ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Phạm Văn Phúc

NGHIÊN CỨU BÀI TOÁN PESP ÁP DỤNG ĐỂ LẬP LỊCH GIỜ TÀU ĐIỆN CHẠY

Ngành: Công nghệ thông tin

HÀ NỘI - 2024

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Phạm Văn Phúc

NGHIÊN CỨU BÀI TOÁN PESP ÁP DỤNG ĐỂ LẬP LỊCH GIỜ TÀU ĐIỆN CHẠY

Ngành: Công nghệ thông tin

Cán bộ hướng dẫn: Tô Văn Khánh

HÀ NỘI - 2024

Tóm tắt

Hiện nay, hiệu năng của các SAT Solver đã cải thiện đáng kể và có thể được ứng dụng trong việc giải các bài toán NP-complete như: *Traveling Salesman, Hamiltonian path , graph k-coloring*.... Vấn đề lập lịch sự kiện định kỳ (Periodic Event Scheduling Problem) từ lâu đã được chương minh là một vấn đề NP-complete. Các phương pháp hiện tại như lập trình ràng buộc (Constraint satisfaction Programing) hay quy hoạch số nguyên (Integer Programing) chưa thực sự hiệu quả với các bộ dữ liệu lớn. Tài liệu này sẽ trình bày thuật toán chuyển hóa vấn đề lập lịch định kỳ (PESP) về bài toán SAT, sau đó giải bài toán sử dụng SAT Solver nhằm đạt được hiệu suất cao hơn.

Từ khóa: Periodic railway timetabling; Optimisation; Periodic Event Scheduling Problem; SAT

Lời cảm ơn

Quá trình thực hiện đề tài khóa luận tốt nghiệp là một hành trình đầy ý nghĩa và thử thách. Em xin gửi lời cảm ơn chân thành nhất đến tất cả những người đã đồng hành cùng em trong suốt thời gian qua. Đặc biệt, em xin bày tỏ lòng biết ơn sâu sắc đến thầy Tô Văn Khánh. Chính sự tận tâm hướng dẫn, những góp ý quý báu và kinh nghiệm phong phú của thầy đã giúp em định hình rõ hơn vấn đề nghiên cứu và vượt qua nhiều khó khăn. Em luôn trân trọng những buổi trao đổi cùng thầy, khi thầy không chỉ truyền đạt kiến thức chuyên môn mà còn giúp em rèn luyện tư duy khoa học.

Em cũng xin gửi lời cảm ơn đến toàn thể quý thầy cô, cán bộ giảng viên trường Đại học Công Nghệ. Nhờ sự tận tình giảng dạy của thầy cô, em đã có được hành trang kiến thức vững chắc để bước vào nghiên cứu. Môi trường học tập thân thiện và cơ sở vật chất hiện đại của nhà trường đã tạo điều kiện thuận lợi cho em hoàn thành tốt khóa luận.

Cuối cùng, em xin kính chúc các thầy cô luôn mạnh khỏe, hạnh phúc và gặt hái nhiều thành công hơn nữa trong sự nghiệp trồng người cao quý.

Sinh viên

Phạm Văn Phúc

Lời cam đoan

Em xin cam đoan khóa luận tốt nghiệp là của em, do em thực hiện dưới sự hướng dẫn của TS. Tô Văn Khánh. Tất cả tham khảo, nghiên cứu và tài liệu liên quan đều được nêu rõ ràng và chi tiết trong danh mục tài liệu tham khảo. Các nội dung trình bày trong khóa luận này là hoàn toàn trung thực và không sao chép bất kỳ nguồn nào khác mà không trích dẫn.

Hà Nội, ngày 03 tháng 11 năm 2024

Sinh viên

Phạm Văn Phúc

Mục lục

Tóm tắt i

Lời cảm ơn ii

Lời cam đoan iii

Mở đầu ix

Chương 1. Lập lịch sự kiện định kỳ **1**

1.1 Mạng sự kiện định kỳ 1

1.2 Bài toán lập lịch sự kiện định kỳ 7

Chương 2. Kiến thức nền tảng **8**

2.1 Logic mệnh đề 8

2.1.1 Mệnh đề 8

2.1.2 Các phép toán logic 8

2.1.3 Biểu thức logic 11

2.1.4 Chuẩn tắc tuyển và chuẩn tắc hội 12

2.2 SAT 14

2.2.1 Vấn đề SAT 14

2.2.2 SAT Solver 16

Chương 3. Mô hình bài toán PESP về bài toán SAT **19**

3.1 Mã hóa đại lượng rời rạc 19

3.1.1 Mã hóa trực tiếp (Direct Encoding) 20

3.1.2 Mã hóa thứ tự (Order Encoding) 22

3.2 PESP as Direct Encoding 24

3.3 PESP as Order Encoding 27

3.3.1 Mã hóa thứ tự ràng buộc thời gian 28

3.3.2 Mã hóa thứ tự ràng buộc đối xứng 33

3.4 So sánh Direct encoding và Order encoding 35

3.4.1 Số biến 35

3.4.2 Số mệnh đề 36

Chương 4. Thực nghiệm và kết quả **38**
4.1 Mô hình bài toán PTSP về bài toán PESP 38
4.2 Thu thập dữ liệu 38
4.3 Kết quả và đánh giá 40
Tài liệu tham khảo **45**

Danh mục viết tắt

CNF:	Conjunctive Normal Form
CSP:	Contraint Satisfaction Problem
MIP:	Mixed Integer Programing
PESP:	Periodic Event Scheduling Problem
PTSP:	Periodic Train Timetable Scheduling Problem
SAT:	Satisfiablility
UNSAT:	Unsatisfiability

Danh mục hình ảnh

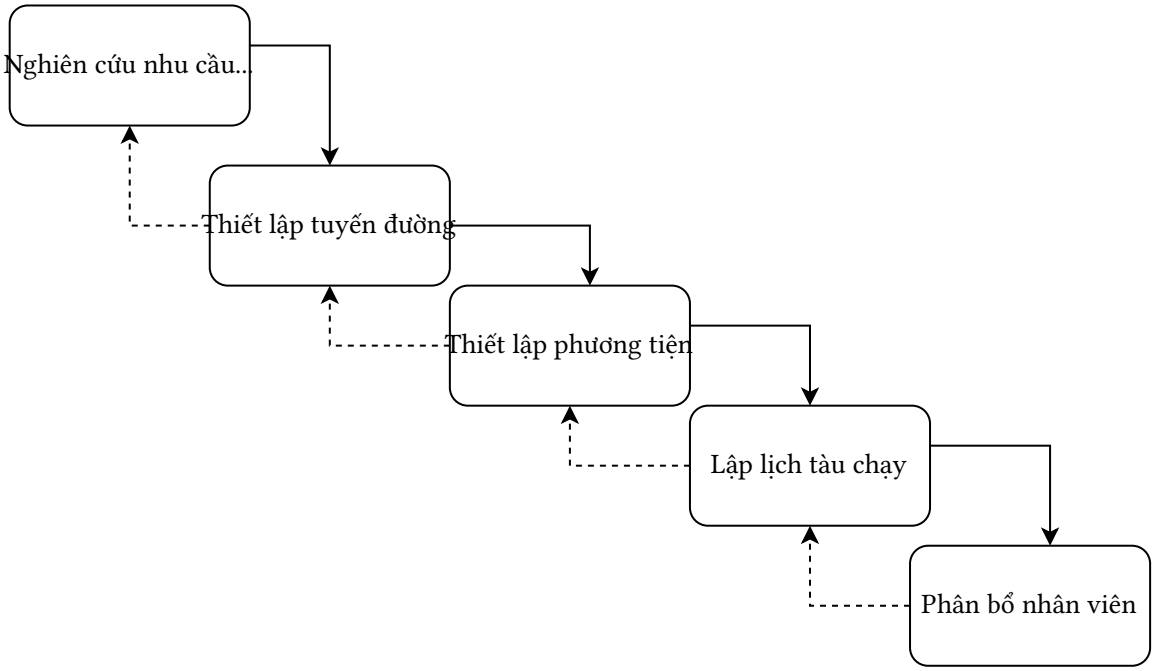
Hình 1: Các giai đoạn lập kế hoạch xây dựng hệ thống tàu điện ngầm 9
Hình 2: Minh họa đoạn mô-đun. -10 và 5 thuộc I 2
Hình 3: Ví dụ minh họa mạng sự kiện định kỳ 2
Hình 4: Ví dụ minh họa mạng lịch trình hợp lệ 4
Hình 5: Ví dụ minh họa mạng lịch trình hợp lệ 5
Hình 6: Sơ đồ đầu vào/đầu ra của SAT Solver 16
Hình 7: Sơ đồ giải bài toán thực tế sử dụng SAT Solver 18
Hình 8: Sơ đồ tổng quan giải bài toán PESP với mã hóa trực tiếp 24
Hình 9: Sơ đồ tổng quan giải bài toán PESP với mã hóa thứ tự 27
Hình 10: Minh họa loại bỏ miền không thỏa mãn ràng buộc thời gian $((A, B), [2, 4]_8)$ 28
Hình 11: Bản chất của miền vô nghiệm$((A, B), [2, 4]_8)$ 29
Hình 12: Minh họa vùng xác định kích thước vùng không thỏa mãn 30
Hình 13: Minh họa tập hợp hình chữ nhật phủ vùng vô nghiệm $((A, B), [2, 4]_8)$ 31
Hình 14: Biểu đồ đường so sánh số biến của Direct và Order Encoding 42
Hình 15: Biểu đồ đường so sánh số mệnh đề của Direct và Order Encoding 42
Hình 16: Biểu đồ đường so sánh thời gian thực thi của Direct và Order Encoding 43

Danh mục bảng biểu

Bảng 1: Bảng chân trị của phép phủ định 9
Bảng 2: Bảng chân trị của phép hội 9
Bảng 3: Bảng chân trị của phép tuyển 10
Bảng 4: Bảng chân trị của phép kéo theo 10
Bảng 5: Bảng chân trị của phép tương đương 10
Bảng 6: Bảng chân trị của hai biểu thức tương đương 12
Bảng 7: Bảng chân trị của biểu thức UNSAT 16
Bảng 8: Bảng mô tả độ phức tạp của dữ liệu PESP đầu vào 40
Bảng 9: Cấu hình máy chạy thực nghiệm 41
Bảng 10: Kết quả chạy thử nghiệm, thời gian tính bằng mili giây (ms) 42

Mở đầu

Lập kế hoạch cho hệ thống tàu điện ngầm là một công việc đầy khó khăn và thử thách, bao gồm nhiều giai đoạn khác nhau, như: nghiên cứu thị trường, thiết lập tuyến đường, thiết lập phương tiện, lập lịch tàu chạy và đào tạo nhân viên. Các giai đoạn lập kế hoạch này liên quan mật thiết đến nhau và thường được tiến hành đổ thác theo thứ tự. Tuy nhiên, có thể quay lại bước trước đó để tối ưu khi các yêu cầu nghiêm vụ được làm rõ hơn.



Hình 1: Các giai đoạn lập kế hoạch xây dựng hệ thống tàu điện ngầm

1. **Nghiên cứu nhu cầu di chuyển**: Khảo sát thị trường và nhu cầu di chuyển của khách hàng nhằm thiết kế tuyến đường phù hợp
2. **Thiết lập tuyến đường**: Dựa trên nhu cầu di chuyển, ta thiết kế các tuyến đường nhằm đạt hiệu quả di chuyển cao nhất, quy trình này cần đảm bảo các chuyến tàu được kết nối với nhau và giảm thiểu số lần chuyển chuyến.
3. **Thiết lập phương tiện**: Dựa theo nhu cầu di chuyển và tuyến đường, ta cần lập danh sách các phương tiện (cần bao nhiêu phương tiện, sức chứa, tốc độ di chuyển...)

4. **Xây dựng lịch trình**: Khi biết rõ tuyến đường và công thông số phương tiện, ta có thể xây dựng lịch trình tàu chạy. Lịch trình cần đáp ứng các yêu chuẩn an toàn cũng như các yêu cầu về nghiệp vụ, và sẵn sàng cho các tình huống sự cố gián đoạn, hủy chuyến...
5. **Phân bổ nhân viên**: Tương tự, việc xây dựng lịch trình cho các nhân viên lái tàu, phục vụ, nhân viên sửa chữa, bảo hành cũng cần được quan tâm.

Trong đó, *xây dựng lịch trình* là giai đoạn thiết yếu đối với hệ thống tàu điện ngầm. Việc cung cấp thời gian khởi hành và đến chính xác giúp lịch trình tàu hoạt động mượt mà và đáng tin cậy, đồng thời quản lý lưu lượng hành khách và ngăn ngừa tình trạng quá tải. Lịch trình hiệu quả cũng phối hợp các kết nối với các phương thức vận chuyển khác, cải thiện kế hoạch vận hành bằng cách lập kế hoạch bảo trì và phân bổ nhân sự, và tối ưu hóa việc sử dụng tài nguyên như tàu và đội ngũ. Tổng thể, lịch trình đáng tin cậy dẫn đến sự hài lòng cao hơn của khách hàng và một hệ thống giao thông công cộng trơn tru, hiệu quả hơn.

Tuy nhiên, lập lịch trình tàu hỏa là một nhiệm vụ vô cùng khó khăn và tốn kém, vì phải đáp ứng nhiều tiêu chí phức tạp. Trước hết, thời gian đệm (recovery times) cần được tính toán để bù đắp cho những gián đoạn trong hệ thống, như sự thay đổi tốc độ do thời tiết hoặc thiên tai, và tình trạng tàu khởi hành muộn so với dự kiến. Để tránh làm gián đoạn toàn bộ hệ thống, lịch trình phải bao gồm thời gian đệm phù hợp. Tiếp theo, thời gian giãn cách tối thiểu (minimum headway time) là cần thiết để đảm bảo an toàn khi hai tàu sử dụng chung một đường ray và phải khởi hành cách nhau một khoảng thời gian tối thiểu. Tính kết nối (connections between trains) cũng rất quan trọng, vì thời gian đến và khởi hành của các tàu tại cùng một bến đỗ cần phải liên tục để phục vụ nhu cầu nối chuyến của hành khách. Cuối cùng, thời gian bảo trì (turn around times at termination) phải được tính toán để bao gồm thời gian cần thiết cho việc bảo trì động cơ, tiếp nhiên liệu và thay ca nhân viên ở ga tàu cuối trước khi tàu quay trở lại.

Trước đây, xây dựng lịch trình tàu chạy chủ yếu được làm thủ công [1], tốn nhiều chi phí cũng như dễ xảy ra sai sót. Vì vậy, trong thập kỷ trước, nhiều nghiên cứu nhằm hỗ trợ và tự động quá trình lập lịch đã được tiến hành [2], [3], [4]. Hầu hết các nghiên cứu đều dựa trên mô hình *lập lịch sự kiện định kỳ* (Periodic Event Schedule Problem - PESP), một mô hình nổi tiếng được giới thiệu bởi Serafini and Ukovich [5]. Tuy nhiên, các phương pháp hiện tại trong việc giải bài toán PESP như lập trình ràng buộc, quy hoạch số nguyên chưa hiệu quả với các bộ dữ liệu lớn. Trong khóa luận này, chúng tôi trình bày phương pháp giải bài toán PESP sử dụng SAT Solver và đề xuất một vài cải tiến về thuật toán mã hóa.

Phần còn lại của khóa luận được tổ chức như sau:

- **Chương 1**: Định nghĩa chi tiết các khái niệm trong bài toán *lập lịch sự kiện định kỳ*, một số cách tiếp cận hiện tại
- **Chương 2**: Trình bày kiến nền tảng về logic mệnh đề và các khái niệm liên quan đến bài toán SAT như NP-complete, Satisfiablility Problem, SAT Solver và quy trình giải bài toán thực tế sử dụng SAT solver.
- **Chương 3**: Đề xuất thuật toán chuyển đổi bài toán PESP về bài toán SAT và cách giải cùng cải tiến thuật toán mã hóa.
- **Chương 4**: Thử nghiệm thực tế và kết luận

CHƯƠNG 1

Lập lịch sự kiện định kỳ

1.1 Mạng sự kiện định kỳ

Chương

Định nghĩa 1.1.1: (Đoạn). Cho $a, b \in \mathbb{Z}$ với $a \leq b$.

$$[a, b] := \{a, a+1, a+2, ..., b-1, b\}$$

được gọi là đoạn từ cận dưới a đến cận trên b hay đoạn từ a đến b.

Định nghĩa 1.1.2: (Đoạn mô-đun). Cho $a, b \in \mathbb{Z}$ and $t \in \mathbb{N}^*$. Với a là cận dưới và b là cận trên,

$$[a, b]_t := \bigcup_{z \in \mathbb{Z}} [a + z \cdot t, b + z \cdot t]$$

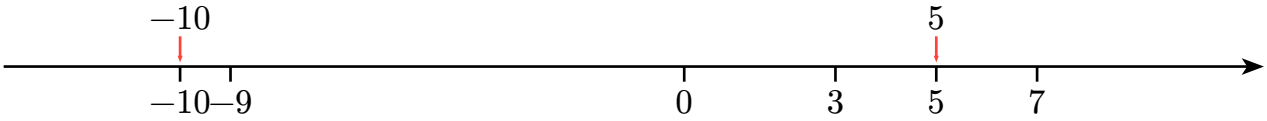
được gọi là *đoạn mô-đun t*.

Ví dụ 1.1.1: Cho

$$\begin{aligned} I &= [3, 7]_8 \\ &= ... \cup [-13, -9] \cup [-5, -1] \cup [3, 7] \cup [11, 15] \cup [19, 23] ... \subset Z \end{aligned}$$

là đoạn mô-đun 8. Khi đó

$$5 \in [3, 7] \subset I$$
$$-10 \in [-13, -9] \subset I$$
$$12 \in [11, 15] \subset I$$



Hình 2: Minh họa đoạn mô-đun. -10 và 5 thuộc I

Định nghĩa 1.1.3: (Mạng sự kiện định kỳ). Một mạng sự kiện định kỳ (Periodic event network) chu kỳ t_T $\mathrm{N} = (\nu, A, t_T)$ bao gồm tập hợp ν sự kiện (danh sách đỉnh) và tập các ràng buộc A (danh sách cạnh). Mỗi ràng buộc $a \in A$ kết nối hai sự kiện, được kí hiệu:

$$a = \Big((i,j), [l_a, u_a]_{t_T}\Big) \in (\nu \times \nu) \times 2^{\mathbb{Z}}$$

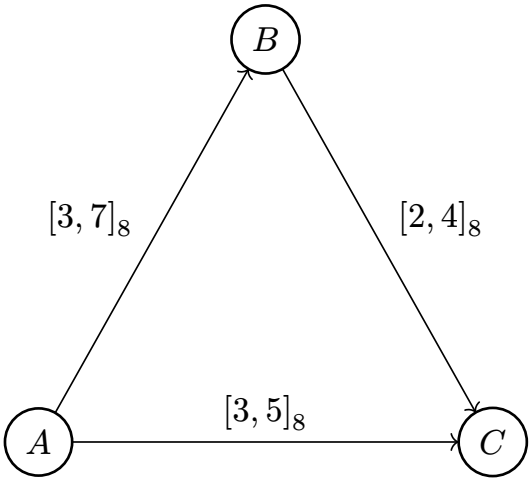
trong đó $l_a, u_a \in \mathbb{Z}$ lần lượt là cận trên và cận dưới, $2^{\mathbb{Z}}$ là tập hợp các tập con của $\mathbb{Z}$.

Tập A là hợp của hai tập hợp S và C lần lượt là tập ràng buộc đối xứng và ràng buộc thời gian.

$$S \cup C = A$$
$$S \cap C = \emptyset$$

Ví dụ 1.1.2: Cho $N = (\{A, B, C\}, \mathrm{N}, 8)$ là một mạng sự kiện định kỳ. Trong đó:

$$\begin{aligned} A = C = \{&((A,B), [3,7]_8),\\ &((B,C), [2,4]_8),\\ &((A,C), [3,5]_8)\} \end{aligned}$$



Hình 3: Ví dụ minh họa mạng sự kiện định kỳ

Định nghĩa 1.1.4: (Tiềm năng sự kiện). Cho $N = (\nu, A, t_T)$ là một *mạng sự kiện định kỳ*. $\pi_n \in \mathbb{Z}$ được gọi là tiềm năng xảy ra của sự kiện $n \in \nu$.

Từ đây đến hết tài liệu, khái niệm này được gọi tắt là *tiềm năng*

Định nghĩa 1.1.5: (Lịch trình). Cho $N = (\nu, A, t_T)$ là một *mạng sự kiện định kỳ*. Ánh xạ:

$$\begin{aligned} \Pi_\nu : \nu &\to \mathbb{Z} \\ n &\mapsto \pi_n \end{aligned}$$

được gọi là một lịch trình của tập sự kiện N

Định nghĩa 1.1.6: (Ràng buộc thời gian). Cho $N = (\nu, A, t_T)$ là một *mạng sự kiện định kỳ* với tập ràng buộc $A = S \cup C$ với ràng buộc $a = \left((i,j), [l_a, u_a]_{t_T}\right) \in C, i, j \in \nu$. Hai tiềm năng π_i và π_j thỏa mãn ràng buộc thời gian a khi và chỉ khi:

$$\pi_j - \pi_i \in [l_a, u_a]_{t_T}$$

Định nghĩa 1.1.7: (Ràng buộc đối xứng). Cho $N = (\nu, A, t_T)$ là một *mạng sự kiện định kỳ* với tập ràng buộc $A = S \cup C$ với ràng buộc $a = \left((i,j), [l_a, u_a]_{t_T}\right) \in S, i, j \in \nu$. Hai tiềm năng π_i và π_j thỏa mãn ràng buộc đối xứng a khi và chỉ khi:

$$\pi_j + \pi_i \in [l_a, u_a]_{t_T}$$

Ví dụ 1.1.3: Cho A, B là hai sự kiện và $a = ((A, B), [3, 7]_8)$ là *ràng buộc thời gian*.

Các tiềm năng $(\pi_a, \pi_b) = (1, 5), (3, 2)$ thỏa mãn ràng buộc thời gian a vì:

$$\begin{aligned} 5 - 1 = 4 &\in [3, 7]_8 \\ 2 - 3 = -1 &\in [3, 7]_8 \end{aligned}$$

Ngược lại, tiềm năng $(\pi_a, \pi_b) = (3, 5), (7, 1)$ không thỏa mãn ràng buộc, vì:

$$5 - 3 = 2 \notin [3, 7]_8$$
$$1 - 7 = -6 \notin [3, 7]_8$$

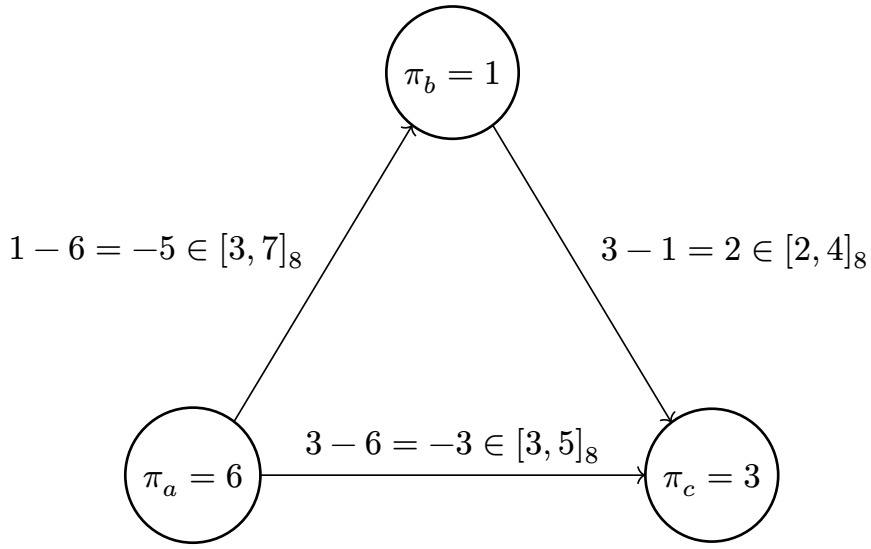
Định nghĩa 1.1.8: Cho $N = (\nu, A, t_T)$ là một *mạng sự kiện định kỳ* và Π_ν là một lịch trình của N. Lịch trình Π thỏa mãn một ràng buộc $a \in A$ khi và chỉ khi $\pi_i = \Pi_{\nu(i)}, \pi_j = \Pi_{\nu(j)}$ thỏa mãn $a = (i, j, I)$

Định nghĩa 1.1.9: (Lịch trình hợp lệ) Cho $N = (\nu, A, t_T)$ là một *mạng sự kiện định kỳ* và Π_ν là một lịch trình của N. Lịch trình Π được xem là hợp lệ nếu thỏa mãn mọi ràng buộc $a \in A$.

Ví dụ 1.1.4: Cho $N = (\nu, A, 8)$ là mạng sự kiện định kỳ ở Ví dụ 1.1.2 và Π_ν là một lịch trình hợp lệ của N với $\pi_a = 6, \pi_b = 1, \pi_c = 3$. Π_ν là hợp lệ bởi vì:

$$\pi_b - \pi_a = 1 - 6 = -5 \in [3, 7]_8$$
$$\pi_c - \pi_b = 3 - 1 = 2 \in [2, 4]_8$$
$$\pi_c - \pi_a = 3 - 6 = -3 \in [3, 5]_8$$

được minh họa trong hình dưới đây:



Hình 4: Ví dụ minh họa mạng lịch trình hợp lệ

Định nghĩa 1.1.10: (Lịch trình tương đương) Cho $N = (\nu, A, t_T)$ là một *mạng sự kiện định kỳ* và Π_ν là một lịch trình của N. Lịch trình Π_ν và Φ_ν được xem là tương đương:

$$\Pi_\nu \equiv \Phi_\nu$$

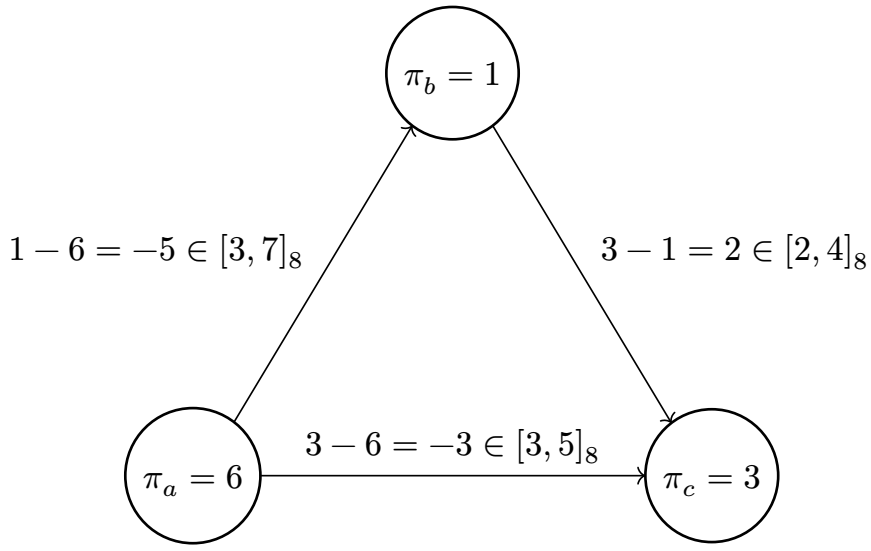
khi và chỉ khi

$$\forall n \in \nu : \Pi_{\nu(n)} \bmod t_T = \Phi_{\nu(n)} \bmod t_T$$

Ví dụ 1.1.5: Cho $N = (\nu, A, 8)$ là mạng sự kiện định kỳ ở Ví dụ 1.1.2 và Π_ν là một lịch trình hợp lệ của N với $\pi_a = 6, \pi_b = 1, \pi_c = 3$. Π_ν là hợp lệ bởi vì:

$$\pi_b - \pi_a = 1 - 6 = -5 \in [3, 7]_8$$
$$\pi_c - \pi_b = 3 - 1 = 2 \in [2, 4]_8$$
$$\pi_c - \pi_a = 3 - 6 = -3 \in [3, 5]_8$$

được minh họa trong hình dưới đây:



Hình 5: Ví dụ minh họa mạng lịch trình hợp lệ

Định lý 1.1.1: (Tính hợp lệ của lịch trình tương đương). Cho $N = (\nu, A, t_T)$ là một mạng sự kiện định kỳ, Π_ν, Φ_ν là hai lịch trình tương đương. Khi đó, với tập ràng buộc A:

$$\Pi_\nu \text{ hợp lệ} \Leftrightarrow \Phi_\nu \text{ hợp lệ}$$

Proof: Không mất tính tổng quát, chỉ cần chứng minh Π_ν hợp lệ $\Rightarrow \Phi_\nu$ hợp lệ.

Ta có: Π_ν hợp lệ với tập ràng buộc $A \Rightarrow \forall a \in A$:

$$\Pi_\nu \text{ thỏa mãn } a \tag{1}$$

Thật vậy, với $A = C \cup S$, cần chứng minh:

$$\forall a \in C : \Phi_\nu \text{ thỏa mãn } a \tag{2}$$

$$\forall a \in S : \Phi_\nu \text{ thỏa mãn } a \tag{3}$$

Giả sử $a = \Big((i,j), [l_a, u_a]_{t_T}\Big) \in C$ là một ràng buộc thời gian bất kỳ.

với $i, j \in \nu, \pi_i = \Pi_\nu(i), \pi_j = \Pi_\nu(j), \varphi_i = \Phi_\nu(i), \varphi_j = \Phi_\nu(j)$

$$\begin{aligned}
(1) &\Rightarrow \pi_j - \pi_i \in [l_a, u_a]_{t_T} \\
&\Rightarrow \pi_j - \pi_i \in \{[l_a + z \cdot t_T, u_a + z \cdot t_T] \mid z \in \mathbb{Z}\} \\
&\Rightarrow \forall w, v \in \mathbb{Z} : \pi_j - \pi_i + w \cdot t_T - v \cdot t_T \in \{[l_a + z \cdot t_T, u_a + z \cdot t_T] \mid z \in \mathbb{Z}\} \\
&\Rightarrow \forall w, v \in \mathbb{Z} : \big(\pi_j + w \cdot t_T\big) - (\pi_i + v \cdot t_T) \in \{[l_a + z \cdot t_T, u_a + z \cdot t_T] \mid z \in \mathbb{Z}\} \\
&\Rightarrow \big(\pi_j \bmod t_T\big) - (\pi_i \bmod t_T) \in \{[l_a + z \cdot t_T, u_a + z \cdot t_T] \mid z \in \mathbb{Z}\} \qquad (4) \\
&\Rightarrow \big(\pi_j \bmod t_T\big) - (\pi_i \bmod t_T) \in [l_a, u_a]_{t_T} \\
&\Rightarrow \big(\varphi_j \bmod t_T\big) - (\varphi_i \bmod t_T) \in [l_a, u_a]_{t_T} \\
&\Rightarrow \varphi_j - \varphi_i \in [l_a, u_a]_{t_T} \\
&\Rightarrow \Phi_\nu \text{ thỏa mãn } a
\end{aligned}$$

Tương tự, ra chứng minh được Phương trình 3 □

Hệ quả 1.1.1.1: Cho $[\Pi_\nu]_{\equiv} := \{\Phi_\nu \mid \Phi_\nu \equiv \Pi_\nu\}$, Π_ν hợp lệ. Khi đó:

$$\forall \Phi_\nu \in [\Pi_\nu]_\equiv \Rightarrow \Phi_\nu \text{ hợp lệ}$$

Hệ quả 1.1.1.2: Cho $[\Pi_\nu]_\equiv := \{\Phi_\nu \mid \Phi_\nu \equiv \Pi_\nu\}$, Π_ν hợp lệ. Khi đó tồn tại một lịch trình $\Phi_\nu \in [\Pi_\nu]_\equiv$ sao cho:

$$\forall n \in \nu : \Phi_\nu(n) \in [0, t_T - 1]$$

Hệ quả 1.1.1.2 là hệ quả quan trọng, giới hạn miền nghiệm của lịch trình trở thành hữu hạn. Vì vậy, khi tìm kiếm lịch trình hợp lệ, ta chỉ cần tìm các tiềm năng trong đoạn $[0, t_T - 1]$. Nếu không tồn tại hệ quả này, ta không thể giải bài toán PESP với logic mệnh đề vì không gian tìm kiếm là vô hạn.

1.2 Bài toán lập lịch sự kiện định kỳ

Định nghĩa 1.2.1: (PESP). Cho $N = (\nu, A, t_T)$ là một *mạng sự kiện định kỳ*, bài toán đặt ra câu hỏi: *Liệu có tồn tại một lịch trình hợp lệ thỏa mãn mạng trên?*

Dễ thấy, PESP là một vấn đề quyết định[6]. Từ minh họa Ví dụ 1.1.2, dễ hình dung PESP có thể chuyển thành bài toán *Vertex Coloring*, vậy PESP là bài toán NP-complete, được chứng minh bằng cách chuyển về bài toán *Vertex Coloring* [7].

CHƯƠNG 2

Kiến thức nền tảng

2.1 Logic mệnh đề

2.1.1 Mệnh đề

Định nghĩa 2.1.1.1: (Mệnh đề): Mệnh đề là một nhận định đúng hoặc sai. Kí hiệu: $x, y, z, A, B, C...$

Ví dụ 2.1.1.1: Các nhận định sau là mệnh đề:

- A = "Hôm nay trời mưa"
- B = "2 là số nguyên tố"
- z = "10 lớn hơn 20"

trong khi các nhận định sau không phải mệnh đề do không có tính đúng sai rõ ràng:

- Lan bao nhiêu tuổi?
- Dọn nhà đi!

Định nghĩa 2.1.1.2: (Chân trị) Một mệnh đề có thể đúng hoặc sai. Tính đúng sai của mệnh đề được gọi là chân trị của mệnh đề. Mệnh đề đúng có chân trị là 1 (`true`), mệnh đề sai có chân trị 0 (`false`).

Ví dụ 2.1.1.2:

Theo Ví dụ 2.1.1.1, A, B là mệnh đề đúng, z là mệnh đề sai:

2.1.2 Các phép toán logic

Tương tự với số học, ta cũng có phép toán giữa các mệnh đề. Sau đây giới thiệu một số phép toán cơ bản thường dùng.

Định nghĩa 2.1.2.1: (Phủ định): Mệnh đề phủ định của mệnh đề a là mệnh đề có chân trị đối lập với a.
Kí hiệu: $\neg a$.

a	$\neg a$
0	1
1	0

Bảng 1: Bảng chân trị của phép phủ định

Ví dụ 2.1.2.1:

$$A = \text{"Hôm nay trời mưa"}$$
$$\Rightarrow \neg A = \text{"Hôm nay trời không mưa"}$$

Định nghĩa 2.1.2.2: (Phép hội): Mệnh đề hội của hai mệnh đề a, b là mệnh đề chỉ đúng khi cả a, b đều đúng.
Kí hiệu: $a \wedge b$

a	b	$a \wedge b$
0	0	0
1	0	0
0	1	0
1	1	1

Bảng 2: Bảng chân trị của phép hội

Định nghĩa 2.1.2.3: (Phép tuyển): Mệnh đề tuyển của hai mệnh đề a, b là mệnh đề chỉ sai khi cả a, b đều sai.
Kí hiệu: $a \vee b$

a	b	$a \vee b$
0	0	0
1	0	1
0	1	1
1	1	1

Bảng 3: Bảng chân trị của phép tuyển

Định nghĩa 2.1.2.4: (Phép kéo theo): Mệnh đề kéo của hai mệnh đề a, b là mệnh đề chỉ sai khi cả a đúng b sai.
Kí hiệu: $a \Rightarrow b$

a	b	$a \Rightarrow b$
0	0	1
1	0	0
0	1	1
1	1	1

Bảng 4: Bảng chân trị của phép kéo theo

Định nghĩa 2.1.2.5: (Phép tương đương): Mệnh đề tương của hai mệnh đề a, b là mệnh đề chỉ đúng khi cả a và b cùng đúng hoặc cùng sai.
Kí hiệu: $a \Leftrightarrow b$

a	b	$a \Leftrightarrow b$
0	0	1
1	0	0
0	1	0
1	1	1

Bảng 5: Bảng chân trị của phép tương đương

2.1.3 Biểu thức logic

Định nghĩa 2.1.3.1: (Mệnh đề sơ cấp): Mệnh đề sơ cấp (literal) là chỉ bao gồm mệnh đề và phủ định của nó (a và $\neg a$). Mệnh đề sơ cấp không thể chia thành các mệnh đề nhỏ hơn.

Định nghĩa 2.1.3.2: (Biểu thức logic): Biểu thức logic được định nghĩa đệ quy như sau:

1. Mỗi mệnh đề sơ cấp là một biểu thức ($A, \neg B, C, x, y, z...$)
2. Nếu A, B là hai biểu thức thì $A \wedge B, A \vee B, A \Rightarrow B, A \Leftrightarrow B$ cũng là các biểu thức.

Ví dụ 2.1.3.1: Các biểu thức logic:

$$f(x, y, z) = (x \vee y) \wedge z$$
$$g(a, b) = a \Rightarrow \neg b$$

Định nghĩa 2.1.3.3: (Bảng chân trị): Bảng chân trị là bảng tính toán chân trị của biểu thức logic theo từng giá trị của các biến tham gia trong biểu thức.

Bảng 1, Bảng 2, Bảng 3 là ví dụ các bảng chân trị.

Định nghĩa 2.1.3.4: (Biểu thức tương đương): Nếu hai biểu thức f, g có cùng bảng chân trị thì f, g được gọi là biểu thức tương đương. *Kí hiệu:*

$$f \Leftrightarrow g$$

Ví dụ 2.1.3.2: Theo định nghĩa, ta có: $a \Rightarrow b \Leftrightarrow \neg a \vee b$

a	b	$a \Rightarrow b$	$\neg a \vee b$
0	0	1	1
1	0	0	0
0	1	0	0
1	1	1	1

Bảng 6: Bảng chân trị của hai biểu thức tương đương

2.1.4 Chuẩn tắc tuyển và chuẩn tắc hội

Định nghĩa 2.1.4.1: (Tuyển sơ cấp): Tuyển của các mệnh đề sơ cấp được gọi là tuyển sơ cấp.

Định nghĩa 2.1.4.2: (Hội sơ cấp): Hội của các mệnh đề sơ cấp được gọi là hội sơ cấp.

Ví dụ 2.1.4.1:

- Các tuyển sơ cấp: $a \vee \neg b \vee c \vee \neg d, x \vee y, y \vee \neg z$
- Các hội sơ cấp: $a \wedge \neg b \wedge c \wedge \neg d, x \wedge y, y \wedge \neg z$

Định nghĩa 2.1.4.3: (Chuẩn tắc tuyển): Tuyển của các hội sơ cấp được gọi là chuẩn tắc tuyển.

Ví dụ 2.1.4.2:

$$f = (x_1 \wedge x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge x_2 \wedge \neg x_3)$$

Định nghĩa 2.1.4.4: (Chuẩn tắc hội): Hội của các tuyển sơ cấp được gọi là chuẩn tắc hội (CNF). Chuẩn tắc hội có thường có dạng:

$$f = (P_1 \vee P_2 \vee P_3 \vee ... \vee P_n)_1 \wedge ... \wedge (Q_1 \vee Q_2 \vee ... \vee Q_n)_p$$

$$n, m, p \in \mathbb{N}^*$$

Ví dụ 2.1.4.3:

$$g = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3)$$

Định lý 2.1.4.1: Mọi biểu thức logic đều có dạng chuẩn tắc hội và chuẩn tắc tuyển tương ứng [8].

Ví dụ 2.1.4.4: Cho biểu thức logic:

$$f = ((p \Rightarrow q) \wedge (r \vee \neg s)) \Rightarrow (t \Leftrightarrow (u \vee v))$$

Ta có thể chuyển nó về dạng CNF như sau:

$$\begin{aligned}
f &= ((p \Rightarrow q) \wedge (r \vee \neg s)) \Rightarrow (t \Leftrightarrow (u \vee v)) \\
&= ((\neg p \vee q) \wedge (r \vee \neg s)) \Rightarrow (t \wedge (u \vee v) \vee (\neg t \wedge \neg(u \vee v))) \\
&= \neg((\neg p \vee q) \wedge (r \vee \neg s)) \vee (t \wedge (u \vee v) \vee (\neg t \wedge \neg(u \vee v))) \\
&= (p \wedge \neg q) \vee (\neg r \wedge s) \vee (t \wedge (u \vee v) \vee (\neg t \wedge (\neg u \wedge \neg v))) \\
&= ((p \wedge \neg q) \vee (\neg r \wedge s) \vee t) \wedge ((p \wedge \neg q) \vee (\neg r \wedge s) \vee u \vee v) \\
&\wedge ((p \wedge \neg q) \vee (\neg r \wedge s) \vee \neg t) \\
&\wedge ((p \wedge \neg q) \vee (\neg r \wedge s) \vee \neg t \vee \neg u) \\
&\wedge ((p \wedge \neg q) \vee (\neg r \wedge s) \vee \neg t \vee \neg v) \\
&= (p \vee \neg r \vee t \vee u \vee v) \\
&\wedge (p \vee s \vee t \vee u \vee v) \wedge (\neg q \vee \neg r \vee t \vee u \vee v) \\
&\wedge (\neg q \vee s \vee t \vee u \vee v) \wedge (p \vee \neg r \vee u \vee v) \\
&\wedge (p \vee s \vee u \vee v) \wedge (\neg q \vee \neg r \vee u \vee v) \\
&\wedge (\neg q \vee s \vee u \vee v) \wedge (p \vee \neg r \vee \neg t) \\
&\wedge (p \vee s \vee \neg t) \wedge (\neg q \vee \neg r \vee \neg t) \\
&\wedge (\neg q \vee s \vee \neg t) \wedge (p \vee \neg r \vee \neg t \vee \neg u) \\
&\wedge (p \vee s \vee \neg t \vee \neg u) \wedge (\neg q \vee \neg r \vee \neg t \vee \neg u) \\
&\wedge (\neg q \vee s \vee \neg t \vee \neg u) \wedge (p \vee \neg r \vee \neg t \vee \neg v) \\
&\wedge (p \vee s \vee \neg t \vee \neg v) \wedge (\neg q \vee \neg r \vee \neg t \vee \neg v) \\
&\wedge (\neg q \vee s \vee \neg t \vee \neg v)
\end{aligned}$$

Tương tự với dạng DNF, ta có biểu thức cuối cùng như sau:

$$
\begin{aligned}
f = &(\neg p \wedge r \wedge t \wedge u) \vee (\neg p \wedge r \wedge t \wedge v) \\
&\vee (\neg p \wedge \neg s \wedge t \wedge u) \vee (\neg p \wedge \neg s \wedge t \wedge v) \\
&\vee (q \wedge r \wedge t \wedge u) \vee (q \wedge r \wedge t \wedge v) \\
&\vee (q \wedge \neg s \wedge t \wedge u) \vee (q \wedge \neg s \wedge t \wedge v) \\
&\vee (p \wedge \neg q \wedge \neg r \wedge \neg t) \vee (p \wedge \neg q \wedge s \wedge \neg t) \\
&\vee (p \wedge \neg q \wedge \neg r \wedge \neg u \wedge \neg v) \vee (p \wedge \neg q \wedge s \wedge \neg u \wedge \neg v)
\end{aligned}
$$

2.2 SAT

Nhằm cung cấp nền tảng kiến thức, sau đây khóa luận sẽ trình bày chi tiết các khái niệm liên quan đến logic mệnh đề nói chung và bài toán SAT. Đây là cơ sở quan trọng cho Chương 3

2.2.1 Vấn đề SAT

Định nghĩa 2.2.1.1: (Suy diễn - Interpretation) Cho $f \in \Sigma_{\text{SAT}}$ là một biểu thức logic mệnh đề, khi đó ánh xạ:

$$I : \Sigma_{\text{SAT}} \to \{\text{true}, \text{false}\}$$
$$f^I = w$$

được gọi là một suy diễn I với giá trị w của f

Ví dụ 2.2.1.1: Cho $f(x, y, z) = x \wedge (y \vee z)$. Với suy diễn $I = \{x : \text{true}, y : \text{true}, z : \text{false}\}$ ta có:

$$f(\text{true}, \text{true}, \text{false}) = \text{true} \wedge (\text{true} \vee \text{false}) = \text{true}$$

hay

$$f^I(x, y, z) = \text{true}$$

Định nghĩa 2.2.1.2: Cho $f \in \Sigma_{\text{SAT}}$ là một biểu thức logic mệnh đề, khi đó ánh xạ:

$$I : \Sigma_{\text{SAT}} \to \{\text{true}, \text{false}\}$$
$$f^I = w$$

được gọi là một suy diễn I với giá trị w của f

Định nghĩa 2.2.1.3: (Khả thỏa - Satisfiability) Cho $f \in \Sigma_{\text{SAT}}$ là một biểu thức logic mệnh đề, khi đó f được gọi là khả thỏa hay SAT nếu tồn tại một suy diễn I:

$$f^I = \text{true}$$

và f không thể thỏa mãn, còn gọi là UNSAT nếu:

$$f^I = \text{false}\ \forall I$$

Định nghĩa 2.2.1.4: (SAT). Cho $f \in \Sigma_{\text{SAT}}$ là một biểu thức logic mệnh đề ở dạng chuẩn tắc hội (CNF). *Liệu có tồn tại một suy diễn I sao cho:*

$$f^I = \text{true}$$

được gọi là bài toán Satisfiability hay bài toán SAT.

Ví dụ 2.2.1.2: Ví dụ về biểu thức SAT

Cho $f = (x \vee y) \wedge \neg z$. Ta thấy tồn tại một suy diễn $I = \{x : \text{true}, y : \text{false}, z : \text{false}\}$ mà $f^I = \text{true}$.

Ví dụ 2.2.1.3: Ví dụ về biểu thức UNSAT

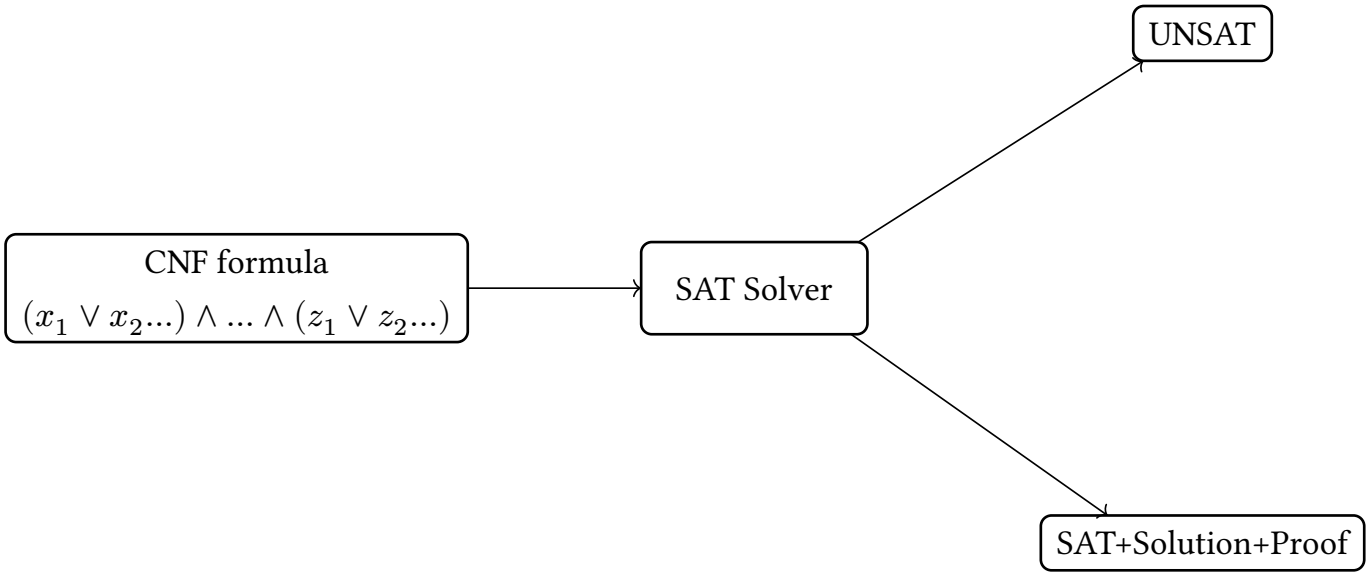
Cho $g = (x \vee y) \wedge (\neg x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee \neg y)$, không tồn tại suy diễn nào để $g^I = \text{true}$.

x	y	g
0	0	0
1	0	0
0	1	0
1	1	0

Bảng 7: Bảng chân trị của biểu thức UNSAT

2.2.2 SAT Solver

Bài toán SAT là bài toán thuộc lớp NP xuất hiện sớm nhất, đồng thời là bài toán đầu tiên được chứng minh là NP-complete [9]. Vì vậy, không tồn tại giải thuật tối ưu giải bài toán SAT có độ phức tạp đa thức. Tuy nhiên, nhiều nghiên cứu đã được tiến hành nhằm xây dựng chương trình giải bài toán SAT, thường gọi là các SAT Solver. Đầu vào chương trình thường là biểu thức logic dạng chuẩn tắc hội (CNF). Nếu biểu thức thỏa mãn được, đưa ra kết luận SAT và một nghiệm bất kỳ kèm chứng minh. Nếu không tồn tại nghiệm thoả mãn, kết luận UNSAT.



Hình 6: Sơ đồ đầu vào/đầu ra của SAT Solver

Nhiều kĩ thuật đã được nghiên cứu nhằm cải thiện độ hiệu quả các SAT Solver theo thời gian, tiêu biểu như:

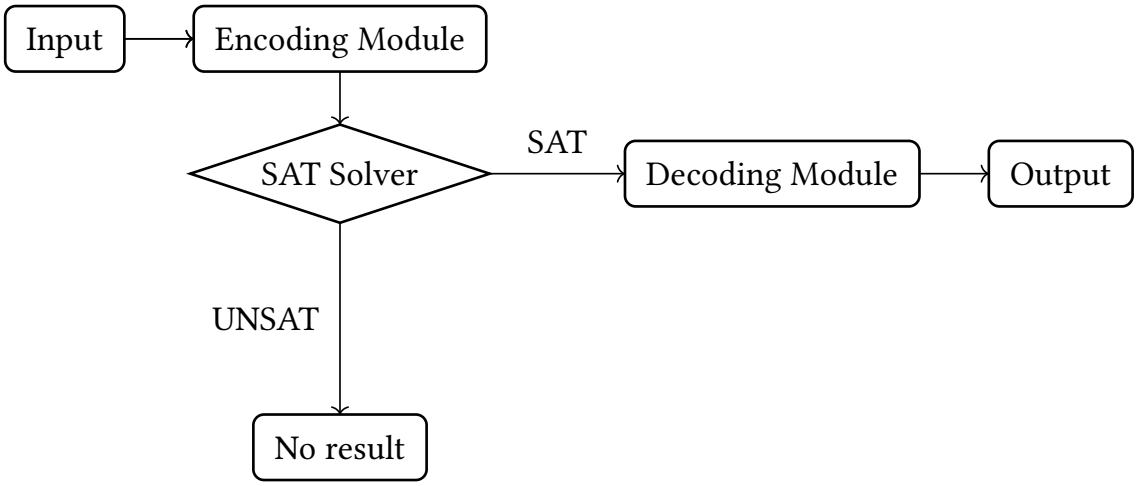
1. Thuật toán David-Putnam(1960)[10]: Giảm số biến bằng thuật toán luận giải (resolusion).
2. Thuật toán Davis-Putnam-Logemann-Loveland (DPLL)[11]: Cải thiện thuật toán David-Putnam sử dụng kĩ thuật quay lui và nổi bọt đơn vị(unit propagation).

3. Conflict-Driven Clause Learning[12]: Mở rộng thuật toán DPLL, tối ưu giải thuật quay lui. Khi gặp một cặp mệnh đề mâu thuẫn và cần quay lui, thuật toán sẽ phân tích sai lầm này để tránh lặp lại tương tự. Thêm vào đó, thuật toán có thể quay lại trực tiếp điểm gây mâu thuẫn, giảm phần lớn nhánh cần tìm kiếm.
4. Những cải tiến khác về cơ sở dữ liệu, tiền xử lý, tận dụng khả năng xử lý song song [13], [14], [15].

Do vậy, các SAT Solver hiện nay đã có khả năng giải các bài toán cực kì phức tạp, với hàng triệu biến và mệnh đề. Hằng năm, các cuộc thi về SAT Solver được tổ chức nhằm cải thiện hiệu suất thuật toán, tiêu biểu như SAT competition. Phần lớn những người tham gia công bố SAT Solver dưới dạng thử viện mã nguồn mở, có thể dễ dàng tích hợp và sử dung. Sau đây liệt kê một số Solver có ảnh hưởng quan trọng trong lịch sử phát triển của các SAT Solver:

- **CaDiCal**: CaDiCal là bộ giải SAT dựa trên thuật toán CDCL Mục tiêu chính của CaDiCal không phải hiệu năng, mà là một cơ sở thuật toán dễ hiểu và mở rộng. Vì vậy đặt nền móng cho nhiều SAT Solver khác sau này.
- **Kissat**: Dựa trên CaDiCal, nhưng được viết lại bằng C, với nhiều cải tiến về cấu trúc dữ liệu, xếp lịch tiến trình xử lý, tối ưu hóa cài đặt thuật toán. Xếp hạng đầu trong hạng mục các công cụ giải SAT tuần tự trong cuộc thi giải SAT quốc tế năm 2022.
- **MiniSAT**: Một SAT Solver hiện đại, trở thành tiêu chuẩn trong công nghiệp. Dựa trên thuật toán CDCL, và giành chiến thắng trong cuộc thi giải SAT quốc tế năm 2005. Đây vẫn là một trong những SAT Solver được sử dụng nhiều nhất do chất lượng mã nguồn cao, rõ ràng và dễ cải tiến.
- **Glucose**: SAT Solver được dựa trên MiniSAT, áp dụng thêm nhiều kỹ thuật mới như phương pháp học mệnh đề hiện đại và giải song song.
- **Gini**: Một solver hiện đại được viết bằng Go, điểm đặc biệt của solver này là giao thức chia sẻ tính toán, cho phép giải song song sử dụng các goroutine. Đây cũng là solver được chọn để giải bài toán PESP khi thực nghiệm.

Để giải các bài toán thực tế sử dụng SAT Solver, ta cần định nghĩa hình thức các yêu cầu nghiệp vụ của bài toán thành các logic mệnh đề, giải bài toán SAT, sau đó suy luận kết quả từ đầu ra của SAT Solver. Sơ đồ có thể giải một bài toán sử dụng SAT Solver được minh họa trong Hình 7. Thuật toán encoding và decoding là một quá trình phức tạp, chương tiếp theo sẽ minh họa rõ hơn quá trình này.



Hình 7: Sơ đồ giải bài toán thực tế sử dụng SAT Solver

FIXME: có thể ví dụ thêm về 1 giải 1 bài toán đơn giản giải bằng SAT hoặc move phần giới thiệu encoding lên trên này.

Mô hình bài toán PESP về bài toán SAT

Trong chương này, khóa luận sẽ trình bày thuật toán nhằm chuyển hóa một bài toán PESP thành bài toán SAT. Điều này có nghĩa là, khi cho trước một mạng lưới sự kiện định kỳ N, ta cần tìm ra một lịch trình hợp lệ hoặc chứng minh rằng không tồn tại một giải pháp như vậy thỏa mãn. Các thuộc tính vào ràng buộc của bài toán phải được mã hóa thành bài toán SAT, tức là một công thức mệnh đề ở dạng chuẩn tắc hội (CNF), và sau đó được chứng minh bởi một bộ giải SAT.

Nếu bộ giải SAT trả về UNSAT, chúng ta biết rằng không tồn tại một lịch trình hợp lệ cho mạng lưới sự kiện định kỳ N đã mã hóa. Ngược lại, nếu nhận được một nghiệm cho công thức mệnh đề, điều đó đảm bảo rằng có tồn tại một lịch trình hợp lệ cho N. Tính chính xác của thuật toán mã hóa các ràng buộc của bài toán về dạng chuẩn tắc hội và cách truy xuất lịch trình hợp lệ từ nghiệm sẽ được chứng minh ở phần sau.

Sẽ có hai cách mã hóa khác nhau được giới thiệu cho một bài toán PESP đã được giảm thành bài toán SAT. Đầu tiên, mã hóa trực tiếp cho các biến của các miền hữu hạn sẽ được trình bày ở Mục 3.2 và cách triển khai cụ thể cho PESP ở Mục 3.3. Thứ hai, mã hóa thứ tự cho các biến có miền hữu hạn có thứ tự sẽ được định nghĩa trong Mục 3.4 và cách nó được sử dụng để mã hóa PESP thành bài toán SAT ở Mục 3.5.

Cuối chương này, hai phương pháp sẽ được so sánh để đưa ra đánh giá và nhận định về việc mã hóa thứ tự có thể giải quyết bài toán PESP nhanh hơn bao nhiêu so với mã hóa trực tiếp và kích thước của nó nhỏ hơn bao nhiêu.

Nhiều nghiên cứu liên quan đề xuất hai phương pháp mã hóa bài toán lập lịch định kỳ (PESP) về bài toán SAT là Direct Encoding và Order Encoding.

3.1 Mã hóa đại lượng rời rạc

Các biến số trong bài toán PESP (các tiềm năng sự kiện) là các biến đại số rời rạc, tức là chúng có thể nhận các giá trị từ một tập hữu hạn, thay vì có thể nhận bất kỳ giá trị nào trong một khoảng liên tục. Ví dụ, một biến rời rạc có thể đại diện cho một khoảng thời gian nhất định giữa các sự kiện, và giá trị của nó có thể là các số nguyên từ 1 đến 10, biểu

thị số phút. Tuy nhiên, logic mệnh đề và biểu thức chuẩn tắc hội chỉ có thể biểu diễn hai trạng thái logic là 0 và 1, tương ứng với giá trị "đúng" và "sai". Điều này tạo ra một vấn đề khi ta cần biểu diễn các giá trị rời rạc, vì không thể trực tiếp gán chúng vào các biến logic chỉ có hai trạng thái.

Do đó, chúng ta cần tìm cách mã hóa các biến rời rạc này sang không gian logic, tức là chuyển đổi các biến có nhiều giá trị tiềm năng sang các tổ hợp biến logic có thể được biểu diễn trong biểu thức mệnh đề. Ví dụ, nếu một biến rời rạc có thể nhận ba giá trị là 1, 2 và 3, ta có thể mã hóa chúng bằng cách sử dụng hai biến logic x_1 và x_2, trong đó:

- $x_1 = 0$ và $x_2 = 0$ có thể đại diện cho giá trị 1,
- $x_1 = 0$ và $x_2 = 1$ có thể đại diện cho giá trị 2,
- $x_1 = 1$ và $x_2 = 0$ có thể đại diện cho giá trị 3.

Cách mã hóa này cho phép chúng ta sử dụng công cụ của logic mệnh đề để xử lý các biến rời rạc, biến chúng thành các biến logic có thể được giải bằng bộ giải SAT. Điều quan trọng là việc chuyển đổi này phải được thực hiện sao cho các ràng buộc của bài toán ban đầu vẫn được duy trì trong không gian logic, đảm bảo rằng các giá trị logic được chọn phải tương ứng với một nghiệm hợp lệ trong miền rời rạc. Các phương pháp mã hóa như vậy được đề xuất và cải tiến bởi nhiều nghiên cứu, tiêu biểu như Direct Encoding [16], Product Encoding [17], Support Encoding... Sau đây khóa luận giới thiệu về hai phương pháp ứng dụng trong giải bài toán PESP: Mã hóa trực tiếp (Direct Encoding) và mã hóa thứ tự (Order Encoding).

3.1.1 Mã hóa trực tiếp (Direct Encoding)

Mã hóa trực tiếp, hay còn gọi là mã hóa nhị thức, là phương pháp đơn giản nhất để mã hóa các biến rời rạc. Nguyên lý chính của phương pháp này là loại bỏ từng cặp giá trị không thể cùng thỏa mãn đồng thời, đảm bảo rằng chỉ một giá trị trong số các giá trị có thể được chọn. Để làm rõ hơn, ta xét ví dụ sau:

Ví dụ 3.1.1.1: Cho $x \in \{1, 2, 3\}$. Hiển nhiên ta có những mệnh đề đúng sau:

$$x = 1 \vee x = 2 \vee x = 3$$
$$x = 1 \Rightarrow x \neq 2 \wedge x \neq 3$$
$$x = 2 \Rightarrow x \neq 1 \wedge x \neq 3$$
$$x = 3 \Rightarrow x \neq 1 \wedge x \neq 2$$

Sử dụng các biến logic: x_1, x_2, x_3 tương ứng với mệnh đề $x = 1, x = 2, x = 3$ ta có:

$$x_1 \vee x_2 \vee x_3$$
$$x_1 \Rightarrow \neg x_2 \wedge \neg x_3$$
$$x_2 \Rightarrow \neg x_1 \wedge \neg x_3$$
$$x_3 \Rightarrow \neg x_1 \wedge \neg x_2$$

$\Leftrightarrow$

$$x_1 \vee x_2 \vee x_3$$
$$(x_1 \Rightarrow \neg x_2) \wedge (x_1 \Rightarrow \neg x_3)$$
$$(x_2 \Rightarrow \neg x_1) \wedge (x_2 \Rightarrow \neg x_3)$$
$$(x_3 \Rightarrow \neg x_1) \wedge (x_3 \Rightarrow \neg x_2)$$

$\Leftrightarrow$

$$x_1 \vee x_2 \vee x_3$$
$$(\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3)$$
$$(\neg x_2 \vee \neg x_1) \wedge (\neg x_2 \vee \neg x_3)$$
$$(\neg x_3 \vee \neg x_1) \wedge (\neg x_3 \vee \neg x_2)$$

$\Leftrightarrow$

$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3)$$

Tổng quát hóa ví dụ trên, ta có thể áp dụng phương pháp mã hóa trực tiếp cho bất kỳ tập giá trị hữu hạn nào $x \in \{1, 2, ..., n\}$. Khi đó, mã hóa trực tiếp được định nghĩa như sau:

Định nghĩa 3.1.1.1: Cho $x \in X |\ X = \{1, 2, ..., n |\ n \in \mathbb{N}\}$ và các mệnh đề: $x_1, x_2, ..., x_n$ đúng khi và chỉ khi $x = n$. Ta định nghĩa ánh xạ:

$$\text{encode\_direct}(X) = \bigvee_{i=1}^{n} x_i \wedge \left(\bigwedge_{i=1}^{n} \bigwedge_{j=i+1}^{n} (\neg x_i \vee \neg x_j) \right)$$

Phương pháp mã hóa trực tiếp đảm bảo rằng chỉ một biến logic duy nhất có giá trị "đúng" (true) trong khi tất cả các biến còn lại phải có giá trị "sai" (false) trong mọi suy diễn hợp lệ I.

3.1.2 Mã hóa thứ tự (Order Encoding)

Trái ngược với Chương 3.1.1, phần này giả định rằng miền hữu hạn có thứ tự. Ví dụ tốt nhất cho điều này là một tập con thực sự của tập số tự nhiên $\mathbb{N}$. Các số này luôn có thứ tự theo quan hệ "<". Trong phần tiếp theo, ta sẽ thảo luận cách mã hóa hiệu quả thuộc tính này vào một công thức mệnh đề. Vì trong khóa luận này, ta chỉ xét các biến có miền là một tập con,chính xác hơn là một khoảng, của các số tự nhiên $[a, b]$, nên ta biết cách áp dụng quan hệ thứ tự "<" của chúng. Nhìn chung, mọi tập hợp đều có thể xác định quan hệ thứ tự cụ thể. Tương tự, cùng tiếp cận phương pháp mã hóa này với một ví dụ.

Ví dụ 3.1.2.1: Cho $x \in \{1, 2, 3, 4, 5\} = [1, 5] \subset \mathbb{N}$. Ta có các mệnh đề đúng sau:

$$
\begin{aligned}
&x \geq 1 \Rightarrow \neg(x \leq 0) \\
&x \leq 5 \\
&\forall i \in \{1, 2, 3, 4, 5\} : (x \leq i - 1) \rightarrow (x \leq i)
\end{aligned}
$$

$\Leftrightarrow$

$$
\begin{aligned}
&x \geq 1 \Rightarrow \neg(x \leq 0) \\
&x \leq 5 \\
&\forall i \in \{1, 2, 3, 4, 5\} : \neg(x \leq i - 1) \vee (x \leq i)
\end{aligned}
$$

$\Leftrightarrow$

$$(\neg x \leq 0) \wedge (\neg x \leq 1 \vee x \leq 2) \wedge (\neg x \leq 2 \vee x \leq 3) \wedge (\neg x \leq 3 \vee x \leq 4) \wedge (x \leq 5)$$

Thế các mệnh đề: $x_i \Leftrightarrow x \leq i$ ta có:

$$(\neg x_0) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_3 \vee x_4) \wedge (x_5)$$

Tương tự, ta có thể áp dụng phương pháp mã hóa trực tiếp cho bất kỳ tập giá trị hữu hạn **có thứ tự** nào $x \in \{1, 2, ..., n\}$. Khi đó, mã hóa thứ tự được định nghĩa như sau:

Định nghĩa 3.1.2.1: Cho $x \in X | \ X = \{1, 2, ..., n | \ n \in \mathbb{N}\}$ và các mệnh đề: $x_0, x_1, x_2, ..., x_n$ đúng khi và chỉ khi $x \leq n$. Ta định nghĩa ánh xạ:

$$\text{encode\_order(X)} = (\neg x_0) \wedge \bigwedge_{i=2}^{n-1} (\neg x_{i-1} \vee x_i) \wedge (x_n) \tag{5}$$

Khác với mã hóa trực tiếp, mệnh đề trong mã hóa thứ tự mang ý nghĩa rộng hơn:

$$x_i \Leftrightarrow x \leq i$$

Điều này có nghĩa, với mỗi suy diễn I, việc trích xuất thông tin từ mệnh đề không quá đơn giản:

$$x_i^I = \text{true} \Leftrightarrow x \leq i$$
$$x_i^I = \text{false} \Leftrightarrow x \nleq i$$
$$\Rightarrow x = ?$$

Vì vậy ta cần chứng minh luôn có thể suy diễn thông tin từ một suy diễn khi mã hóa thứ tự, được chứng minh trong định lý sau:

Định lý 3.1.2.1: Cho $x \in X | \ X = \{1, 2, 3, ..., n\} \mid n \in \mathbb{N}$ với I là một suy diễn thỏa mãn Định nghĩa 3.1.2.1:

$$I \vDash \text{encode\_order}(X) \Leftrightarrow \exists k \in [1, n] : \forall i \in [1, k-1] : I \nvDash x_i$$
$$\forall j \in [k, n] : I \vDash x_j$$

Do đó, ta có cách trích xuất giá trị của x từ một suy diễn I như sau:

Định nghĩa 3.1.2.2: Cho $x \in X | \ X = \{1, 2, 3, ..., n\} \mid n \in \mathbb{N}$ với $I \vDash \text{encode\_order(X)}$. Khi đó tồn tại duy nhất một giá trị $k \in X$ mà:

$$x = k \in X \Leftrightarrow x_{k-1}^I = \text{false} \wedge x_k^I = \text{true}$$

Để hiểu rõ hơn các truy xuất thông tin từ suy diễn, cùng xem lại Ví dụ 3.1.2.1:

Ví dụ 3.1.2.2: Cho $x \in X = \{1, 2, 3, 4, 5\}$

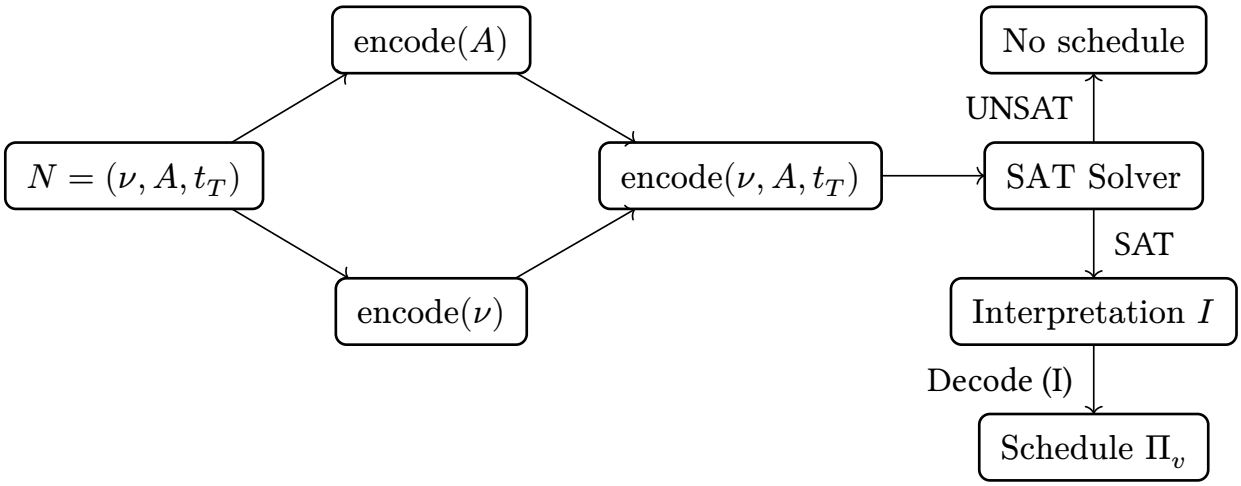
$$\text{encode\_order(X)} = (\neg x \leq 0) \wedge (\neg x \leq 1 \vee x \leq 2) \wedge$$
$$(\neg x \leq 2 \vee x \leq 3) \wedge (\neg x \leq 3 \vee x \leq 4) \wedge (x \leq 5)$$

Xét một suy diễn hợp lệ I:

$$x_0 = \text{false}$$
$$x_1 = \text{false}$$
$$x_2 = \text{false}$$
$$x_3 = \text{true}$$
$$x_4 = \text{true}$$
$$x_5 = \text{true}$$

Từ Định nghĩa 3.1.2.2, ta suy ra: x = 3

3.2 PESP as Direct Encoding



Hình 8: Sơ đồ tổng quan giải bài toán PESP với mã hóa trực tiếp

Để mã hóa trực tiếp bài toán PESP thành biểu thức mệnh đề, trước tiên ta cần mã hóa các tiềm năng sử kiện π_i. Nhắc lại Hệ quả 1.1.1.2, các tiềm năng sự kiện π_i đều thỏa mãn:

$$\forall \pi_i \in \nu \mid \pi_i \in [0, t_T - 1] = \{0, 1, 2, ..., t_T - 1\} \tag{6}$$

Vì vậy, ta dễ dàng mã hóa toàn bộ tiềm năng sự kiện dựa theo Định nghĩa 3.1.1.1:

$$\Omega^{\nu}_{\text{direct}} := \bigwedge_{n \in \nu} \text{encode\_direct}(\pi_n) \tag{7}$$

do encode_direct(π_n) là một biểu thức chuẩn tắc hội và $\Omega^{n}_{\text{direct}}$ là hội những mệnh đề này, $\Omega^{\nu}_{\text{direct}}$ là một biểu thức dạng chuẩn tắc hội.

Với mỗi suy diễn I thoả mãn Phương trình 7, ta luôn suy ra được một lịch trình hợp lệ theo định lý sau:

$$\forall n \in \nu : \Pi_\nu(n) = i \Leftrightarrow p^I_{\pi_n, i} = \text{true} \tag{8}$$

trong đó $p^I_{\pi_n,i}$ là mệnh đề tương ứng với $\pi_n = i$ trong suy diễn I.

Tiếp theo, ta cần mã hóa các ràng buộc thời gian và ràng buộc đối xứng như ở Định nghĩa 1.1.6 và Định nghĩa 1.1.7. Ý tưởng chính là loại các cặp tiềm năng sự kiện không thỏa mãn ràng buộc, chuyển hóa chúng thành các mệnh đề. Do tập giá trị của các tiềm năng là hữu hạn ($[0, t_T - 1]$), ta loại tất cả các khả năng không khỏa mãn bằng các mệnh đề đảo nghịch.

Thật vậy, ta xét một ràng buộc bất kỳ $a = \Big((i, j), [l_a, u_a]_{t_T}\Big) \in A$, ta định nghĩa:

$$\text{encode\_direct\_cons}(a) = \bigwedge_{m,n \in P_a} \Big(\neg p_{\pi_i, m} \vee \neg p_{\pi_j, n}\Big) \tag{9}$$

trong đó:

- $m, n \in [0, t_T - 1]$
- $p_{\pi_i,m}, p_{\pi_j,n}$ là các biến logic tương ứng với các tiềm năng: $\pi_i = m, \pi_j = n$
- $P_a := \{(m, n) \mid (m, n) \text{ không thỏa mãn } a\}$

Nhằm hiểu rõ hơn Phương trình 9, ta xét ví dụ cụ thể sau:

Ví dụ 3.2.1: Cho hai sự kiện A, B và ràng buộc thời gian $a = ((A, B), [3, 7]_8)$ như ở Ví dụ 1.1.3.

Dễ thấy: $(\pi_A, \pi_B) = (6, 7)$ không thỏa mãn a ($7 - 6 = 1 \notin [3, 7]_8$). Vậy ta cần loại khả năng $\pi_A = 6$ và $\pi_B = 7$, tức là

$$\begin{aligned} &\neg(\pi_A = 6 \wedge \pi_B = 7) \\ \Leftrightarrow\ &\neg\Big(p_{\pi_A,6} \wedge p_{\pi_B,7}\Big) \\ \Leftrightarrow\ &\neg p_{\pi_A,6} \vee \neg p_{\pi_B,7} \end{aligned} \tag{10}$$

Tương tự ta có:

$$
\begin{aligned}
P_a = \{&(0,0),(0,1),(0,2),(1,0),(1,1),(1,2),\\
&(1,3),(2,0),(2,1),(2,2),(2,3),(2,4),(3,0),\\
&(3,1),(3,2),(3,3),(3,4),(3,5),(4,0),(4,1),\\
&(4,2),(4,3),(4,4),(4,5),(4,6),(5,0),(5,1),\\
&(5,2),(5,3),(5,4),(5,5),(5,6),(5,7),(6,0),\\
&(6,1),(6,2),(6,3),(6,4),(6,5),(6,6),(6,7),\\
&(7,0),(7,1),(7,2),(7,3),(7,4),(7,5),(7,6),\\
&(7,7)\}
\end{aligned} \tag{11}
$$

$$
\Rightarrow \text{encode\_direct\_cons}(a) = \left(\neg p_{\pi_A,0} \vee \neg p_{\pi_B,3}\right) \wedge \ldots \wedge \left(\neg p_{\pi_A,7} \vee \neg p_{\pi_B,7}\right)
$$

Áp dụng Phương trình 9 với tất cả ràng buộc $a \in A$, ta có được biểu thức mã hóa trực tiếp các ràng buộc:

$$
\Psi^A_{\text{direct}} \coloneqq \bigwedge_{a \in A} \text{encode\_direct\_cons(a)} \tag{12}
$$

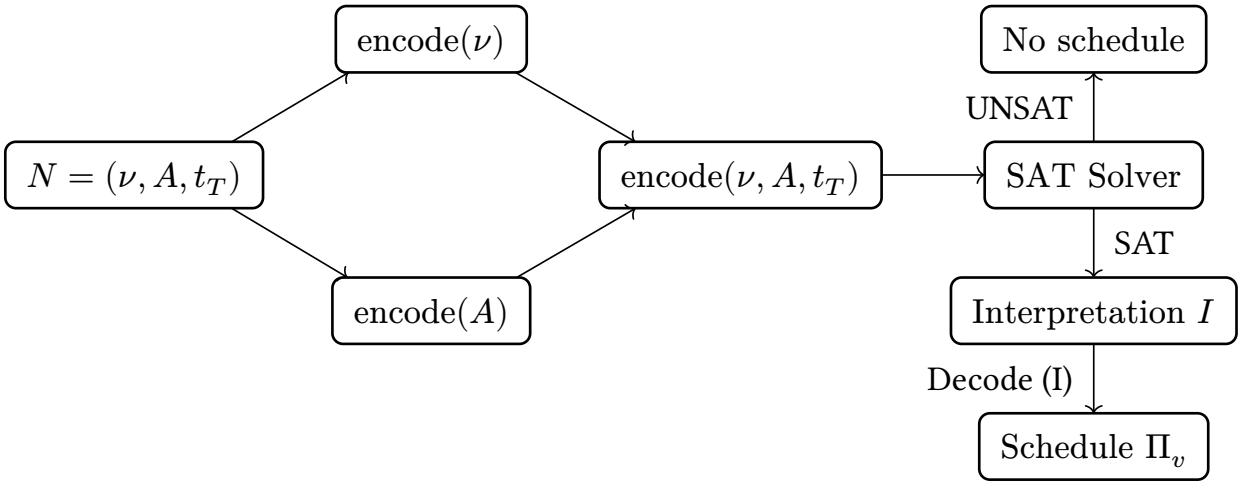
Cuối cùng, sử dụng Phương trình 7 và Phương trình 9, ta mã hóa trực tiếp toàn bộ mạng định kỳ như sau:

$$
\text{encode\_direct\_pesp}(\nu, A, t_T) = \Omega^\nu_{\text{direct}} \wedge \Psi^A_{\text{direct}} \tag{13}
$$

với $\Omega^\nu_{\text{direct}}, \Psi^A_{\text{direct}}$ là các biểu thức có dạng tương ứng như Phương trình 7 và Phương trình 9.

Dễ thấy $\Omega^\nu_{\text{direct}}$ thỏa mãn dạng chuẩn tắc hội. Tương tự, Ψ^A_{direct} là hội của các biểu thức chuẩn tắc hội, nên cũng là một biểu thức chuẩn tắc hội. Do đó, $\text{encode\_direct\_pesp}(\nu, A, t_T)$ cũng có dạng chuẩn tắc hội. Như vậy, chúng ta đã thành công trong việc mã hóa bài toán PESP thành một biểu thức chuẩn tắc hội, mà các SAT Solver hiện đại có thể giải quyết một cách dễ dàng. Từ suy diễn I thu được từ SAT Solver, chúng ta có thể dễ dàng truy xuất ra lịch trình hợp lệ bằng cách sử dụng hàm Phương trình 8.

3.3 PESP as Order Encoding



Hình 9: Sơ đồ tổng quan giải bài toán PESP với mã hóa thứ tự

Tương tự mã hóa trực tiếp, khóa luận sẽ trình bày phương pháp mã hóa thứ tự gồm hai phần chính. Trước hết, ta mã hóa các tiềm năng sự kiện như đã trình bày ở Chương 3.1.2. Sau đó ta sẽ mã hóa các ràng buộc trong miền xác định thứ tự. Cuối cùng, ta tổng hợp các mệnh đề và giải bằng SAT Solver.

Để mã hóa trực tiếp bài toán PESP thành biểu thức mệnh đề, trước tiên ta cần mã hóa các tiềm năng sử kiện π_i. Nhắc lại Hệ quả 1.1.1.2, các tiềm năng sự kiện π_i đều thỏa mãn:

$$\forall \pi_i \in \nu \mid \pi_i \in [0, t_T - 1] \Leftrightarrow 0 \leq \pi_i \leq t_T - 1 \tag{14}$$

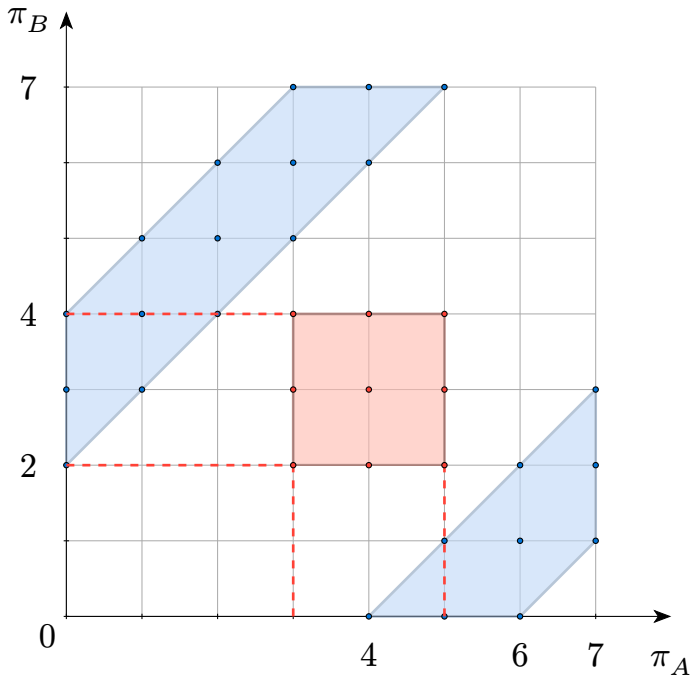
Vì vậy, ta dễ dàng mã hóa toàn bộ tiềm năng sự kiện dựa theo Định nghĩa 3.1.2.1:

$$\Omega^{\nu}_{\text{order}} := \bigwedge_{n \in \nu} \text{encode\_order}(\pi_n) \tag{15}$$

do encode_order(π_n) là một biểu thức chuẩn tắc hội và $\Omega^{n}_{\text{order}}$ là hội những mệnh đề này, $\Omega^{\nu}_{\text{order}}$ là một biểu thức dạng chuẩn tắc hội.

Tiếp theo, khóa luận trình bày chi tiết cách mã hóa ràng buộc thời gian và ràng buộc đối xứng. Tư tưởng căn bản sẽ tương tự như mã hóa trực tiếp, loại bỏ các miền không khỏa mãn trong tập xác định. Tuy nhiên khi sử dụng mã hoá thứ tự, ta có thể loại bỏ từng vùng các hình chữ nhật song song với trục tọa độ do đó tối ưu hiệu quả mã hóa như Hình 10

3.3.1 Mã hóa thứ tự ràng buộc thời gian



Hình 10: Minh họa loại bỏ miền không thỏa mãn ràng buộc thời gian $((A, B), [2, 4]_8)$

Để hiểu ý tưởng chính của thuật toán, ta xét ví dụ cụ thể sau:

> *Ví dụ 3.3.1.1*: Cho ràng buộc thời gian $a = ((A, B), [2, 4]_8)$ với π_A, π_B lần lượt là các tiềm năng tương ứng với A và B. Đặt $p_{\pi_A,i}, p_{\pi_B,j}$ lần lượt là các biến logic tương ứng với mệnh đề $\pi_A \leq i, \pi_B \leq j$

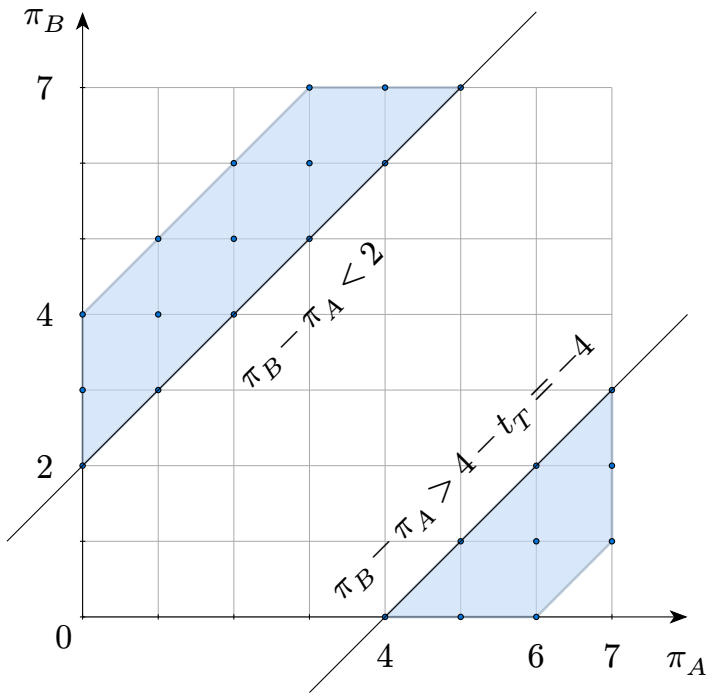
Dễ thấy một vùng không thỏa mãn ràng buộc như minh họa ở Hình 10:

$$r = ([2, 4] \times [2, 4]) \in P_a \tag{16}$$

$$\begin{aligned}
&\Leftrightarrow \nexists(\pi_A, \pi_B) : \pi_A \leq 4, \pi_A \geq 2, \pi_B \leq 4, \pi_B \geq 2 \\
&\Leftrightarrow \neg(\pi_A \leq 4 \wedge \pi_A \geq 2 \wedge \pi_B \leq 4 \wedge \pi_B \geq 2) \\
&\Leftrightarrow \neg(\pi_A \leq 4 \wedge \neg\pi_A \leq 1 \wedge \pi_B \leq 4 \wedge \neg\pi_B \leq 1) \qquad (17) \\
&\Leftrightarrow \neg\pi_A \leq 4 \vee \pi_A \leq 1 \vee \neg\pi_B \leq 4 \vee \pi_B \leq 1 \\
&\Leftrightarrow \neg p_{\pi_A,4} \vee p_{\pi_A,1} \vee \neg p_{\pi_B,4} \vee p_{\pi_B,1} = F
\end{aligned}$$

Do F là một mệnh đề chuẩn tắc hội, ta có được một mệnh đề ràng buộc tương ứng với phần bị loại bỏ trong Hình 10. Như vậy ta đã biết cách loại bỏ một hình chữ nhật khỏi không gian tìm kiếm. Vấn đề còn lại là tìm tất cả các hình chữ nhật nhằm lấp đầy vùng không thỏa mãn.

Cùng xem lại Hình 10 ở một góc nhìn khác. Ta thấy hai miền nghiệm tương ứng với $[2.4]$, $[-6,-4]$ và khoảng vô nghiệm giữa chúng $(-4,2)$. Sau đây ta sẽ định nghĩa các hàm số nhằm phủ miền này bằng tập hợp các hình chữ nhật ở Hình 10. Bằng cách tương tự, ta cũng có thể phủ hai vùng vô nghiệm còn lại$((4,8),(-8,-6))$.



Hình 11: Bản chất của miền vô nghiệm$((A,B),[2,4]_8)$

Định nghĩa 3.3.1.1: Cho $u, l \in \mathbb{Z}$ là hai số nguyên với u < l.

$$\begin{aligned} \delta : \mathbb{Z} \times \mathbb{Z} &\to \mathbb{Z} \\ (l,u) &\mapsto l - u - 1 \end{aligned} \tag{18}$$

được gọi là *khoảng cách trong* giữa u và l.

Định nghĩa 3.3.1.2: Cho $u, l \in \mathbb{Z}$ là hai số nguyên với u < l.

$$\begin{aligned} \delta y : \mathbb{Z} \times \mathbb{Z} &\to \mathbb{Z} \\ (l,u) &\mapsto \left\lfloor \frac{\delta(l,u)}{2} \right\rfloor \end{aligned} \tag{19}$$

được gọi là chiều rộng của hình chữ nhật giữa u và l.

Định nghĩa 3.3.1.3: Cho $u, l \in \mathbb{Z}$ là hai số nguyên với u < l.

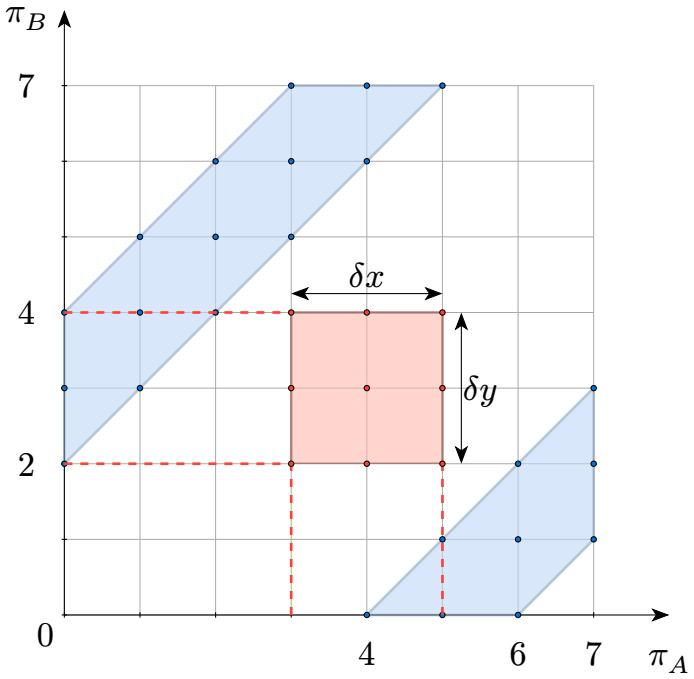
$$\begin{aligned} \delta x : \mathbb{Z} \times \mathbb{Z} &\to \mathbb{Z} \\ (l, u) &\mapsto \left\lceil \frac{\delta(l,u)}{2} \right\rceil - 1 \end{aligned} \tag{20}$$

được gọi là chiều dài của hình chữ nhật giữa u và l.

Ví dụ 3.3.1.2: Xét hai sự kiện A, B với $a = ((A, B), [2, 4]_8)$ như ở Ví dụ 3.3.1.1, với π_A, π_B là hai tiềm năng sự kiện. Chọn $l = 2, u = -4$, ta có:

$$\begin{aligned} \delta(2, -4) &= 2 - (-4) - 1 = 5 \\ \delta y(2, 4) &= \left\lfloor \frac{\delta(2,-4)}{2} \right\rfloor = 2 \\ \delta x(2, 4) &= \left\lceil \frac{\delta(2,-4)}{2} \right\rceil - 1 = 2 \end{aligned} \tag{21}$$

được minh họa như Hình 12.



Hình 12: Minh họa vùng xác định kích thước vùng không thỏa mãn

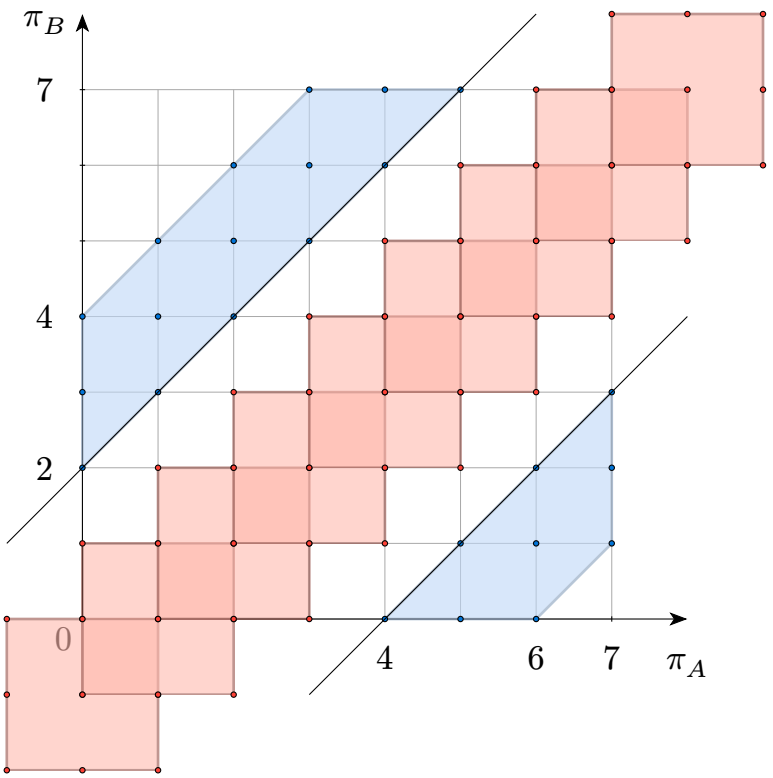
Định nghĩa 3.3.1.4: Cho $u, l \in \mathbb{Z}$ với $u < l$ và $t_T \in \mathbb{N}$. Khi đó

$$\begin{aligned}\varphi_{t_T} : \mathbb{Z} \times \mathbb{Z} &\to 2^{2^{\mathbb{Z}} \times 2^{\mathbb{Z}}} \\ (l, u) &\mapsto \{([x, x + \delta x(l,u)] \times [y, y + \delta y(l,u)]) | \\ &\quad \forall y \in [-\delta y(l,u), t_T - 1] : \\ &\quad x = y - u - 1 - \delta x(l,u)\} \\ &\quad x + \delta x(l,u) \geq 0 \wedge x \leq t_T - 1\end{aligned} \tag{22}$$

là ánh xạ tới tập hợp hình chữ nhật giữa u và l.

Áp dụng Định nghĩa 3.3.1.4 với hai sự kiện A, B, $a = ((A, B), [2, 4]_8)$ như ở Ví dụ 3.3.1.1, với π_A, π_B là hai tiềm năng sự kiện. Chọn $l = 2, u = -4$, ta thấy tập hợp các hình chữ nhật hoàn toàn phủ vùng vô nghiệm trong khoảng $(-4, 2)$, được minh họa ở Hình 13

$$\begin{aligned}\varphi_{t_T}(2, -4) = \{&([-1, 1] \times [-2, 0]), ([0, 2] \times [-1, 1]), ([1, 3] \times [0, 2]) \\ &([2, 4] \times [1, 3]), ([3, 5] \times [2, 4]), ([4, 6] \times [3, 5]) \\ &([5, 7] \times [4, 6]), ([6, 8] \times [5, 7]), ([7, 9] \times [6, 8])\}\end{aligned} \tag{23}$$



Hình 13: Minh họa tập hợp hình chữ nhật phủ vùng vô nghiệm $((A, B), [2, 4]_8)$

Tương tự, ta cần phủ hai vùng vô nghiệm còn lại ở góc trái trên và góc phải, tương ứng với các khoảng $(4, 8), (-8, -6)$. Từ đó ta có công thức tổng quát cho ràng buộc thời gian $\Big((A, B), [l, u]_{t_T}\Big)$

Định nghĩa 3.3.1.5: Cho ràng buộc thời gian $a = \Big((A, B), [l, u]_{t_T}\Big)$. Với

$$\begin{aligned} k = -1 \text{ nếu } 0 \in [l, u]_{t_T} \\ k = 0 \text{ nếu } 0 \notin [l, u]_{t_T} \end{aligned} \tag{24}$$

$$\begin{aligned} \zeta_{t_T} : \mathbb{Z} \times \mathbb{Z} &\to 2^{(\mathbb{Z}\times\mathbb{Z})\times(\mathbb{Z}\times\mathbb{Z})} \\ (l, u) &\mapsto \bigcup_{i \in [k, 1]} \varphi_{t_T}(l + i \cdot t_T, u + (i - 1) \cdot t_T) \end{aligned} \tag{25}$$

là hàm số sinh tất cả các hình chữ nhật phủ miền vô nghiệm của a

Tiếp theo, ta cần mã hóa các hình chữ nhật này thành các mệnh đề chuẩn tắc. Điều này có thể tổng quát trực tiếp từ Ví dụ 3.3.1.1. Từ đó ta có ánh xạ tổng quát ràng buộc thời gian thành các mệnh đề chuẩn tắc.

Định nghĩa 3.3.1.6: Cho $A = ([x_1, x_2] \times [y_1, y_2]) \in 2^{\mathbb{Z}} \times 2^{\mathbb{Z}}$ với $(x, y) \in A$. Khi đó

$$\begin{aligned} \text{encode\_order\_rect} : 2^{\mathbb{Z}} \times 2^{\mathbb{Z}} &\to \Sigma_{\text{SAT}} \\ [x_1, x_2] \times [y_1, y_2] &\mapsto \Big(\neg p_{x,x_2} \vee p_{x,x_1-1} \vee \neg p_{y,y_2} \vee p(y, y_1 - 1)\Big) \end{aligned} \tag{26}$$

là mã hóa thứ tự loại trừ miền phủ bởi A, trong đó $p_{l,k}$ là các biến logic tương ứng với mệnh đề $l \leq k$ ở Định nghĩa 3.1.2.1

Định nghĩa 3.3.1.7: Cho ràng buộc thời gian $a = \Big((A, B), [l, u]_{t_T}\Big)$. Khi đó

$$\begin{aligned} &\text{encode\_order\_time\_con} : C \to L(\Sigma_{\text{SAT}}) \\ &\Big((A, B), [l, u]_{t_T}\Big) \mapsto \bigwedge_{A \in \zeta_{t_T}(l,u)} \text{encode\_order\_rect}(A) \end{aligned} \tag{27}$$

là hàm số mã hóa thứ tự ràng buộc thời gian

3.3.2 Mã hóa thứ tụ ràng buộc đối xứng

Mã hóa ràng buộc đối xứng hoàn toàn tương tự với ràng buộc thời gian. Ta dễ dàng có được các kết quả sau:

Định nghĩa 3.3.2.1: Cho $u, l \in \mathbb{Z}, u < l$ và $t_T \in \mathbb{N}$. Khi đó

$$\begin{aligned} \psi_{t_T} : \mathbb{Z} \times \mathbb{Z} &\to 2^{2^{\mathbb{Z}} \times 2^{\mathbb{Z}}} \\ (l, u) &\mapsto \{([x, x + \delta x(l, u)] \times [y, y + \delta y(l, u)]) \\ &\qquad \forall y \in [-\delta y, t_T - 1] : \\ &\qquad x = -y + l - 1 - \delta x(l, u)\} \end{aligned} \tag{28}$$

là hàm số ánh xạ tất cả hình chữ nhật giữa u và l

Định nghĩa 3.3.2.2: Cho ràng buộc thời gian $a = \Big((A, B), [l, u]_{t_T}\Big)$. Với

$$\begin{aligned} k = 1 \text{ nếu } 0 \in [l, u]_{t_T} \\ k = 2 \text{ nếu } 0 \notin [l, u]_{t_T} \end{aligned} \tag{29}$$

$$\begin{aligned} \Lambda_{t_T} : \mathbb{Z} \times \mathbb{Z} &\to 2^{(\mathbb{Z} \times \mathbb{Z}) \times (\mathbb{Z} \times \mathbb{Z})} \\ (l, u) &\mapsto \bigcup_{i \in [0, k]} \varphi_{t_T}(l + i \cdot t_T, u + (i - 1) \cdot t_T) \end{aligned} \tag{30}$$

là hàm số sinh tất cả các hình chữ nhật phủ miền vô nghiệm của a

Định nghĩa 3.3.2.3: Cho ràng buộc đối xứng $a = \Big((A, B), [l, u]_{t_T}\Big)$. Khi đó

$$\begin{aligned} \text{encode\_order\_sym\_con} : S &\to L(\Sigma_{\text{SAT}}) \\ \Big((A, B), [l, u]_{t_T}\Big) &\mapsto \bigwedge_{A \in \Lambda_{t_T}(l,u)} \text{encode\_order\_rect}(A) \end{aligned} \tag{31}$$

là hàm số mã hóa thứ tự ràng buộc đối xứng

Với toàn bộ thông tin từ các phần trước, ta có hàm số mã hóa thứ tự toàn bộ bài toán PESP như sau:

Định nghĩa 3.3.2.4: Cho $A = S \cup C$ là tập hợp các ràng buộc ở Định nghĩa 1.1.3, khi đó:

$$\begin{aligned} \text{encode\_order\_con} : A &\to L(\Sigma_{\text{SAT}}) \\ a &\mapsto \begin{cases} \text{encode\_order\_time\_con}(a) \text{ nếu } a \in C \\ \text{encode\_order\_sym\_con}(a) \text{ nếu } a \in S \end{cases} \end{aligned} \tag{32}$$

Định nghĩa 3.3.2.5: Cho $A = S \cup C$ là tập hợp các ràng buộc ở Định nghĩa 1.1.3, khi đó:

$$\Psi^A_{\text{order}} = \bigwedge_{a \in A} \text{encode\_order\_con}(a) \tag{33}$$

Định nghĩa 3.3.2.6: Cho $N = (\nu, A, t_T)$ là mạng sự kiện định kỳ như đã định nghĩa ở Định nghĩa 1.1.3, khi đó:

$$\begin{aligned} \text{encode\_direct\_pesp} : 2^{\nu^+} \times 2^{A^+_{t_T}} \times 2^{\mathbb{N}} &\to L(\Sigma_{\text{SAT}}) \\ (\nu, A, t_T) &\mapsto \left(\Omega^\nu_{\text{order}} \wedge \Psi^A_{\text{order}}\right) \end{aligned} \tag{34}$$

là hàm số mã hóa thứ tự của mạng định kỳ N.

3.4 So sánh Direct encoding và Order encoding

Trong chương này khóa luận sẽ so sánh hai phương pháp mã hóa trực tiếp và thứ tự ở các thông số như số mệnh đề và số biến. Các thông số khác như thời gian giải thực tế sẽ được trình bày ở Chương 4. Để thuận tiện, ta định nghĩa thêm một số khái niệm sau.

Định nghĩa 3.4.1: Cho $F \in L(\Sigma_{\text{SAT}})$ là một biểu thức chuẩn tắc hội. Khi đó:

$$|F| \in \mathbb{N} \tag{35}$$

là số mệnh đề (số tuyển sơ cấp) của F.

Định nghĩa 3.4.2: Cho $F \in L(\Sigma_{\text{SAT}})$ là một biểu thức chuẩn tắc hội. Khi đó:

$$|\text{vars}(F)| \in \mathbb{N} \tag{36}$$

là số biến được sử dụng trong F.

3.4.1 Số biến

Với F là biểu thức chuẩn tắc hội sinh ra từ mã hóa trực tiếp của mạng định kỳ $N = (\nu, A, t_T)$. Từ Định nghĩa 3.1.1.1 ta thấy với mỗi tiềm năng $\pi_n (n \in \nu)$, ta cần t_T biến vì $\pi_n \in [0, t_T - 1]$. Do đó

$$|\text{vars}(\text{encode\_direct\_pesp}(N))| = |\text{vars}(F)| = t_T \cdot |\nu| \tag{37}$$

tức là số biến bằng tích của chu kỳ và số sự kiện.

Tương tự ta xem xét G là biểu thức chuẩn tắc hội sinh từ mã hóa thứ tự của mạng định kỳ $N = (\nu, A, t_T)$. Từ Định nghĩa 3.1.2.1

$$|\text{vars}(\text{encode\_order\_pesp}(N))| = |\text{vars}(G)| = (t_T - 1) \cdot |\nu| \tag{38}$$

Trong thực tế, chu kì $t_T = 60$ hoặc $t_T = 120$. Vì vậy sự khác biệt về số biến là không lớn.

$$|\text{vars}(\text{encode\_direct\_pesp}(N))| \approx |\text{vars}(\text{encode\_order\_pesp}(N))| \tag{39}$$

3.4.2 Số mệnh đề

Để ước tính số mệnh đề sử dụng, ta cần ước tính lần lượt số mệnh đề dùng cho mã hóa tiềm năng và số biến mã hóa ràng buộc, Ω_t^ν và $\Psi_t^A t \in \{\text{direct}, \text{order}\}$. Dễ thấy

$$\begin{gathered} \left|\Omega_t^\nu \wedge \Psi_t^A\right| = |\Omega_t^\nu| + |\Omega_t^\nu| \\ t \in \{\text{direct}, \text{order}\} \end{gathered} \tag{40}$$

Đầu tiên, với mã hóa biến Ω_t^ν, do các biến có cùng tập xác định $[0, t_T - 1]$.

$$\begin{aligned} |\Omega_t^\nu| &= |\nu| \cdot |\text{encode\_direct}(\pi_n)| \\ |\Omega_t^\nu| &= |\nu| \cdot |\text{encode\_order}(\pi_n)| \\ n &\in \nu \end{aligned} \tag{41}$$

Từ các kết quả ở Chương 3.1, ta có:

$$\begin{aligned} |\text{encode\_direct}(\pi_n)| &= \frac{|[0, t_T - 1]| \cdot (|[0, t_T - 1]| - 1)}{2} \\ &= \frac{t_T \cdot (t_T - 1)}{2} \\ |\text{encode\_order}(\pi_n)| &= |[0, t_T - 1]| - 2 \\ &= t_T - 2 \end{aligned} \tag{42}$$

Từ Phương trình 41 và Phương trình 42 kết hợp với kí pháp $O(n)$, ta có:

$$\begin{aligned} |\Omega_{\text{direct}}^\nu| &\in O(t^2|\nu|) \\ |\Omega_{\text{order}}^\nu| &\in O(t|\nu|) \end{aligned} \tag{43}$$

Tiếp theo, ta cần xem xét mã hóa các ràng buộc $\Psi_t^A t$. Do mã hóa trực tiếp loại trừ các căp không thỏa mãn P_a nên ta có:

$$\left|\Psi_{\text{direct}}^A\right| = \sum_{a \in A} |P_a| \tag{44}$$

Hiển nhiên $|P_a| \in O(t_T^2)$ do $\forall (i, j) \in P_a \Rightarrow (i, j) \in [0, t_T - 1] \times [0, t_T - 1]$. Do đó

$$\left|\Psi_{\text{direct}}^A\right| \in O(t_T^2 |A|) \tag{45}$$

Với mã hóa thứ tự, mỗi ràng buộc chỉ cần hợp của 2 hoặc 3 lần (tùy theo $[l, u]$) φ_{t_T}, mà φ_{t_T} tỉ lệ với $[-\delta y, t_T - 1]$ theo như Định nghĩa 3.3.1.4. Do vậy

$$\left|\Psi^{A}_{\text{order}}\right| \in O(t_T|A|) \tag{46}$$

Kết hợp Phương trình 43, Phương trình 45 và Phương trình 46 ta có:

$$\begin{aligned} |\text{vars}(\text{encode\_direct\_pesp}(N))| &\in O(t^2(|\nu| + |A|)) \\ |\text{vars}(\text{encode\_order\_pesp}(N))| &\in O(t_T(|\nu| + |A|)) \end{aligned} \tag{47}$$

Từ kết quả này ta thấy mã hóa thứ tự nhanh gấp t_T lần so với mã hóa trực tiếp trên cùng một mạng định kỳ. Kết quả này sẽ được kiểm chứng ở Chương 4.

Thực nghiệm và kết quả

4.1 Mô hình bài toán PTSP về bài toán PESP

Vấn đề lập lịch tàu chạy (PTSP) trong thực tế còn phức tạp và có nhiều yếu tố tác động. Tuy nhiên, với sai số cho phép, ta có thể chuyển hoá các yêu cầu nghiệp vụ về sự kiện và các ràng buộc trong bài toán PESP.

Ứng với mỗi tàu L và ga s, tạo một sự kiện khởi hành và cập bến $L_{t,s}(t \in \{\text{dep}, \text{arr}\})$. Mỗi sự kiện khởi hành và cập bến phải tuân theo chu kì t_T. Giữa hai sự kiện này có một ràng buộc thời gian $\left(L_{\text{arr},s}, L_{\text{dep},s}, [l,u]_{t_T}\right)$, ràng buộc thời gian tối thiểu và tối đa tàu dừng tại ga (arrival $\rightarrow$ departure). Tương tự, ta ràng buộc thời gian đi từ gia $s_1 \rightarrow s_2$ bằng ràng buộc $\left(L_{\text{dep},s_1}, L_{\text{arr},s_2}, [l',u']_{t_T}\right)$. l', u' có thể ước lượng từ khoảng cách hai ga và vận tốc của tàu.

Để ngăn hai tàu sử dụng cùng một đường ray, ta giới hạn thời gian đến cùng 1 ga phải cách nhau một thời gian đệm tương đối: $\left(L_{\text{arr},s}, J_{\text{arr},s}, [l,u]_{t_T}\right)$. Tương tự với các yêu cầu ràng buộc khác.

Ta thu được thời gian biểu chính xác khi giải được bài toán PESP tương ứng.

4.2 Thu thập dữ liệu

Dữ liệu thử nghiệm được thu thập từ PESPlib [18], một tập dữ liệu PESP đã được chuẩn hóa và xử lý nhằm đánh giá hiệu quả của các thuật toán giải PESP. PESPlib được cộng đồng học thuật đánh giá cao và được dùng làm tiêu chuẩn đánh giá trong nhiều nghiên cứu.[19], [20]. Dữ liệu đầu vào gồm các file csv, có định dạng sau:

```
N; O; T; L; U; W
```

- N số thứ tự ràng buộc
- O sự kiện bắt đầu
- T sự kiện kết thúc

- L cận dưới của thời gian chuyển sự kiện
- U cận trên của thời gian chuyển sự kiện
- W là hệ số (độ quan trọng) của ràng buộc này

Lời giải của bài toán nên có định dạng sau:

```
N; D
```

- N là số thứ tự của sự kiện
- D là thời điểm sự kiện xảy ra

Ví dụ 4.2.1:
Input:
1; 1; 2; 50; 55; 10
2; 2; 3; 40; 50; 20
3; 1; 3; 30; 40; 15

Output:
1; 50
2; 40
3; 30

Toàn bộ dữ liệu đầu vào gồm 18 file với độ khó tăng dần, định dạng như mô tả ở trên. Thông qua tiền sử lý sơ bộ, ta có thông tin cơ bản của dữ liệu đầu vào như sau:

Instance	Events	Constraints	Period
R1L1	6.385	3.664	60
R1L2	6.543	3.668	60
R1L3	7.031	4.184	60
R1L4	8.528	4.760	60
R2L1	7.361	4.156	60
R2L2	7.563	4.204	60
R2L3	8.286	5.048	60
R2L4	13.173	7.660	60
R3L1	9.145	4.516	60
R3L2	9.251	4.452	60
R3L3	11.169	5.724	60
R3L4	15.657	8.180	60
R4L1	10.262	4.932	60
R4L2	10.735	5.048	60
R4L3	13.238	6.368	60
R4L4	17.754	8.384	60
BL1	7.985	2.688	60
BL2	7.485	2.606	60
BL3	9.308	3.044	60
BL4	13.499	3.816	60
R1L1v	6.495	3.664	60
R4L4v	18.020	8.384	60

Bảng 8: Bảng mô tả độ phức tạp của dữ liệu PESP đầu vào

4.3 Kết quả và đánh giá

Để tiến hành thử nghiệm hai phương pháp đã nêu ở Chương 3, khoá luận đã cài đặt một công cụ dòng lệnh giải bài toán PESP có tên là pesp-sat.

Chương trình thử nghiệm được cài đặt bằng ngôn ngữ Go, sử dụng SAT Solver Gini. Công cụ hỗ trợ đa nền tảng, được kiểm thử kĩ lưỡng, độ bao phủ đạt 80%, mã nguồn lưu tại: ppvan/pesp-sat. Tất cả tài liệu và dữ liệu liên quan, bao gồm mã nguồn công cụ thử nghiệm, tài liệu khóa luận và slide trình bày khóa luận được lưu trữ tại git repo này.

Để kiểm chứng chương trình thử nghiệm, vui lòng làm theo hướng dẫn trong README.md. Thực nghiệm sau đây được tiến hành trên máy tính (laptop) sau:

			Direct			Order		
Index	Events	Cons	Vars	Clauses	Time	Vars	Clauses	Time
R1L1	6.385	3.664	219.840	18.480.424	3.266	216.176	815.595	280
R1L2	6.543	3.668	220.080	18.489.668	3.101	216.412	825.065	291
R1L3	7.031	4.184	251.040	21.100.184	3.287	246.856	915.429	331
R1L4	8.528	4.760	285.600	23.997.680	3.813	280.840	1.073.150	446
R2L1	7.361	4.156	249.360	20.960.656	3.442	245.204	930.706	387
R2L2	7.563	4.204	252.240	21.195.364	3.815	248.036	948.427	383
R2L3	8.286	5.048	302.880	25.446.548	3.903	297.832	1.091.753	633
R2L4	13.173	7.660	459.600	38.653.720	7.447	451.940	1.693.148	561
R3L1	9.145	4.516	270.960	22.747.456	3.853	266.444	1.078.972	475
R3L2	9.251	4.452	267.120	22.427.892	3.809	262.668	1.077.765	452
R3L3	11.169	5.724	343.440	28.812.444	4.710	337.716	1.341.989	506
R3L4	15.657	8.180	490.800	41.269.220	7.485	482.620	1.902.987	702
R4L1	10.262	4.932	295.920	24.776.172	4.374	290.988	1.193.318	598
R4L2	10.735	5.048	302.880	25.350.248	4.448	297.832	1.235.149	494
R4L3	13.238	6.368	382.080	31.965.608	6.711	375.712	1.539.958	629
R4L4	17.754	8.384	503.040	42.205.124	7.387	494.656	2.048.635	638
BL1	7.985	2.688	161.280	13.713.408	2.663	158.592	776.655	327
BL2	7.485	2.606	156.360	13.270.826	2.683	153.754	737.430	317
BL3	9.308	3.044	182.640	15.536.744	3.083	179.596	895.294	395
BL4	13.499	3.816	228.960	19.608.456	23.416	225.144	1.232.514	437
R1L1v	6.495	3.664	219.840	18.480.424	3.055	216.176	822.085	393
R4L4v	18.020	8.384	503.040	42.205.124	8.286	494.656	2.064.329	681

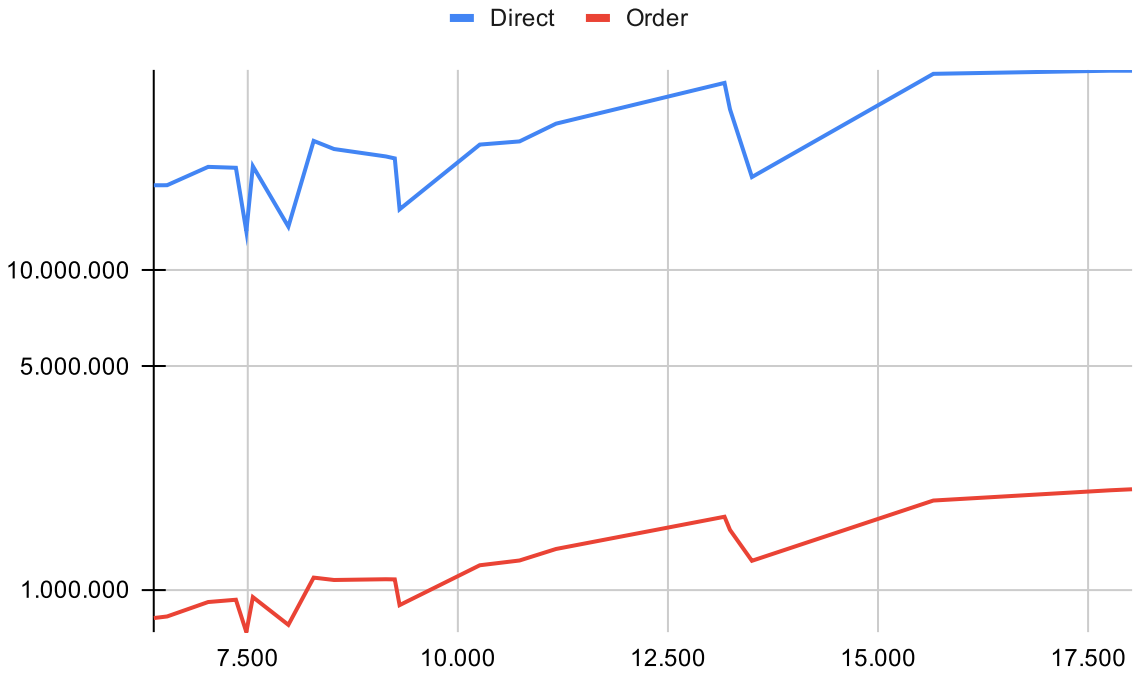
Bảng 10: Kết quả chạy thử nghiệm, thời gian tính bằng mili giây (ms)

Component	Details
CPU	AMD Ryzen™ 7 7735H
RAM	32GB DDR4
Disk	512GB SSD NVme
OS	Linux 6.6.51-1-lts
Gini(SAT Solver)	v1.0.4 - Go 1.23

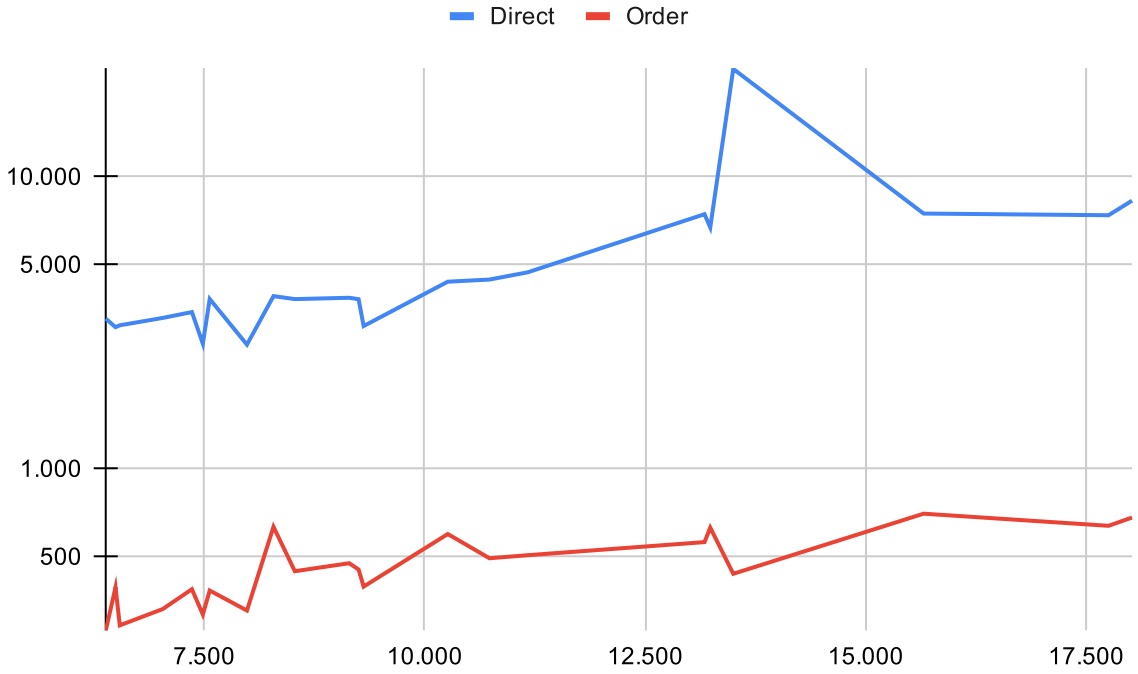
Bảng 9: Cấu hình máy chạy thực nghiệm

Khóa luận sẽ tiến hành đo thời gian chạy (ms), số mệnh đề, số biến của hai thuật toán mã hóa trình bày ở Chương 3, chi tiết trong Bảng 10. Mỗi ví dụ đều được tính trung bình 10 lần chạy để giảm sai số.

Hình 14: Biểu đồ đường so sánh số biến của Direct và Order Encoding



Hình 15: Biểu đồ đường so sánh số mệnh đề của Direct và Order Encoding



Hình 16: Biểu đồ đường so sánh thời gian thực thi của Direct và Order Encoding

Quan sát bảng dữ liệu và các biểu đồ trên, ta thấy cả hai thuật toán đều tăng độ phức tạp nhất quán với độ phức tạp tăng dần của vấn đề PESP đầu vào. Khoảng cách giữa Direct và Order Encoding là khá rõ rệt (khoảng 7x-50x về thời gian, 15x-20x về số mệnh đề). Tuy nhiên về số biến, hai phương pháp tương đối đồng đều. Như vậy, phương pháp mã hóa Order tỏ ra tương đối ưu việt so với Direct, điều này có thể dễ dàng giải thích bởi Order encoding loại bỏ không gian tìm kiếm theo từng vùng thay vì từng điểm như Direct, dẫn đến số mệnh đề ít hơn. Hơn nữa, theo mô tả ở Chương 3, các mệnh đề Order encoding chồng chéo lên nhau kiến vùng mâu thuẫn được tìm ra nhanh chóng bởi SAT Solver.

Với sức mạnh phần cứng hiện tại, cả hai phương pháp đều giải ra khá nhanh (từ 100ms đến 24s) dù số mệnh đề lên đến hàng chục triệu, do giới hạn của dữ liệu đầu vào, ta chưa thống kê được giới hạn của hai giải thuật. Mặt khác, bài toán PESP sinh ra khá nhiều nghiệm thỏa mãn, dẫn đến nhu cầu tìm ra nghiệm tối ưu (bài toán lập lịch tàu chạy tối ưu). Tuy nhiên, việc tìm ra các nhân tố đánh giá lịch trình đang gặp nhiều khó khăn, cần nghiên cứu thêm yêu cầu thực tế và cải thiện mô hình toán học [21], [22], không được trình bày đầy đủ trong khóa luận này. Đây là thiếu sót khóa luận chưa thể khắc phục, cần cải thiện trong tương lai.

Khoá luận đã trình bày nghiên cứu mới nhất về bài toán lập lịch định kì(PESP) và phương hướng tiếp cận bài toán sử dụng định nghĩa hình thức và các SAT Solver. Hai giải thuật mã hóa đã được cài đặt và thực nghiệm nhằm giải các bài toán PESP. Kết quả thực

nghiệm cho thấy phương pháp Order Encoding tỏ ra hiệu quả hơn nhiều so với phương pháp còn lại, thách thức nhiều giới hạn trong tương lai.

Quá trình nghiên cứu và thực nghiệm khóa luận đã giúp tôi có điều kiện tìm tòi, suy luận về bài toán lập lịch định kỳ cũng như phương pháp giải nó sử dụng kĩ thuật định nghĩa hình thức và SAT Solver. Khóa luận đã cho tôi những tri thức, trải nghiệm tuyệt vời khi nghiên cứu khoa học. Bên cạnh đó, tôi đã tiếp thu được nhiều bài học và phong cách làm việc, nghiên cứu khoa học từ thầy hướng dẫn.

Trên đây là toàn bộ nghiên cứu của tôi trong thời gian qua, tài liệu khó tránh khỏi sai sót, mong nhận được sự góp ý của các thầy cô và các bạn nghiên cứu về SAT, giúp tôi có thể hoàn thiện hơn nữa trong tương lai.

Tài liệu tham khảo

[1] L. Peeters, "Cyclic Railway Timetable Optimization", tr , 2003.

[2] C. Liebchen và R. H. Möhring, "The modeling power of the periodic event scheduling problem: railway timetables—and beyond", trong *Algorithmic Methods for Railway Optimization: International Dagstuhl Workshop, Dagstuhl Castle, Germany, June 20-25, 2004, 4th International Workshop, ATMOS 2004, Bergen, Norway, September 16-17, 2004, Revised Selected Papers*, 2007, tr 3–40.

[3] M. A. Odijk, "A constraint generation algorithm for the construction of periodic railway timetables", *Transportation Research Part B: Methodological*, vol 30, số p.h 6, tr 455–464, 1996.

[4] F. Yan, N. Bešinović, và R. M. Goverde, "Multi-objective periodic railway timetabling on dense heterogeneous railway corridors", *Transportation Research Part B: Methodological*, vol 125, tr 52–75, 2019.

[5] P. Serafini và W. Ukovich, "A Mathematical Model for Periodic Scheduling Problems", *SIAM Journal on Discrete Mathematics*, vol 2, số p.h 4, tr 550–581, 1989, doi: 10.1137/0402049.

[6] D. C. Kozen, *Automata and computability*. Springer Science & Business Media, 2012.

[7] M. A. Odijk, *Construction of Periodic Timetables. Pt. 1. A Cutting Plane Algorithm*. TU Delft, 1994.

[8] B. Pfahringer, "Conjunctive Normal Form", *Encyclopedia of Machine Learning*. Springer, tr 209–210, 2010. doi: 10.1007/978-0-387-30164-8_158.

[9] S. A. Cook, "The complexity of theorem-proving procedures", trong *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, trong STOC '71. Shaker Heights, Ohio, USA: Association for Computing Machinery, 1971, tr 151–158. doi: 10.1145/800157.805047.

[10] M. Davis và H. Putnam, "A Computing Procedure for Quantification Theory", *J. ACM*, vol 7, số p.h 3, tr 201–215, tháng 7 1960, doi: 10.1145/321033.321034.

[11] M. Davis, G. Logemann, và D. Loveland, "A machine program for theorem-proving", *Commun. ACM*, vol 5, số p.h 7, tr 394–397, tháng 7 1962, doi: 10.1145/368273.368557.

[12] J. Marques Silva và K. Sakallah, "GRASP-A new search algorithm for satisfiability", trong *Proceedings of International Conference on Computer Aided Design*, 1996, tr 220–227. doi: 10.1109/ICCAD.1996.569607.

[13] T. Balyo, P. Sanders, và C. Sinz, "HordeSat: A Massively Parallel Portfolio SAT Solver". [Online]. Available at: https://arxiv.org/abs/1505.03340

[14] R. Martins, V. Manquinho, và I. Lynce, "An overview of parallel SAT solving", *Constraints*, vol 17, tr 304–347, 2012.

[15] Y. Hamadi, S. Jabbour, và L. Sais, "ManySAT: a parallel SAT solver", *Journal on Satisfiability, Boolean Modeling and Computation*, vol 6, số p.h 4, tr 245–262, 2010.

[16] T. Tanjo, N. Tamura, và M. Banbara, "A Compact and Efficient SAT-Encoding of Finite Domain CSP", 2011, tr 375–376. doi: 10.1007/978-3-642-21581-0_36.

[17] J. Chen, "A new SAT encoding of the at-most-one constraint", *Proc. constraint modelling and reformulation*, tr 8, 2010.

[18] M. Goerigk, M. Schachtebeck, và A. Schöbel, "Evaluating line concepts using travel times and robustness: Simulations with the LinTim toolbox", *Public Transport*, vol 5, tr 267–284, 2013, doi: 10.1007/s12469-013-0072-x.

[19] M. Goerigk và A. Schöbel, "An empirical analysis of robustness concepts for timetabling", *Erlebach*, vol 14, tr 100–113, 2010.

[20] J.-W. Goossens, "Models and algorithms for railway line planning problems", tr , 2004.

[21] G. Caimi, M. Fuchsberger, M. Laumanns, và K. Schüpbach, "Periodic railway timetabling with event flexibility", *Networks*, vol 57, số p.h 1, tr 3–18, 2011, doi: https://doi.org/10.1002/net.20379.

[22] F. Yan, N. Bešinović, và R. M. Goverde, "Multi-objective periodic railway timetabling on dense heterogeneous railway corridors", *Transportation Research Part B: Methodological*, vol 125, tr 52–75, 2019, doi: https://doi.org/10.1016/j.trb.2019.05.002.