trên bộ dữ liệu BI

Thời gian chạy của CE^+ và SWP là nhanh nhất, với những ma trận đầu vào càng lớn thì 2 thuật toán này cho tốc độ càng tiến sát với nhau. C1st chạy khá chậm, nhưng tính ổn định cao, với những ma trận đầu vào lớn nó vẫn có thể giải được dù hơi lâu, không giống CC những map lớn cho thời gian giải quá lâu. Tốc độ của CE^+ đã nhanh hơn rất nhiều so với CE, điều này chứng tỏ những nâng cấp của CE^+ là rất hiệu quả về mặt hiệu năng.

- **Số biến sử dụng**

Size	CC	C1st	CE	CE^+
4x4	16	272	44	16
5x5	25	650	99	18
6x6	36	1332	178	34
7x7	49	2450	331	59
8x8	64	4160	460	128
9x9	81	6642	758	34
10x10	100	10100	955	96
11x11	121	14762	1396	143
12x12	144	20880	2144	93
13x13	169	28730	1963	89
14x14	196	38612	1109	189
15x15	225	50850	5578	27
16x16	256	65792	2494	252
17x17	289	83810	7374	52
20x20	400	160400	19769	34
22x22	484	234740	47354	0
23x23	529	280370	51047	19
24x24	576	332352	80070	0
25x25	625	391250	80272	0
26x26	676	457652	114330	0
27x27	729	532170	113518	0
28x28	784	615440	137284	0
30x30	900	810900	188102	0

Bảng 3.2: So sánh số biến được sử dụng giữa các phương pháp trên bộ dữ liệu BI



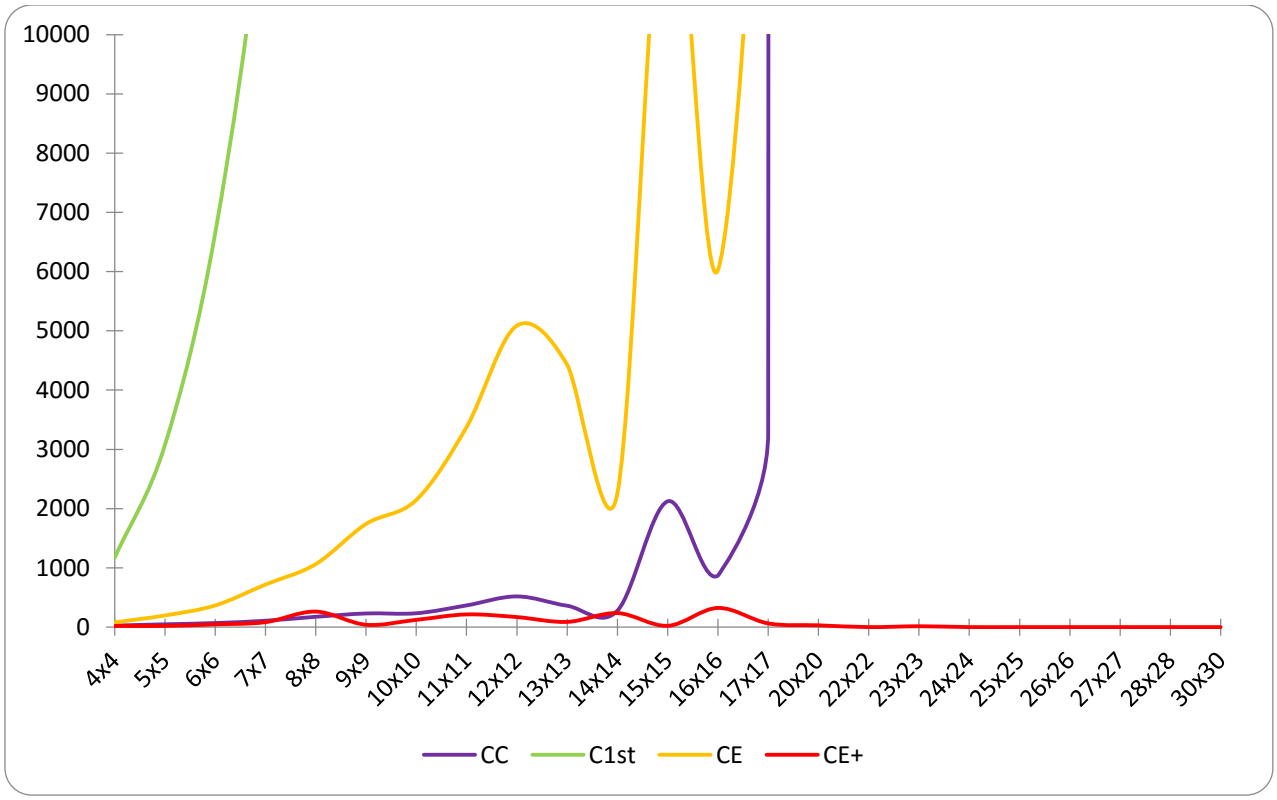
Biểu đồ 3.3: So sánh số biến được sử dụng giữa các phương pháp trên bộ dữ liệu BI

Số biến sử dụng của CE^+ là rất tốt. Do còn phụ thuộc vào dữ liệu đầu vào nên không phải cứ với những ma trận đầu vào lớn là số biến sẽ lớn, nó phụ thuộc vào những ô mà ta gọi là CsNF. Số lượng CsNF càng nhỏ thì số biến được encoding hay số biến sử dụng lại càng ít. Số biến được sử dụng bởi CC là ít nhưng nó lại chạy khá chậm, bởi vì tốc độ còn phụ thuộc nhiều vào số mệnh đề được tạo ra nữa.

- **Số mệnh đề được tạo ra**

Size	CC	C1st	CE	CE^+
4x4	26	1171	80	20
5x5	47	3074	197	20
6x6	68	6650	365	46
7x7	107	12671	718	83
8x8	174	22044	1063	262
9x9	231	35836	1741	39
10x10	233	55227	2146	123
11x11	366	81582	3376	214
12x12	518	116379	5087	169
13x13	360	161223	4431	88
14x14	277	217930	2258	237
15x15	2123	288468	13316	22
16x16	869	374755	6027	324
17x17	3283	479162	17731	62
20x20	Timeout	924789	49013	30
22x22	Timeout	1359194	135172	0
23x23	Timeout	1626268	136966	15
24x24	Timeout	1930914	217212	0
25x25	Timeout	2276390	221792	0
26x26	Timeout	2666340	320343	0
27x27	Timeout	3104255	314764	0
28x28	Timeout	3594025	385187	0
30x30	Timeout	4744986	514606	0

Bảng 3.3: So sánh số mệnh đề được tạo ra giữa các phương pháp trên bộ dữ liệu BI



Biểu đồ 3.4: So sánh số mệnh đề được tạo ra giữa các phương pháp trên bộ dữ liệu BI

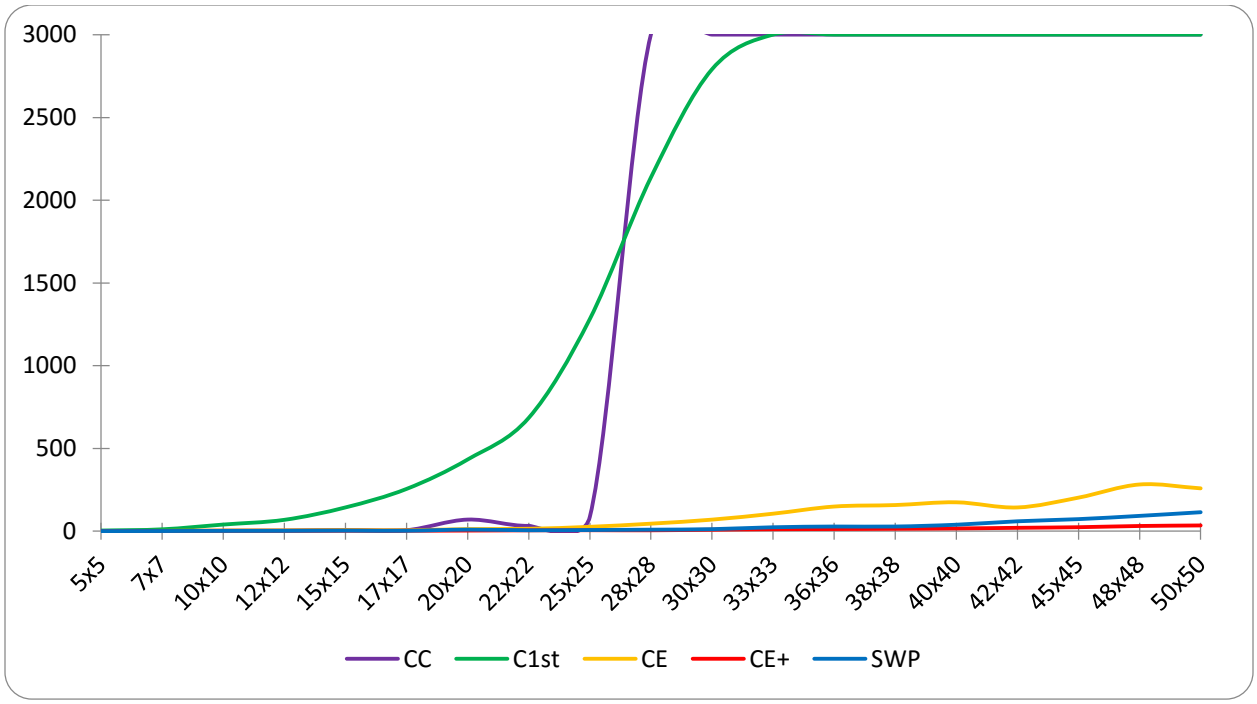
Cũng giống như số biến, số mệnh đề được tạo ra bởi CE^+ cũng là rất tốt và phụ thuộc nhiều vào CsNF, số lượng CsNF càng nhỏ thì số mệnh đề được tạo ra lại càng nhỏ. Dù số biến được sử dụng của CC là rất ít nhưng số mệnh đề được tạo ra lại lớn nên tốc độ của CC là không được tốt lắm. Thuật toán C1st thể hiện rõ được sự ổn định của nó, số lượng mệnh đề được tạo ra nhiều và tăng ổn định.

3.2.2 Bộ dữ liệu CB

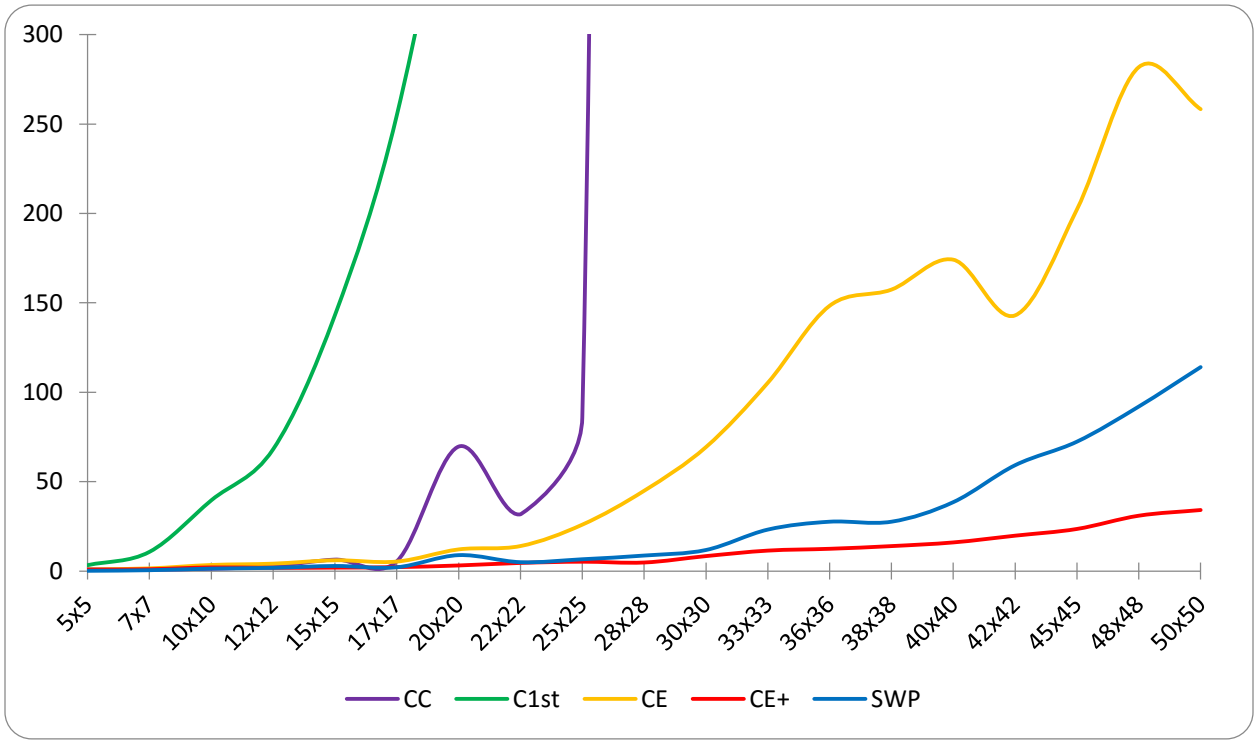
- **Thời gian chạy**

Size	CC	C1st	CE	CE$^+$	SWP
5x5	1.04	3.32	0.89	0.77	0.17
7x7	0.84	10.75	1.52	1.05	0.50
10x10	1.32	39.58	3.43	2.06	1.41
12x12	2.65	68.12	4.17	1.82	1.80
15x15	6.43	143.17	6.04	1.94	2.87
17x17	5.48	254.89	5.33	2.21	2.15
20x20	69.63	432.75	12.13	3.15	8.92
22x22	31.77	684.39	14.00	4.57	4.98
25x25	85.86	1281.92	25.96	5.26	6.62
28x28	Timeout	2137.45	44.92	4.80	8.70
30x30	Timeout	2791.25	69.34	8.39	11.8
33x33	Timeout	Timeout	105.28	11.46	23.2
36x36	Timeout	Timeout	148.36	12.47	27.62
38x38	Timeout	Timeout	157.41	13.98	27.62
40x40	Timeout	Timeout	174.12	16.03	38.56
42x42	Timeout	Timeout	143.01	19.80	59.22
45x45	Timeout	Timeout	202.26	23.56	72.38
48x48	Timeout	Timeout	281.80	30.98	92.02
50x50	Timeout	Timeout	258.31	34.15	114.07

Bảng 3.4: So sánh thời gian chạy giữa các phương pháp trên bộ dữ liệu CB



Biểu đồ 3.5: So sánh thời gian chạy tổng quát giữa các phương pháp trên bộ dữ liệu CB



Biểu đồ 3.6: So sánh thời gian chạy chi tiết giữa các phương pháp trên bộ dữ liệu CB

Với những ma trận đầu vào trong bộ CB này kết quả rất tuyệt vời với tốc tộ cho ra rất tốt của CE^+ so với SWP. Với ma trận đầu vào có kích cỡ 50x50 tốc độ của CE^+ nhanh gấp 3.34 lần tốc độ của SWP.

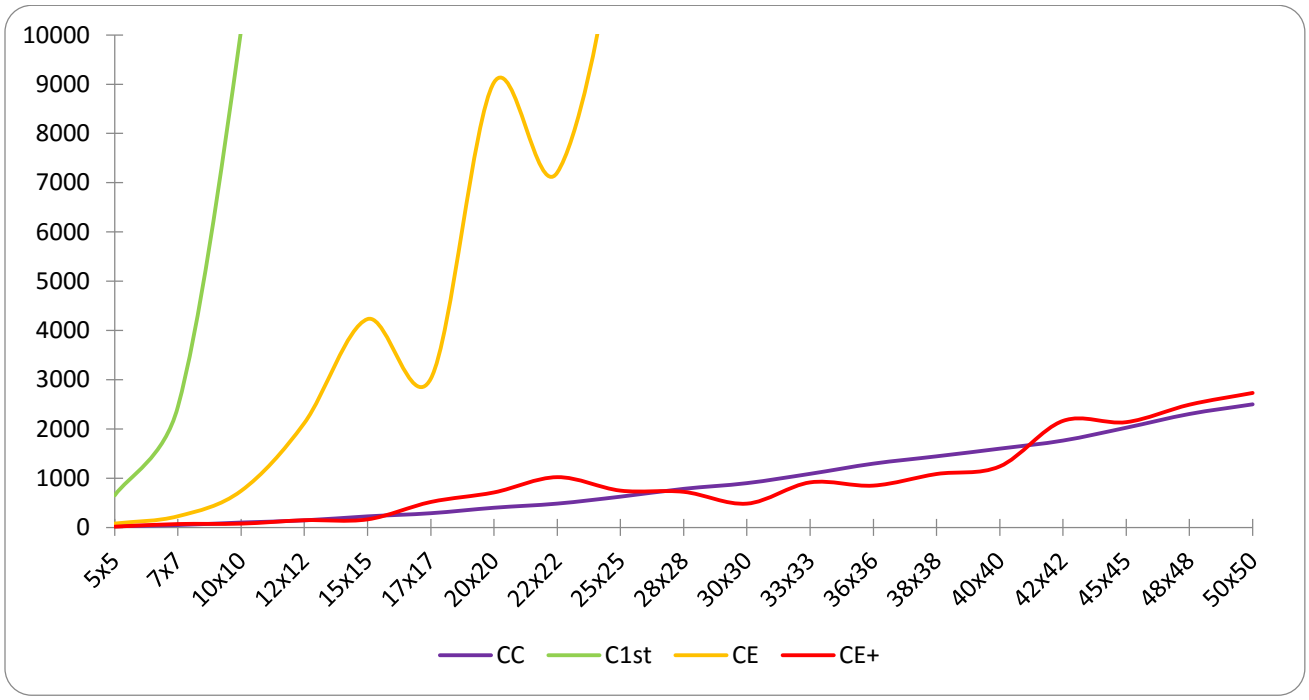
Sự khác biệt này so với bộ BI về tốc độ giữa CE^+ và SWP là do CsNF trong bộ CB này là rất lớn. Nó ảnh hưởng nhiều tới 2 phương pháp này, CE^+ với việc Encoding hết những ô CsNF còn SWP thì phải đệ quy rất nhiều những ô CsNF. Thời gian xử lý sẽ chênh lệch nhau ở đây. Trong bộ BI do các mẫu tìm kiếm đã phủ gần như toàn bộ ma trận nên CE^+ có vẻ không được hiệu quả như SWP, nhưng với bộ CB đã có sự khác biệt.

Tốc độ của CE^+ và SWP vẫn nhanh hơn rất nhiều so với CE và các phương pháp khác. Trong đó CC có thể chạy nhanh với những ma trận đầu vào nhỏ, nhưng khi ma trận đầu vào càng lớn thì tốc độ của nó lại càng chậm lại. C1st vẫn còn rất chậm.

- **Số biến được tạo ra**

Size	CC	C1st	CE	CE^+
5x5	25	650	78	16
7x7	49	2450	227	69
10x10	100	10100	742	77
12x12	144	20880	2120	147
15x15	225	50850	4231	164
17x17	289	83810	3020	517
20x20	400	160400	9036	711
22x22	484	234740	7205	1021
25x25	625	391250	12920	748
28x28	Timeout	615440	26828	724
30x30	Timeout	810900	42409	484
33x33	Timeout	Timeout	47026	915
36x36	Timeout	Timeout	71077	850
38x38	Timeout	Timeout	90939	1085
40x40	Timeout	Timeout	102959	1239
42x42	Timeout	Timeout	93805	2164
45x45	Timeout	Timeout	140621	2138
48x48	Timeout	Timeout	190338	2493
50x50	Timeout	Timeout	156729	2731

Bảng 3.5: So sánh số biến được sử dụng giữa các phương pháp trên bộ dữ liệu CB



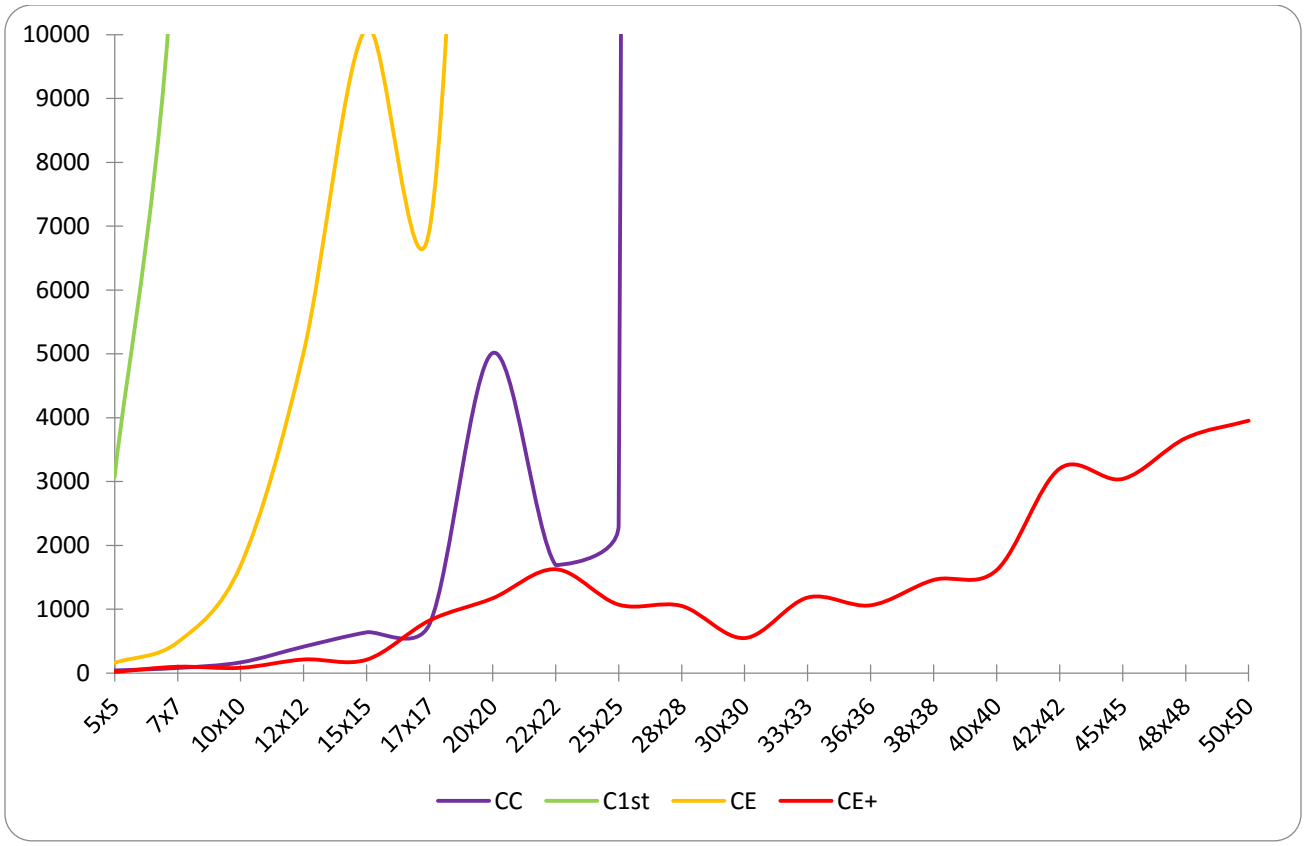
Biểu đồ 3.7: So sánh số biến được sử dụng giữa các phương pháp trên bộ dữ liệu CB

Số biến được sử dụng của CE^+ là ít hơn rất nhiều so với CE, thể hiện rõ được sự cải tiến của CE^+ so với CE. Số lượng biến sử dụng của C1st vẫn là rất nhiều. Dù sử dụng ít biến, nhưng tốc độ của CC vẫn không tốt do số lượng mệnh đề còn lớn.

- **Số mệnh đề được tạo ra**

Size	CC	C1st	CE	CE^+
5x5	42	3074	161	19
7x7	81	12666	484	99
10x10	168	55215	1686	83
12x12	415	116369	5025	213
15x15	638	288449	10104	210
17x17	771	479105	6960	825
20x20	5013	924718	21389	1172
22x22	1692	1358936	16809	1627
25x25	2323	2276062	30424	1072
28x28	Timeout	3593625	65067	1051
30x30	Timeout	4744550	107053	548
33x33	Timeout	Timeout	113703	1183
36x36	Timeout	Timeout	174592	1062
38x38	Timeout	Timeout	225960	1459
40x40	Timeout	Timeout	259161	1613
42x42	Timeout	Timeout	234448	3202
45x45	Timeout	Timeout	343050	3041
48x48	Timeout	Timeout	481325	3680
50x50	Timeout	Timeout	383111	3953

Bảng 3.6: So sánh số mệnh đề được tạo ra giữa các phương pháp trên bộ dữ liệu CB



Biểu đồ 3.8: So sánh số mệnh đề được tạo ra giữa các phương pháp trên bộ dữ liệu CB

Số mệnh đề của CE^+ tạo ra là rất ít so với CE và các phương pháp khác, hiển nhiên tốc độ của nó là rất tốt. CC dù tạo ra ít mệnh đề với những ma trận đầu vào nhỏ, nhưng khi ma trận đầu vào càng lớn, số mệnh đề được tạo ra lại rất lớn.

Chương 4. Hitori Solver

Chương này khóa luận xin được giới thiệu về chương trình được cài đặt để thực nghiệm những phương pháp đã giới thiệu ở chương 2.

4.1 Cấu hình

Một số thông số hệ thống được đặt trong file **config.properties**

```
################################################################################
MIN_ROWS = 4
MIN_COLUMNS = 4
MAX_ROWS = 55
MAX_COLUMNS = 55
SELECTED_ROWS = 4
SELECTED_COLUMNS = 4
#Support for SOLVER_SELECTED
#1.1.0 Chain and Circle              => 1
#2.1.0 Connect from 1st Cell         => 2
#2.2.0 Connect from 1st Cell Plus    => 3
#3.1.0 Connectivity Encoding         => 4
#3.1.1 Connectivity Enc use Zones    => 5
#3.2.0 Connectivity Encoding + SwP   => 6
#3.2.1 ConnectEnc use Zones + SwP    => 7
#3.2.2 ConnectEnc use ZonesSwP Plus  => 8
#4.0.1 SWP Old version Tiep          => 9
#4.0.2 SWP Old version Anh           => 10
#4.1.0 SWP - 8 Pattern - Normal cter => 11
#4.1.1 SWP - more EasyPattern - cter => 12
#4.2.0 SWP - use Create              => 13
#4.2.1 SWP - use Recursion           => 14
#5.1.0 Pattern Encoding              => 15
SOLVER_SELECTED = 8
TAB_SELECTED = 1

################################################################################
NUMBER_OF_PREY = 7
TAB_MULTI_SELECTED_NCIRCLE = 1
TAB_MULTI_SELECTED_NMAP = 20
TAB_MULTI_SELECTED_NMAPSIZE = 23

################################################################################
TAB_CREATE_MAP_SELECTED_NMAP = 5
TAB_CREATE_MAP_SELECTED_FROM_PERCENT = 20
TAB_CREATE_MAP_SELECTED_TO_PERCENT = 35
PATH_CREATE_MAP_DATA = map/createmap

################################################################################
PATH_GLUEMINISAT = /home/doduy/app/glueminisat-2.2.8/binary/glueminisat-simp
PATH_LINGELING = /home/doduy/app/lingeling-ayv-86bf266-140429/binary/lingeling
#Support for SELECTED_SATSOLVER
#SAT4J        => 1
#SAT4J_QUICKSOLVE => 2
#SAT4J_INPROCESS  => 3
```

Hình 4.1: Các thông số mặc định trong file config

```
39 ##############################################################################################
40 PATH_GLUEMINISAT = /home/doduy/app/glueminisat-2.2.8/binary/glueminisat-simp
41 PATH_LINGELING = /home/doduy/app/lingeling-ayv-86bf266-140429/binary/lingeling
42 #Support for SELECTED_SATSOLVER
43 #SAT4J        => 1
44 #SAT4J_QUICKSOLVE => 2
45 #SAT4J_INPROCESS  => 3
46 #MINISAT      => 4
47 #GLUEMINISAT      => 5
48 #LINGELING    => 6
49 SELECTED_SATSOLVER = 1
50
51 ##############################################################################################
52 #CPU_TIME     => 1
53 #USER_TIME    => 2
54 #TOTAL_TIME       => 3
55 SELECTED_TIMER = 1
56
57 ##############################################################################################
58 PATH_MAP = map/mapI
59 PATH_ICON_GAME = image/icon.png
60 HEADER_LENGTH_FILE_CNF = 50
61 PATH_FILE_CNF = file/Cnf.in
62 PATH_FILE_CNF_OUT = file/Cnf.out
63 PATH_SAT4J_JAR_FILE = lib/org.sat4j.core.jar
64 PATH_FILE_EXCEL = file/TestHitoriExcel.xlsx
65
66 ##############################################################################################
67 TIMEOUT = 2
68 TIMEOUT_SAT4J = 60000
69
```

Hình 4.2: Các thông số mặc định trong file config (tiếp)

Các thông số được đặt như sau:

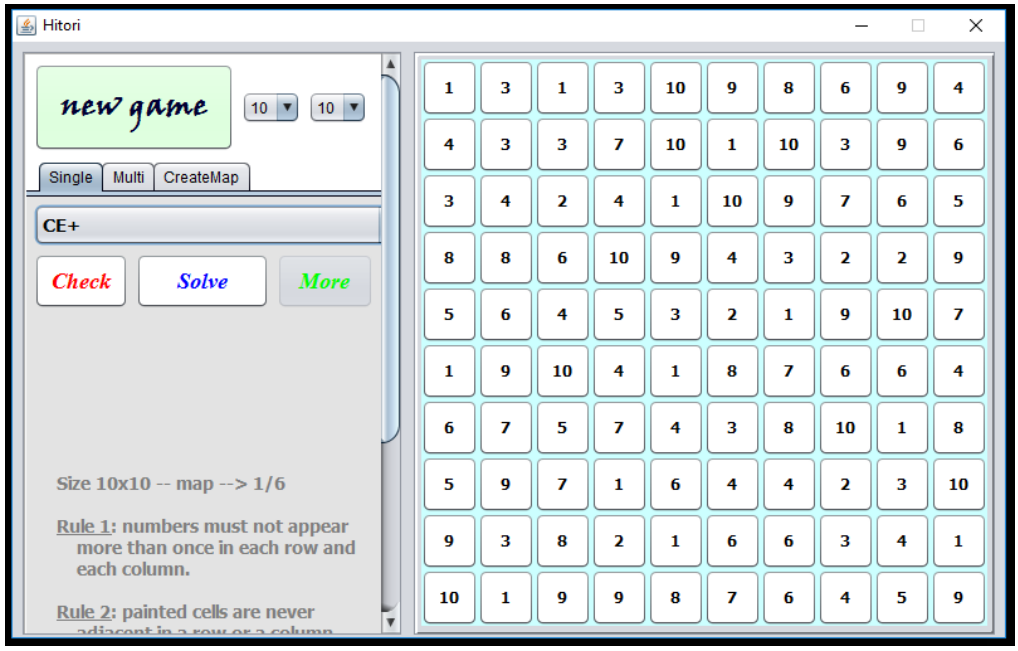
- **MIN_ROWS**: giá trị nhỏ nhất có thể được chọn cho hàng của ma trận.
- **MIN_COLUMNS**: giá trị nhỏ nhất có thể được chọn cho cột của ma trận.
- **MAX_ROWS**: giá trị lớn nhất có thể được chọn cho hàng của ma trận.
- **MAX_COLUMNS**: giá trị lớn nhất có thể được chọn cho cột của ma trận.
- **SELECTED_ROWS**: giá trị mặc định được chọn cho hàng của ma trận.
- **SELECTED_COLUMNS**: giá trị mặc định được chọn cho cột của ma trận.
- **SOLVER_SELECTED**: phương pháp giải quyết câu đố mặc định được chọn.
- **TAB_SELECTED**: giao diện mặc định được chọn.
- **NUMBER_OF_PREY**: Số vòng thực hiện trước khi bắt đầu kiểm thử kết quả.

- **TAB_MULTI_SELECTED_NCIRCLE**: số vòng thực hiện lặp lại việc kiểm thử ma trận.
- **TAB_MULTI_SELECTED_NMAP**: số map tối đa được thực hiện trong một kích cỡ ma trận đầu vào.
- **TAB_MULTI_SELECTED_NMAPSIZE**: số kích cỡ ma trận tối đa được chọn.
- TAB_CREATE_MAP_SELECTED_NMAP: số ma trận sẽ được tạo ra.
- **TAB_CREATE_MAP_SELECTED_FROM_PERCENT**: giá trị % bắt đầu của những ô có thể tô đen.
- **TAB_CREATE_MAP_SELECTED_TO_PERCENT**: giá trị % kết thúc của những ô có thể tô đen.
- **PATH_CREATE_MAP_DATA**: đường dẫn nơi những ma trận được tạo ra sẽ lưu.
- **PATH_GLUEMINISAT**: đường dẫn tới vị trí cài đặt glueminisat.
- **PATH_LINGELING**: đường dẫn tới vị trí cài đặt lingeling.
- **SELECTED_SATSOLVER**: lựa chọn SAT Solver trong danh sách các SAT Solver được hỗ trợ.
- **SELECTED_TIMER**: lựa chọn loại đơn vị đo cho thời gian.
- **PATH_MAP**: đường dẫn tới thư mục lưu các ma trận.
- **PATH_ICON_GAME**: đường dẫn tới icon của chương trình.
- **HEADER_LENGTH_FILE_CNF**: kích cỡ header của file CNF.
- **PATH_FILE_CNF**: đường dẫn tới file CNF.
- **PATH_FILE_CNF_OUT**: đường dẫn tới file kết quả của SAT Solver.
- **PATH_SAT4J_JAR_FILE**: đường dẫn nơi lưu sat4j.
- **PATH_FILE_EXCEL**: vị trí file excel được tạo ra sẽ lưu.
- **TIMEOUT**: time out của các phương pháp khi kiểm thử.
- **TIMEOUT_SAT4J**: time out của sat4j.

4.2 Giao diện Hitori Solver

4.2.1 Giao diện giải quyết bài toán lôgic Hitori

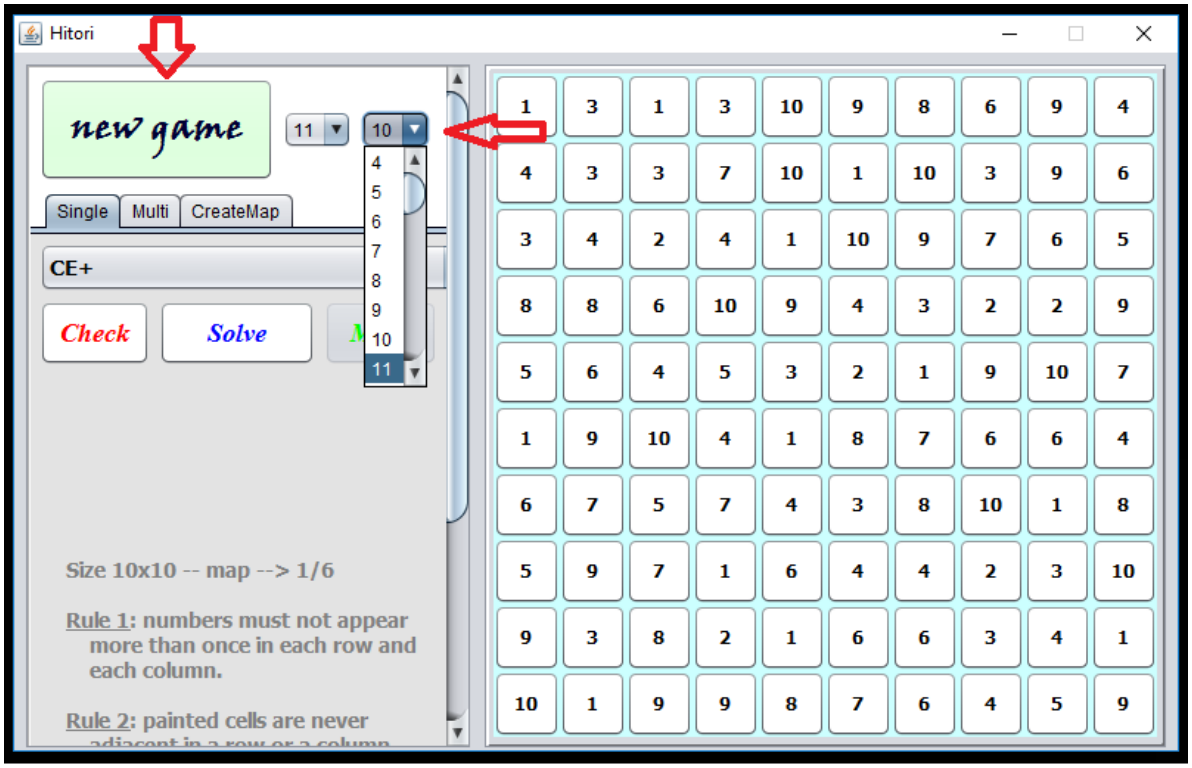
Giao diện này giúp người thực nghiệm có thể quan sát cơ bản kết quả thực hiện của từng phương pháp đối với các ma trận Hitori khác nhau.



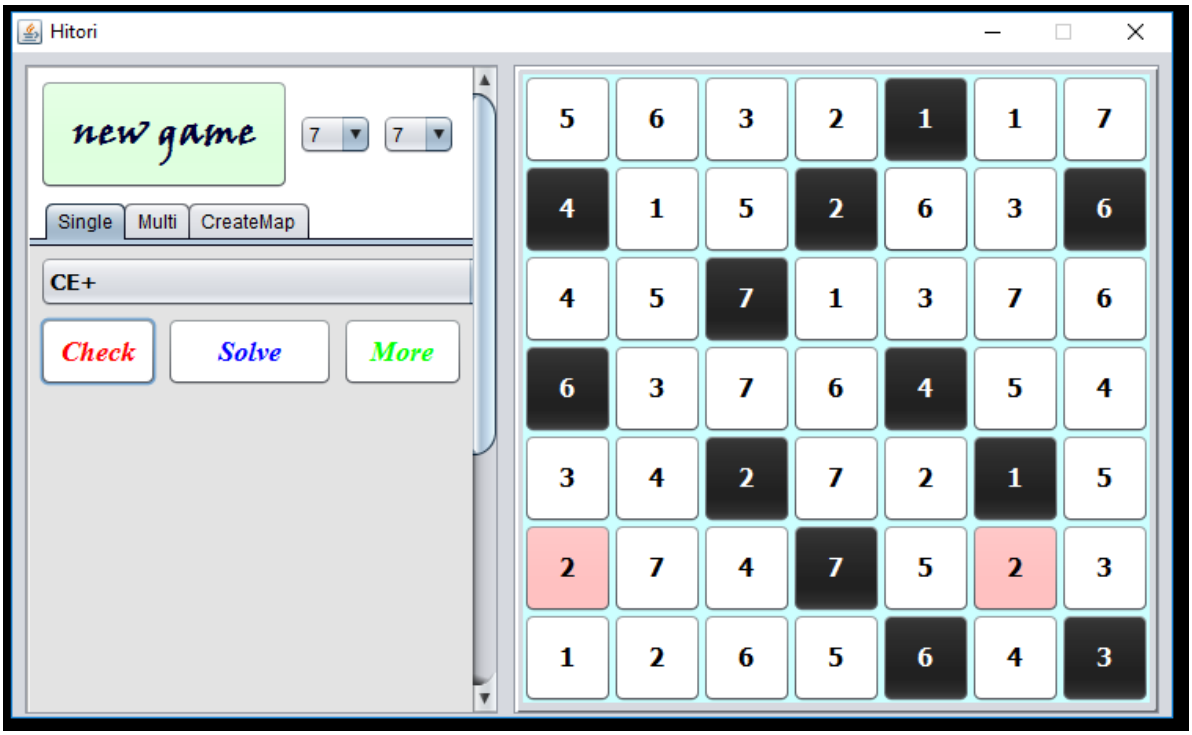
Hình 4.3: Giao diện giải quyết bài toán hitori

Các chức năng mà giao diện này hỗ trợ:

- Thay đổi ma trận đầu vào từ cơ sở dữ liệu tùy vào kích cỡ được chọn (Hình 4.4).
- Chơi thử trò chơi bằng cách nhấn vào các ô trên ma trận hiển thị, kiểm tra kết quả đã đáp ứng đủ cả 3 luật của trò chơi hay chưa.
- Người chơi có thể chơi thử bằng cách nhấn vào các ô trên ma trận hiển thị để tô đen hay bỏ tô đen ô mà người dùng muốn.
- Nút "Check" giúp người thực hiện kiểm tra lỗi nếu có (những ô gây ra lỗi của trò chơi sẽ hiển thị màu hồng) từ đó tìm ra nguyên nhân và khắc phục lỗi đó khi chơi cũng như cài đặt các phương pháp giải quyết bài toán (Hình 4.5).

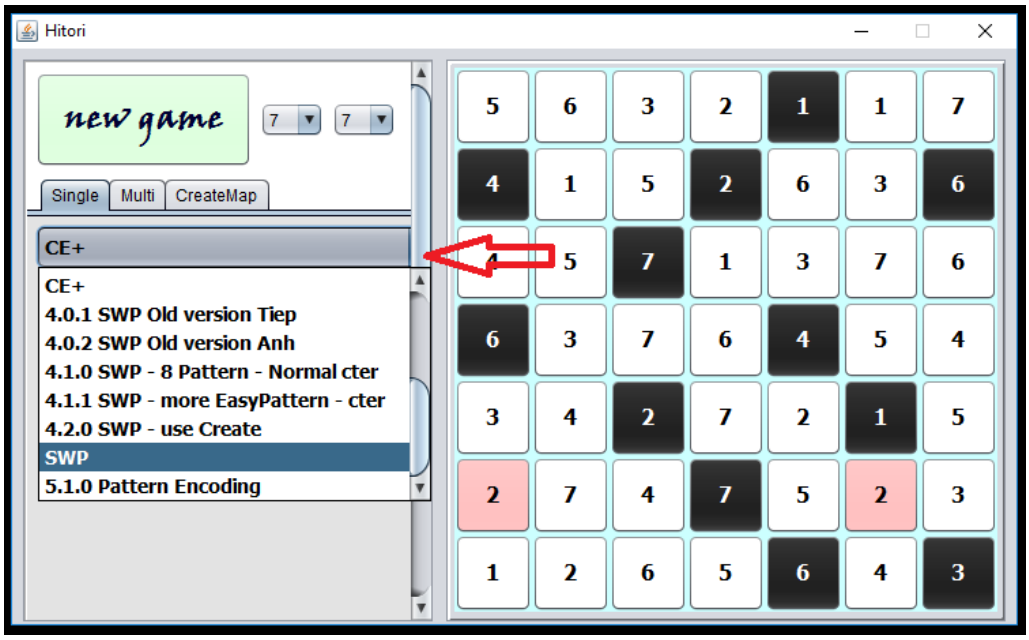


Hình 4.4: Chức năng chọn kích cỡ ma trận và hiển thị ma trận mới



Hình 4.5: Chức năng chơi và kiểm tra lỗi

- Lựa chọn phương pháp giải quyết trò chơi Loogic Hitori từ list các phương pháp đã cài đặt (Hình 4.6).



Hình 4.6: Danh sách chọn phương pháp giải quyết trò chơi logic Hitori

- Giải quyết bài toán và hiển thị kết quả, trong đó có lời giải của trò chơi và các thông số hệ thống của phương pháp được lựa chọn đã sử dụng để giải trò chơi (Hình 4.7).

Hình 4.7: Kết quả của một ma trận Hitori

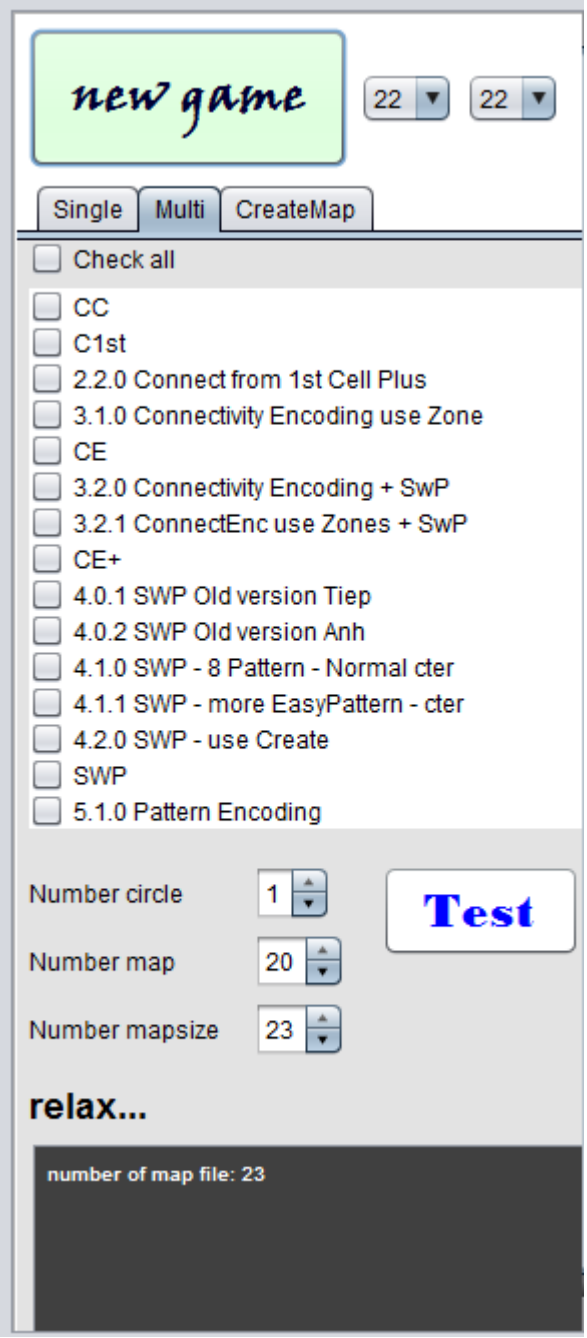
- Hiển thị thêm kết quả nếu có của ma trận vừa được giải quyết (Hình 4.8).

Hình 4.8: Một kết quả khác cho ma trận đầu vào của Hitori

4.2.2 Giao diện tự động chạy thử bài toán lôgic Hitori

Giao diện này với nhiệm vụ chính là tự động giải quyết nhiều ma trận đầu vào với nhiều kích cỡ khác nhau và sử dụng nhiều phương pháp khác nhau. Kết quả sẽ được tổng hợp lại và ghi lại trong file excel, từ đó chương trình sẽ tự động sinh một biểu đồ excel để người tiến hành thực nghiệm dễ dàng so sánh.

Các thành phần của giao diện:



Hình 4.9: Các thành phần của giao diện tự động kiểm thử

- Lựa chọn các phương pháp sẽ được thực hiện
- Lựa chọn số vòng chạy với một ma trận
- Lựa chọn số ma trận với từng kích cỡ

- Lựa chọn số kích cỡ ma trận
- Thanh trạng thái
- Bảng trạng thái

4.2.3 Giao diện sinh tự động ma trận đầu vào cho bài toán lôgic Hitori

Giao diện giúp người tiến hành thực nghiệm dễ dàng quan sát cũng như tiến hành sinh ma trận đầu vào cho trò chơi Hitori.

Hình 4.10: Giao diện sinh ma trận đầu vào

Các thành phần của giao diện:

- Số map sẽ được tạo ra cùng lúc
- Phần trăm số ô có thể được tô đen được tạo ra
- Nút tạo ma trận
- Nút lưu ma trận
- Nút tạo và lưu nhiều ma trận cùng lúc

Mỗi khi ma trận được tạo ra nó sẽ được hiển trị trên ma trận hiển thị, từ ma trận hiển thị này người tiến hành thực nghiệm có thể đo được những thông số chính xác của ma trận vừa tạo ra thông qua giao diện giải quyết bài toán trong phần 1.

Kết luận

Khóa luận trình bày phương phát SAT Encoding CE^+ giải trò chơi lôgic Hitori, CE^+ được cải tiến dựa trên CE [2] với việc áp dụng các mẫu tìm kiếm để xác định những ô chắc chắn bị bôi đen trong Hitori.

Thực nghiệm đánh giá CE^+ so với 4 phương pháp giải Hitori hiện nay (bao gồm CC, C1st, CE, SWP) trên 2 bộ dữ liệu BI (dữ liệu thu thập từ các trang web online về giải Hitori, gồm 190 ma trận đầu vào, kích thước tối đa là 30x30) và bộ dữ liệu CB (dữ liệu do nhóm tạo ra với 110 ma trận đầu vào, kích thước tối đa là 50x50).

CE^+ cho kết quả cải tiến đáng kể so với CE trên tất cả các đánh giá: về thời gian chạy, số lượng biến sử dụng, số mệnh đề sinh ra. Trên bộ dữ liệu BI, CE^+ cho kết quả cạnh tranh so với SWP (phương pháp tìm kiếm dựa trên các mẫu tìm kiếm, SWP được cho là phương pháp cho kết quả giải Hitori nhanh nhất hiện nay [5]). Trên bộ dữ liệu CB (với kích thước ma trận tối đa lên tới 50x50), CE^+ cho kết quả tốt hơn SWP (biểu đồ 3.6). Trên bộ dữ liệu CB, SWP thực hiện phương pháp đệ quy - quay lui cho các ô chưa xác định của ma trận Hitori, trong khi đó CE^+ tận dụng và khai thác những ưu việt của SAT solver (minisat) trong việc tìm kiếm và xác định những ô bôi đen trong ma trận Hitori.

Các kết quả thực nghiệm trong khóa luận cho thấy được sự cải tiến đáng kể của CE^+ so với CE và đã chứng minh được tính hiệu quả của CE^+ so với các phương pháp giải trò chơi lôgic Hitori hiện nay. Kết quả của khóa luận cũng đã khẳng định được khả năng ứng dụng của phương pháp SAT Encoding vào các bài toán về trò chơi lôgic (Logic puzzle), trí tuệ nhân tạo…

Thông qua quá trình nghiên cứu và thực hiện khóa luận này, bản thân tôi đã được tiếp xúc với một cách tiếp cận hoàn toàn mới để giải quyết một số các trò chơi logic mà trong đó điển hình là Hitori. Việc hoàn thành khoá luận này đã giúp tôi có những kiến thức vững chắc về bài toán SAT, SAT Solver và SAT Encoding. Bên cạnh những kiến thức tiếp thu được, tôi còn học hỏi được phong cách làm việc nghiêm túc từ thầy hướng dẫn.

Toàn bộ tài liệu là công sức nghiên cứu của tôi trong thời gian qua. Việc có sai sót là điều khó tránh khỏi, rất mong nhận được những ý kiến đánh giá, đóng góp của các thầy cô và những bạn có chung niềm đam mê với SAT Solver nói chung và trò chơi logic lý thú nói riêng để tôi có thể hoàn thiện hơn nữa trong tương lai.

Tài Liệu Tham Khảo

Tiếng Việt

[1] Đỗ Văn Duy. SAT Encoding kết hợp mẫu tìm kiếm giải trò chơi logic Hitori. *Giải nhì - Hội nghị sinh viên với Nghiên cứu khoa học cấp khoa - Khoa CNTT - ĐH Công Nghệ , 2016*.

[2] Trần Trọng Tiệp, Vũ Đình Thắng, Đào Thị Thúy, Nguyễn Tuấn Anh and Nguyễn Thị Mai Hương, Phương pháp SAT Encoding hiệu quả giải trò chơi Hitori. *Giải ba - Hội nghị sinh viên với Nghiên cứu khoa học cấp trường - ĐH Công Nghệ, 2015*.

Tiếng Anh

[3] DPLL algorithm, http://www.dis.uniroma1.it/~liberato/ar/dpll/dpll.html.

[4] Een, N. , Sorensson, N. : MiniSAT, http://minisat.se.

[5] Gander, M. : Hitori solver. Bachelor Thesis: http://homepage.uibk.ac.at/~csae1761/hitori/website/res/MGCH. pdf, 2006.

[6] GlueMiniSat Solver, http://www.jaist.ac.jp/~hirokawa/aj12/nabeshima.pdf.

[7] Hitori database, http:// www.menneske.no/hitori/eng/.

[8] Loogic puzzle, http://www.puzzlebooks.net/.

[9] Lynce, I., Ouaknine, J.: Sudoku as a sat problem. In: In Proc. of the Ninth International Symposium on Artificial Intelligence and Mathematics, Springer (2006).

[10] Michael Genesereth (Stanford University): Introduction to Lôgic , http://Lôgic.stanford.edu/introLôgic/chapters/chapter_05.html.

[11] Hitori puzzle, http://www.nikoli.com/en/puzzles/hitori/.

[12] Numberlink puzzle, http://www.nikoli.com/en/puzzles/numberlink/rule.html.

[13] P. Cabalar, http://www.dc.fi.udc.es/~cabalar/kr/2014/hitori.c, 2014.

[14] Pfeiffer, U., Karnagel, T., Scheffler, G.: A sudoku-solver for large puzzles using sat. In Voronkov, A., Sutcliffe, G., Baaz, M., Fermüller, C., eds.: LPAR-17-short. Volume 13 of EPiC Series., EasyChair (2013) 52–57.

[15] S.-J. Yen, T.-C. Su and S.-Y. Chiu.: http://www1.rdoffice.ndhu.edu.tw/exchange/abroad/abroad98/34_paper.pdf.

[16] SAT Solver: UBCSAT, http://ubcsat.dtompkins.com.

[17] Sat4j Solver, http://www.sat4j.org/.

[18] SATLIVE, www.satlive.Org.

[19] Slitherlink puzzle, http://www.gmpuzzles.com/blog/slitherlink-rules-and-info/.

[20] Sudoku with SAT, http://www.cs.cmu.edu/~hjain/papers/sudoku-as-SAT.pdf.