ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - CƠ - TIN HỌC

Nguyễn Thị Thanh Hương

Xây dựng các trường hợp khó với kích thước tùy chỉnh cho bài toán tái cấu hình các tập độc lập trong đồ thị

Khóa luận tốt nghiệp đại học hệ chính quy
Ngành Toán Tin
(Chương trình đào tạo chuẩn)

Người hướng dẫn: TS. Hoàng Anh Đức

Hà Nội - 2025

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - CƠ - TIN HỌC

Nguyễn Thị Thanh Hương

Xây dựng các trường hợp khó với kích thước tùy chỉnh cho bài toán tái cấu hình các tập độc lập

Khóa luận tốt nghiệp đại học hệ chính quy
Ngành Toán Tin
(Chương trình đào tạo chuẩn)

Cán bộ hướng dẫn: TS. Hoàng Anh Đức

Hà Nội - 2025

LỜI CẢM ƠN

Trước hết, em xin gửi lời tri ân sâu sắc đến TS. Hoàng Anh Đức, người đã tận tâm hướng dẫn và kiên nhẫn chỉ bảo em trong suốt quá trình thực hiện khóa luận tốt nghiệp.

Bên cạnh đó, em cũng xin bày tỏ lòng biết ơn đến các thầy cô giáo, cán bộ, nhân viên trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Hà Nội nói chung, cũng như các thầy cô giáo khoa Toán - Cơ - Tin học nói riêng, đã truyền đạt cho em những kiến thức quý giá và luôn tạo điều kiện thuận lợi để em có thể học tập trong một môi trường tốt nhất. Đồng thời, em xin gửi lời cảm ơn đến những người bạn đã đồng hành, động viên, chia sẻ cùng em trong suốt quá trình học tập.

Đặc biệt, em xin dành những tình cảm trân quý nhất đến gia đình, một chỗ dựa vững chắc, nguồn động viên lớn lao giúp em kiên trì và vững bước trên con đường học tập.

Hà Nội, ngày 19 tháng 05 năm 2025

Sinh viên

Nguyễn Thị Thanh Hương

BẢNG KÝ HIỆU

STT	Tên viết tắt	Tên đầy đủ	Ý nghĩa
1	BMC	Bounded Model Checking	Kiểm thử mô hình có giới hạn
2	LCR	List Coloring Reconfiguration	Tái cấu hình danh sách tô màu đồ thị
3	SAT	Boolean Satisfiability Problem	Bài toán thỏa mãn logic mệnh đề
4	SPR	Shortest Path Reconfiguration	Tái cấu hình đường đi ngắn nhất

Danh sách hình vẽ

Danh sách bảng

Mục lục

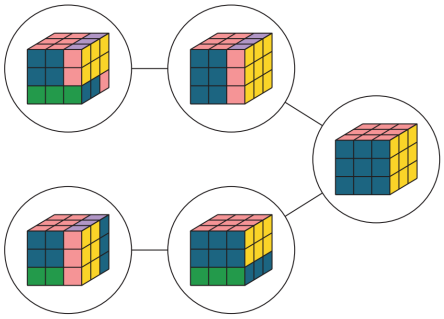
LỜI MỞ ĐẦU

Trong một bài toán tổ hợp, có khi ta không chỉ quan tâm đến việc tìm ra một lời giải, mà còn muốn biết cách biến đổi từ một lời giải này sang một lời giải khác thông qua một chuỗi các bước tuân theo quy tắc. Đây chính là cốt lõi của bài toán tái cấu hình (reconfiguration problem). Bài toán này có thể được tiếp cận theo hai cách:

Góc nhìn quá trình: Một chuỗi các bước chuyển đổi giữa các trạng thái, trong đó mỗi bước đều tuân theo một tập hợp quy tắc nhất định.

Góc nhìn đồ thị: Không gian các trạng thái hợp lệ được biểu diễn dưới dạng một đồ thị, trong đó mỗi đỉnh là một trạng thái hợp lệ, và có một cạnh nối giữa hai đỉnh nếu có thể chuyển đổi trực tiếp giữa chúng bằng một bước chuyển tuân theo quy tắc.

Một ví dụ tiêu biểu cho bài toán tái cấu hình là bài toán giải khối rubik (Hình 1)



Hình 1: Ví dụ một phần đồ thị tái cấu hình bài toán giải Rubik

Ở đây, mỗi trạng thái của khối Rubik tương ứng với một cách sắp xếp các màu trên sáu mặt của khối. Các bước chuyển hợp lệ là các phép xoay mặt của Rubik, mỗi lần xoay sẽ tạo ra một trạng thái mới. Mục tiêu của người chơi là tìm một chuỗi các bước xoay sao cho từ một trạng thái ban đầu có thể đạt đến trạng thái hoàn chỉnh - mỗi mặt của khối rubik cùng màu. Nếu xét theo góc nhìn đồ thị, mỗi trạng thái của Rubik là một đỉnh và có một cạnh nối hai trạng thái nếu có thể chuyển đổi giữa chúng bằng một phép xoay duy nhất. Trong lý thuyết đồ thị, một trong những bài toán tái cấu hình quan trọng là bài toán Token Jumping. Trong bài toán này, ta xem xét một đồ thị và hai tập độc lập có cùng kích thước.

Bài toán Token Jumping đã được nghiên cứu rộng rãi trong lý thuyết thuật toán, nhưng việc nghiên cứu thực nghiệm vẫn còn hạn chế. Để đánh giá hiệu suất của các

thuật toán giải bài toán này, ta cần có bộ dữ liệu kiểm thử (benchmark datasets) phù hợp, tức là tập hợp các trường hợp đầu vào có hai đặc điểm: Có thể mở rộng (Scalable), tức là có thể điều chỉnh kích thước của bài toán bằng cách thay đổi số đỉnh của đồ thị, trong khi vẫn giữ nguyên các đặc trưng quan trọng. Khó giải quyết (Hard), tức là số bước biến đổi được tính theo cấp số mũ tương ứng với kích thước của đồ thị.

Khóa luận này tập trung tìm hiểu và trình bày ba dạng trường hợp đầu vào cho bài toán Token Jumping, được đề xuất trong bài báo [1], đồng thời sử dụng thuật toán SAT để tiến hành thực nghiệm sử dụng dữ liệu đầu vào là ba trường hợp đã trình bày.

Bố cục của khóa luận bao gồm 4 chương:

- Chương 1 của khóa luận trình bày tóm tắt một số định nghĩa trong đồ thị, bài toán tái cấu hình và bài toán Token Jumping.
- Chương 2 của khóa luận tập trung tìm hiểu và trình bày ba trường hợp đầu vào cho bài toán Token Jumping.
- Chương 3 trình bày thuật toán được sử dụng để tiến hành thực nghiệm cho ba trường hợp đầu vào.
- Chương 4 thực nghiệm và kết quả.

Do kiến thức và kinh nghiệm còn hạn chế, bài khóa luận chắc chắn không thể tránh khỏi những thiếu sót. Em rất mong nhận được sự thông cảm và sẵn lòng đón nhận những nhận xét, góp ý từ quý thầy cô trong Hội đồng để có thể hoàn thiện hơn.

Xin chân thành cảm ơn!

Hà Nội, ngày 19 tháng 05 năm 2025

Sinh viên

Nguyễn Thị Thanh Hương

Chương 1

Kiến thức chuẩn bị

1.1. Đồ thị

Định nghĩa 1.1.1. *Một **đồ thị vô hướng (undirected graphs)** G được định nghĩa là một bộ đôi (V, E), trong đó:*

- *V là tập đỉnh của G.*
- *E là tập cạnh của G, với mỗi cạnh là một cặp không có thứ tự $\{u, v\}$, trong đó $u, v \in V$.*

Định nghĩa 1.1.2. ***Một đồ thị con (subgraph)** H của G là đồ thị có tập đỉnh $V(H) \subseteq V$ và tập cạnh $E(H) \subseteq E$.*

Tập độc lập (independent set)

Định nghĩa 1.1.3. *Một tập $S \subseteq V$ được gọi là một **tập độc lập** của đồ thị G nếu không có cạnh nào nối giữa hai đỉnh bất kỳ trong tập S, tức là $uv \notin E$ với mọi $u, v \in S$.*

Tập độc lập cực đại (**maximum independent set**).

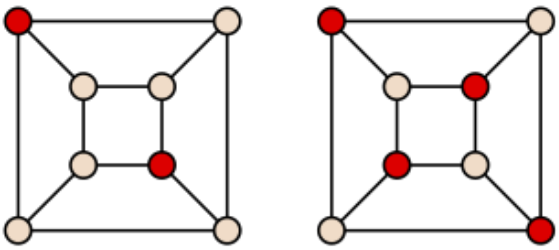
Đây là bài toán quyết định tìm một tập độc lập có lực lượng lớn nhất trong đồ thị đã cho. Bài toán quyết định tương ứng được phát biểu như sau:

Input: Một đồ thị G và một số nguyên k.

Output: TRUE nếu và chỉ nếu tồn tại một tập độc lập của G có lực lượng ít nhất k.

Định nghĩa 1.1.4. ***Mức độc lập (independence number)*** *của đồ thị G, ký hiệu $\alpha(G)$, là kích thước của một tập độc lập lớn nhất trong G.*

Khái niệm về tập độc lập và mức độc lập được minh họa trong Hình (1.1)



Hình 1.1: Ví dụ về đồ thị G và hai tập độc lập của G. Tập độc lập bên phải là tập độc lập cực đại, $\alpha(G) = 4$

Tô màu đồ thị (graph coloring):

Một tô màu hợp lệ của đồ thị G là một cách gán màu cho mỗi đỉnh của G sao cho không có hai đỉnh kề nhau có cùng màu. Nói cách khác, tô màu đồ thị chính là một phép phân hoạch tập đỉnh V thành các tập độc lập, trong đó mỗi tập độc lập tương ứng với một màu.

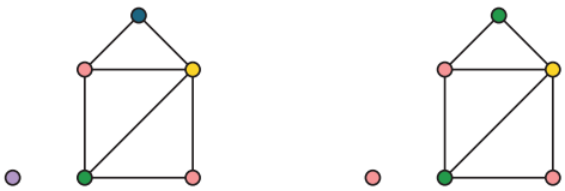
Bài toán quyết định tương ứng được phát biểu như sau:

Input: Một đồ thị G và một số nguyên k.

Output: TRUE nếu và chỉ nếu tồn tại một tô màu hợp lệ của G sử dụng không quá k màu.

Định nghĩa 1.1.5. *Số lượng màu ít nhất trong một tô màu hợp lệ của G được gọi là* ***sắc số (chromatic number)*** *của đồ thị, ký hiệu $\chi(G)$.*

Khái niệm này được minh họa trong Hình (1.2).



Hình 1.2: Ví dụ về đồ thị G và hai cách tô màu thích hợp của G. Đồ thị bên phải sử dụng số màu tối thiểu, $\chi(G) = 3$.

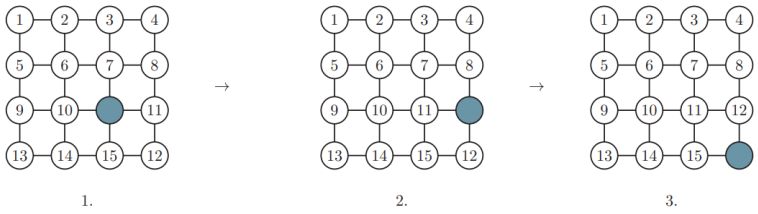
1.2. Bài toán Token Jumping

Trong phần này định nghĩa cơ bản về một bài toán tái cấu hình và đồ thị tái cấu hình, minh họa các định nghĩa này thông qua ví dụ về bài toán 15-puzzle.

1.2.1. Giới thiệu bài toán tái cấu hình (reconfiguration problem)

Một ví dụ kinh điển cho bài toán tái cấu hình là trò chơi 15-puzzle. Trò chơi này gồm một bảng có kích thước 4x4 với các ô được đánh số từ 1 đến 15 và một ô trống. Ở trạng thái bắt đầu, các ô được sắp đặt ngẫu nhiên, và nhiệm vụ của người chơi là tìm cách đưa các ô về đúng thứ tự lớn dần từ trái qua phải và từ trên xuống dưới.

Trong mô hình toán học của trò chơi 15-puzzle, ta xét một tập hợp gồm 15 quân cờ, được đánh số từ 1 đến 15, đặt chúng trên các đỉnh của đồ thị lưới 4x4. Đỉnh không chứa quân cờ nào là đỉnh trống. Mỗi nước đi là việc trượt một quân cờ dọc theo một cạnh từ đỉnh hiện tại tới đỉnh trống. Mục đích của người chơi là thực hiện một dãy các nước đi từ cấu hình ban đầu đến cấu hình đích (quân cờ có nhãn i nằm trên đỉnh v_i), được minh họa trong Hình (1.3).



Hình 1.3: Một cách giải trò chơi 15-puzzle qua 2 bước

Một bài toán tái cấu hình được xác định bởi hai yếu tố chính: các cấu hình (configuration) cần xét và quy tắc chuyển từ cấu hình này sang cấu hình khác (reconfiguration rule).

Dưới đây sẽ định nghĩa các khái niệm này và sử dụng bài toán 15-puzzle để minh họa.

Định nghĩa 1.2.1. *Một cấu hình là một nghiệm khả khi (giải pháp hợp lệ) của một bài toán.*

Ví dụ 1.2.1. Trong bài toán 15-puzzle, một cấu hình là một cách sắp xếp các quân cờ trên đồ thị lưới 4×4, các quân cờ được đánh số từ 1 đến 15.

Định nghĩa 1.2.2. ***Quy tắc tái cấu hình*** *là các quy tắc mô tả phép biến đổi hợp lệ giữa hai cấu hình trong một bước duy nhất* ***bước chuyển hợp lệ (allowed rules)****.*

Ví dụ 1.2.2. Trong trò chơi 15-puzzle, bước chuyển hợp lệ là việc trượt một quân cờ dọc theo một cạnh trên đồ thị lưới từ đỉnh hiện tại đến đỉnh trống. Việc giải trò chơi 15-puzzle là một dãy các bước chuyển.

1.2.2. Đồ thị tái cấu hình

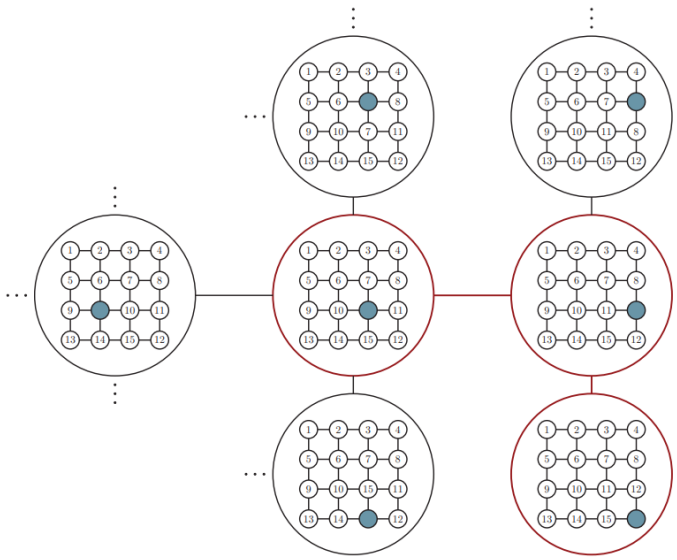
Định nghĩa 1.2.3. *Cho bài toán nguồn P với hai nghiệm khả thi X_s và X_t. Một* ***dãy tái cấu hình (reconfiguration sequence)*** *từ X_S đến X_t là một dãy $S = \langle X_1 := X_s, \ldots, X_\ell := X_t \rangle$ sao cho:*

- *Mỗi $X_i \in S$ là một nghiệm khả thi của P*
- *Với mọi cặp nghiệm khả thi liên tiếp X_r và X_{r+1} của S tồn tại một bước chuyển hợp lệ từ X_r đến X_{r+1}*

Như vậy, việc giải trò chơi 15-puzzle tương đương với việc tìm một dãy tái cấu hình từ trạng thái đầu tiên tới trạng thái đích của các quân cờ.

Định nghĩa 1.2.4. *Một dãy tái cấu hình từ X_s đến X_t là một đường đi trong* ***đồ thị tái cấu hình (reconfiguration graph)****, ký hiệu R. Các đỉnh của đồ thị R biểu diễn các nghiệm khả thi. Hai nghiệm nằm trên hai đỉnh kề nhau trên R nếu có một bước chuyển duy nhất từ nghiệm này sang nghiệm kia.*

Một phần đồ thị tái cấu hình cho bài toán 15-puzzle được biểu diễn trong Hình (1.4)



Hình 1.4: Một phần đồ thị tái cấu hình cho bài toán 15-puzzle. Cấu hình và các cạnh tô đỏ là ví dụ cho một dãy tái cấu hình

1.2.3. Bài toán Token Jumping

Tái cấu hình tập độc lập:

Bài toán tái cấu hình tập độc lập được xét trên một đồ thị và hai tập độc lập của đồ thị đã cho có cùng kích thước. Mục đích của bài toán là xác định xem tập độc lập này có thể chuyển đổi thành tập độc lập khác thông qua chuỗi các tập độc lập trung gian được không?

Định nghĩa 1.2.5. *Giả sử $G = (V, E)$ là một đồ thị vô hướng không trọng số. Một tập con đỉnh $I \subseteq V$ là tập độc lập của G.*

Với một số nguyên dương k, không gian nghiệm $S_k(G)$ được định nghĩa là một đồ thị có:

- *Mỗi đỉnh trong $S_k(G)$ tương ứng với một tập độc lập I của G sao cho $|I| = k$.*
- *Hai đỉnh trong $S_k(G)$ được nối với nhau bởi một cạnh nếu và chỉ nếu hai tập độc lập tương ứng I và I' thỏa mãn $|I \setminus I'| = |I' \setminus I| = 1$, tức là chúng chỉ khác nhau đúng một phần tử.*

*Một đường đi trong $S_k(G)$ nối hai đỉnh I và I' được gọi là **dãy tái cấu hình (reconfiguration sequence)** giữa I và I'. **Độ dài (length)** của dãy tái cấu hình*

được định nghĩa là số cạnh trong đường đi đó.

Định nghĩa 1.2.6. *Cho một đồ thị G và hai tập độc lập I_s và I_t của G sao cho $|I_s| = |I_t| = k$, bài toán **Token Jumping** yêu cầu xác định liệu có tồn tại một dãy tái cấu hình giữa I_s và I_t trong $S_k(G)$ hay không.*

Định nghĩa 1.2.7. *Một bộ ba (G, I_s, I_t) sẽ được dùng để biểu diễn một trường hợp của bài toán **Token Jumping**.*

*Với một bộ dữ liệu (G, I_s, I_t), **khoảng cách (distance)** của nó được xác định là độ dài ngắn nhất của bất kỳ dãy tái cấu hình nào giữa I_s và I_t.*

Nếu không tồn tại dãy tái cấu hình giữa I_s và I_t, khoảng cách được gán giá trị $+\infty$.

*Một bộ dữ liệu của bài toán **Token Jumping** được coi là khó giải quyết nếu khoảng cách của nó là hàm mũ theo kích thước đầu vào.*

Chương 2

Các trường hợp khó có kích thước tùy chỉnh

2.1. Trường hợp đầu vào xây dựng dựa trên bài toán tái cấu hình SAT (SAT Reconfiguration)

2.1.1. Bài toán tái cấu hình SAT

Định nghĩa 2.1.1. *Một **công thức CNF** là một công thức mệnh đề có dạng $C_1 \wedge C_2 \wedge \cdots \wedge C_n$ trong đó mỗi C_i là một mệnh đề (clause), mỗi mệnh đề là phép tuyển được lập nên từ các biến và phủ định của các biến.*

*Nếu k là một số nguyên dương, thì một công thức k**-CNF** là một công thức CNF trong đó mỗi mệnh đề C_i là một phép hoặc của tối đa k biến và phủ định các biến.*

Ví dụ 2.1.1. Công thức mệnh đề dưới đây là một ví dụ về công thức CNF với ba mệnh đề:

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

Ở đây: $C_1 = (x_1 \vee \neg x_2)$, $C_2 = (\neg x_1 \vee x_3)$, $C_3 = (x_2 \vee x_3)$

Công thức mệnh đề sau là một công thức 3-CNF:

$$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_4) \wedge (x_2 \vee x_3 \vee x_4)$$

Định nghĩa 2.1.2. *Một* ***quan hệ logic R*** *là một tập con khác rỗng của* $\{0,1\}^k$*, với* $k \geq 1$*, trong đó* k *là bậc (arity) của* R

Ví dụ 2.1.2. Nếu $k = 2$, một quan hệ logic R có thể là một tập con rỗng của $\{0,1\}^2 = \{(0,0),(0,1),(1,0),(1,1)\}$.

Định nghĩa 2.1.3. *Cho một tập hữu hạn các quan hệ logic* S*, một* ***công thức CNF(S)*** *trên tập biến* $V = \{x_1, \ldots, x_n\}$ *là một công thức có dạng:* $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_n$*, trong đó mỗi mệnh đề* C_i *được xây dựng theo các quan hệ từ tập S, các biến từ tập V, và các hằng số 0 và 1; tức là, mõi mệnh đề* C_1 *là một biểu thức có dạng* $R(\xi_1, \ldots, \xi_k)$*, trong đó:*

- $R \in S$ *là một quan hệ logic có bậc* k*,*
- *Mỗi* ξ_j *là một biến trong* V *hoặc là các hằng số 0 hoặc 1.*

Ví dụ 2.1.3. Giả sử ta có:

- Tập biến: $V = \{x_1, x_2, x_3\}$
- Tập quan hệ logic: $S = \{R_1\}$, trong đó: $R_1 = \{(0,1),(1,0),(1,1)\}$

Công thức CNF(S):

$$\varphi = R_1(x_1, x_2) \wedge R_1(x_2, x_3) \wedge R_1(x_2, 0)$$

Định nghĩa 2.1.4. ***Một nghiệm (solution) (hay phép gán thỏa mãn)*** *của công thức* CNF(S) φ *là một phép gán* $s = (a_1, \ldots, a_n) \in \{0,1\}^n$ *sao cho mọi mệnh đề trong* φ *đều đúng.*

Công thức φ *được gọi là thỏa mãn (satisfiable) nếu tồn tại ít nhất một nghiệm như vậy.*

Ví dụ 2.1.4. Xét công thức CNF(S):

$$\varphi = R_1(x_1, x_2) \wedge R_1(x_2, x_3) \wedge R_1(x_2, 0)$$

Một phép gán thỏa mãn công thức là $s = (x_1, x_2, x_3) = (1,1,0)$, trong đó:

- $R_1(1,1) = (1,1) \in R_1$

- $R_1(1,0) = (1,0) \in R_1$
- $R_1(1,0) = (1,0) \in R_1$

Bài toán SAT Reconfiguration:

Định nghĩa 2.1.5. *Cho một công thức mệnh đề φ thuộc một lớp công thức CNF(S), và hai phép gán thỏa mãn s và t của φ.* ***Bài toán SAT Reconfiguration*** *yêu cầu xác định liệu có tồn tại một dãy các phép gán $s = s_0, s_1, \ldots, s_\ell = t$ sao cho:*

- *Mỗi phép gán s_i (với $0 \leq i \leq \ell$) đều là một nghiệm (phép gán thỏa mãn) của φ;*
- *Mỗi cặp liên tiếp s_i và s_{i+1} chỉ khác nhau đúng một biến (tức là s_{i+1} thu được từ s_i bằng cách lật giá trị của đúng một biến).*

Như vậy, nói cách khác, bài toán tái cấu hình SAT yêu cầu kiểm tra xem có thể biến đổi phép gán s thành phép gán t thông qua một chuỗi các phép gán trung gian, mỗi lần chỉ thay đổi giá trị một biến, và mọi phép gán trung gian vẫn thỏa mãn công thức φ.

Ví dụ 2.1.5. Xét công thức CNF:

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

Với các phép gán thỏa mãn:

$$s = (1,1,1), \quad t = (1,0,0)$$

Dãy các phép gán có thể là:

$$s = s_0 = (1,1,1) \to s_1 = (1,1,0) \to s_2 = (1,0,0) = t$$

Ở đây, mỗi phép gán chỉ thay đổi giá trị tại một biến, và mọi phép gán trong dãy đều thỏa mãn công thức CNF.

2.1.2. Phép giảm từ bài toán tái cấu hình SAT sang bài toán Token Jumping

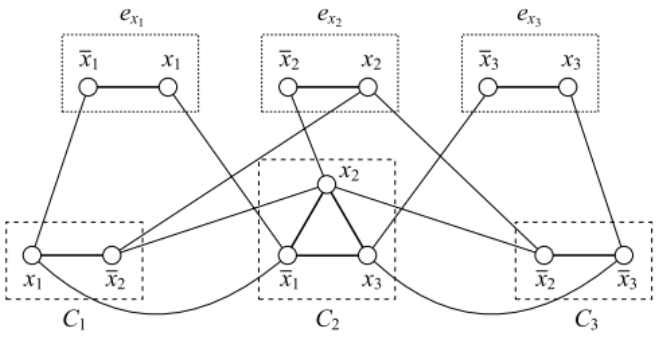
Định lý 2.1.1. *Cho φ là một công thức 3-CNF với N biến và M mệnh đề, được xem là một trường hợp của bài toán tái cấu hình SAT. Khi đó, tồn tại một trường hợp tương ứng của bài toán Token Jumping sao cho kích thước đầu vào là $O(NM)$ và khoảng cách (số bước tái cấu hình) ít nhất bằng khoảng cách của trường hợp ban đầu.*

Chứng minh. **Xây dựng phép giảm từ bài toán tái cấu hình SAT sang bài toán Token Jumping**:

Phép giảm này đã được Ito và cộng sự trình bày [24]. Cụ thể, ta xây dựng một đồ thị $\rho(\varphi)$ từ một công thức 3-CNF φ với N biến $(x_1, x_2, ..., x_N)$ và M mệnh đề $(C_1, C_2, ..., C_M)$ như sau: (Để không mất tính tổng quát, giả sử công thức φ không có mệnh đề nào chứa cả biến và phủ định của biến đó)

- Bước 1: Với mỗi biến x_i trong φ, ta thêm một cạnh e_x vào đồ thị; hai đầu mút của nó được gán nhãn x_i và $\overline{x_i}$.
- Bước 2: Với mỗi mệnh đề C_j trong φ, ta thêm một cụm có kích thước $|C_j|$; mỗi đỉnh trong cụm tương ứng với một biến hoặc phủ định của một biến trong mệnh đề C_j.
- Bước 3: Thêm một cạnh giữa một biến x_i xuất hiện trong một mệnh đề với biến $\overline{x_i}$ trong mệnh đề khác.

Hình 2.1 dưới đây là minh họa cho cách xây dựng đồ thị $\rho(\varphi)$:



Hình 2.1: Đồ thị $\rho(\varphi)$ tương ứng với công thức 3-CNF φ gồm ba mệnh đề $C_1 = (x_1 \vee \bar{x}_2)$, $C_2 = (\bar{x}_1 \vee x_2 \vee x_3)$ và $C_3 = (\bar{x}_2 \vee \bar{x}_3)$.

Đồ thị $\rho(\varphi)$ thu được có:

Số đỉnh (V):

Theo bước 1: Có $2N$ đỉnh.

Theo bước 2: Có nhiều nhất $3M$ đỉnh

Tổng số đỉnh: $O(2N + 3M) = O(N + M)$

Số cạnh (E):

Theo bước 1: Có N cạnh.

Theo bước 2: Có nhiều nhất $N.M$ cạnh

Tổng số cạnh: $O(N + NM) = O(NM)$

Như vậy, đồ thị $\rho(\varphi)$ có kích thước $O(NM)$

Xét tất cả các tập độc lập có kích thước k trong $\rho(\varphi)$; ta có thể phân chúng thành các lớp con dạng $\rho(s)$ tương ứng với các phép gán thỏa mãn s của ϕ. Mỗi tập độc lập trong cùng một lớp $\rho(s)$ tương ứng với một cách chọn các biến hoặc phủ định của biến đúng để thỏa mãn từng mệnh đề. Tất cả các tập trong $\rho(s)$ liên thông qua các tập độc lập trung gian có kích thước ít nhất $k - 1$.

Do đó, có thể biến đổi giữa hai tập độc lập có kích thước k trong $\rho(\varphi)$ thông qua các tập độc lập trung gian có kích thước ít nhất $k - 1$. Như vậy, số bước tái cấu hình trong Token Jumping ít nhất bằng số bước trong bài toán tái cấu hình SAT. □

2.1.3. Xây dựng trường hợp đầu vào SAT

Bổ đề 2.1.1. *Với số chẵn n, tồn tại một công thức 3-CNF φ_n với n biến và $\mathcal{O}(n^2)$ mệnh đề, sao cho $G(\varphi_n)$ là một đường đi có độ dài lớn hơn $2^{\frac{n}{2}}$.*

Chứng minh. Quá trình xây dựng công thức 3-CNF này gồm hai bước:

- Đầu tiên, ta xây dựng một đồ thị con G_n của siêu lập phương n chiều có đường kính lớn.
- Sau đó, ta xây dựng công thức 3-CNF φ_n sao cho $G_n = G(\varphi_n)$.

Đồ thị G_n là một đường đi có độ dài 2^{2^n}. Ta xây dựng nó bằng phương pháp quy nạp:

Với $n = 2$, chọn $V(G_2) = \{(0, 0), (0, 1), (1, 1)\}$ có đường kính 2.

Giả sử đã xây dựng được G_{n-2} với $2^{2^{n-2}}$ đỉnh, cùng hai đỉnh đặc biệt s_{n-2}, t_{n-2} sao cho đường đi ngắn nhất từ s đến t trong G_{n-2} có độ dài $2^{2^{n-2}}$.

Ta mô tả tập đỉnh $V(G_n)$ như sau:

- Với mỗi đỉnh $v \in V(G_{n-2})$, tập $V(G_n)$ chứa hai đỉnh $(v, 0, 0)$ và $(v, 1, 1)$.
- Đồ thị con tạo bởi các đỉnh này gồm hai bản sao rời rạc của G_{n-2}.
- Để kết nối hai thành phần này, ta thêm đỉnh $m = (t, 0, 1)$ (nối với $(t, 0, 0)$ và $(t, 1, 1)$).

Đồ thị G_n thu được liên thông, nhưng mọi đường đi từ $(u, 0, 0)$ đến $(v, 1, 1)$ đều phải đi qua m. Theo giả thiết quy nạp, G_n cũng là một đường đi. Hai đỉnh $s_n = (s_{n-2}, 0, 0)$ và $t_n = (s_{n-2}, 1, 1)$ là điểm đầu và điểm cuối của đường đi này.

Độ dài đường đi ngắn nhất là $2.2^{\frac{n-2}{2}} + 2 > 2^{\frac{n}{2s}}$.

Xây dựng dãy công thức 3-CNF $\varphi_n(x_1, \ldots, x_n)$ sao cho $G_n = G(\varphi_n)$:

Với $\varphi_2(x_1, x_2) = \neg x_1 \vee x_2$.

Giả sử $\varphi_{n-2}(x_1, \ldots, x_{n-2})$. Thêm hai biến x_{n-1}, x_n và các mệnh đề:

$$\varphi_{n-2}(x_1, \ldots, x_{n-2}), \quad \neg x_{n-1} \wedge x_n$$

$$x_{n-1} \vee \neg x_n \vee \neg x_i \quad \text{với } i \leq n-4 \tag{2.1.1}$$

$$x_{n-1} \vee \neg x_n \vee x_i \quad \text{với } i = n-3, n-2 \tag{2.1.2}$$

Mệnh đề 2.1.1 là một mệnh đề kéo theo

$$(\neg x_{n-1} \wedge x_n) \to \neg x_i \quad \text{với } i \leq n-4$$

. Các mệnh đề 2.1.1 và 2.1.2 đảm bảo rằng nếu $x_{n-1} = 0$ và $x_n = 1$, thì $(x_1, \ldots, x_{n-2}) = t_{n-2} = (0, \ldots, 0, 1, 1)$. □

Bổ đề 2.1.1 đã xây dựng các trường hợp bài toán tái cấu hình SAT sao cho các công thức đều là 3-CNF và khoảng cách giữa các trường hợp trong chuỗi tăng theo hàm mũ theo kích thước đầu vào. Bằng cách lấy các trường hợp tái cấu hình SAT này, Định lý 2.1.1 cho ta các trường hợp khó cho bài toán Token Jumping.

2.2. Trường hợp đầu vào xây dựng dựa trên bài toán tái cấu hình đường di ngắn nhất (Shortest Path Reconfiguration)

2.2.1. Bài toán tái cấu hình đường đi ngắn nhất

Định nghĩa 2.2.1. *Cho đồ thị vô hướng, không trọng số $G = (V, E)$ với hai đỉnh s và t. Giả sử P và Q là hai đường đi ngắn nhất khác nhau từ s đến t. Bài toán* ***Tái cấu hình đường đi ngắn nhất (SPR)*** *yêu cầu xác định xem có tồn tại một dãy chuyển đổi hợp lệ giữa P và Q thoả mãn các điều kiện sau:*

- *Tồn tại một dãy các đường đi ngắn nhất $P_0, P_1, \ldots, P_k$ sao cho:*
  - *$P_0 = P$ và $P_k = Q$,*
  - *Với mọi $i \in \{0, \ldots, k-1\}$, đường đi P_i và P_{i+1} chỉ khác nhau tại đúng một đỉnh (khác s và t).*
- *Mỗi P_i phải là một đường đi ngắn nhất từ s đến t trong đồ thị G.*

Như vậy, theo cách nói đơn giản, khi ta được cho hai đường đi ngắn nhất từ đỉnh s đến đỉnh t trong một đồ thị không trọng số, bài toán SPR là việc xác định liệu có thể biến đổi một đường đi thành đường còn lại bằng cách thay thế một đỉnh duy nhất trong mỗi bước, sao cho mọi đường đi trung gian đều là đường đi ngắn nhất từ đỉnh s đến đỉnh t hay không.

2.2.2. Phép giảm từ bài toán SPR sang bài toán Token Jumping

Kamiński và cộng sự [24] đã chỉ ra phép giảm từ bài toán *tái cấu hình đường đi ngắn nhất*, trong đó hai đường đi ngắn nhất (s, t) được xem là kề nhau trong đồ thị tái cấu hình khi và chỉ khi chúng khác nhau đúng một đỉnh khi biểu diễn dưới dạng dãy.

Giả sử ta có đồ thị G với hai đỉnh $s, t \in V(G)$. Gọi $\widetilde{G}$ là đồ thị thu được từ G bằng cách xóa tất cả các đỉnh và cạnh không thuộc bất kỳ đường đi ngắn nhất (s, t) nào.

Đặt $d = \text{dist}_G(s,t)$ (khoảng cách giữa s và t trong G), và với mỗi $i \in \{0,1,\ldots,d\}$, gọi D_i là tập các đỉnh trong $\widetilde{G}$ sao cho khoảng cách từ s đến đỉnh đó là i và từ đỉnh đó đến t là $d-i$.

Xây dựng đồ thị G' cho bài toán Token Jumping từ đồ thị $\widetilde{G}$:

- Biến mỗi tập D_i thành một cụm (thêm tất cả các cạnh giữa các đỉnh trong cùng D_i),
- Thêm cạnh giữa hai lớp liên tiếp D_i và D_{i+1}: giữa hai đỉnh $u \in D_i$ và $v \in D_{i+1}$, ta thêm cạnh uv vào G' nếu và chỉ nếu $uv \notin E(\widetilde{G})$.
- $V(G') = V(\widetilde{G})$
- $E(G') = \{uv : u \neq v, \exists i \text{ sao cho } u,v \in D_i\} \cup \bigcup_{i=0}^{d-1}\{uv : u \in D_i, v \in D_{i+1}, uv \notin E(\widetilde{G})\}$

Định lý 2.2.1. *Cho G' là một đồ thị với N đỉnh, là một trường hợp của bài toán Tái cấu hình đường đi ngắn nhất. Khi đó, tồn tại một trường hợp tương ứng của bài toán Token Jumping sao cho kích thước đầu vào là $O(N^2)$, và khoảng cách tái cấu hình bằng với khoảng cách của trường hợp ban đầu.*

2.2.3. Xây dựng trường hợp đầu vào SPR

Quy tắc tái cấu hình cho các đường đi ngắn nhất được định nghĩa như sau: Hai đường đi (s,t) ngắn nhất là liền kề trong đồ thị tái cấu hình các đường đi (s,t) ngắn nhất nếu và chỉ nếu chúng khác nhau tại đúng một đỉnh trong chuỗi các đỉnh tạo thành đường đi.

Phần này sẽ trình bày một họ các đồ thị G^k có kích thước tuyến tính theo k, nhưng đường kính của đồ thị tái cấu hình là $\Omega(2^k)$.

Đồ thị G^1 gồm các đỉnh:

$$\{x_i^1 \mid 1 \leq i \leq 7\} \cup \{y_i^1 \mid 1 \leq i \leq 6\} \cup \{s,t\}$$

và các cạnh:

$$\{(x_i^1, y_i^1), (x_{i+1}^1, y_i^1), (y_i^1, t) \mid i \leq 6\} \cup \{(s, x_i^1) \mid 1 \leq i \leq 7\}$$

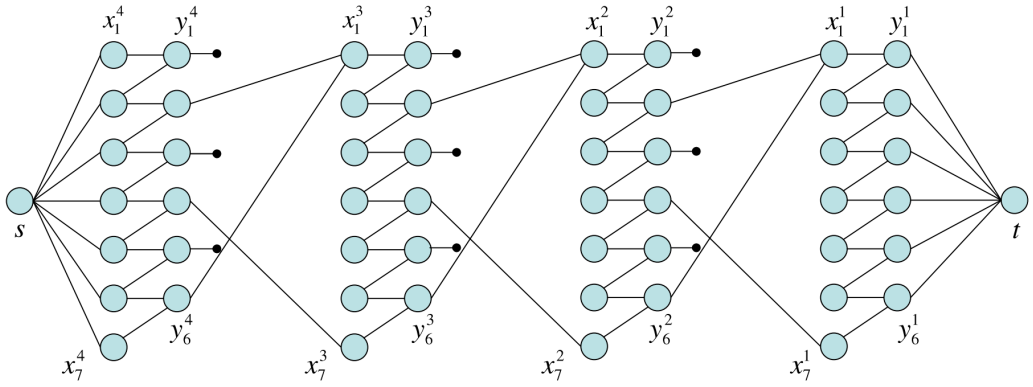
Đồ thị G^k (Hình 2.2) được định nghĩa đệ quy với các đỉnh:

$$\{x_i^k \mid 1 \leq i \leq 7\} \cup \{y_i^k \mid 1 \leq i \leq 6\} \cup V(G^{k-1})$$

và các cạnh:

$$\{(x_i^k, y_i^k), (x_{i+1}^k, y_i^k) \mid i \leq 6\} \cup \{(y_i^k, x_j^{k-1}) \mid i \in \{1,3,5\}, j \leq 7\} \cup$$
$$\{(y_2^k, x_1^{k-1}), (y_4^k, x_7^{k-1}), (y_6^k, x_1^{k-1})\} \cup E(G^{k-1} \setminus \{s\}) \cup \{(s, x_i^k) \mid 1 \leq i \leq 7\}$$

Gọi: $p_b^k = s, x_1^k, y_1^k, \ldots, x_1^1, y_1^1, t$, $p_e^k = s, x_7^k, y_6^k, x_1^{k-1}, \ldots, x_1^1, y_1^1, t$. Ta xét bài toán tái cấu hình từ p_b^k đến p_e^k trong đồ thị G^k.



Hình 2.2: Đồ thị G^k với $k = 4$, trong đó, khoảng cách tái cấu hình giữa $p_b^k = s, x_1^k, y_1^k, \ldots, x_1^1, y_1^1, t$ và $p_\epsilon^k = s, x_7^k, y_6^k, x_1^{k-1}, x_1^{k-1}, \ldots, x_1^1, y_1^1, t$ là $\Theta(2^k)$. Một cạnh có đầu mút là chấm nghĩa là đỉnh đó được nôi với tất cả các đỉnh ở lớp tiếp theo.

Bổ đề 2.2.1. *Gọi p là đường đi ngắn nhất trong G^k đi qua y_1^k, và q là đường đi đi qua y_6^k. Khi đó khoảng cách tái cấu hình giữa p và q ngắn nhất là $9(2^{k-1})$.*

Chứng minh. Ta chứng minh bằng phương pháp quy nạp theo k, trường hợp cơ sở là hiển nhiên. Gọi $\rho = p_1, \ldots, p_n$ là dãy tái cấu hình ngắn nhất từ p đến q.

Gọi i' là chỉ số nhỏ nhất sao cho $p_{i'+1}$ chứa y_4^k, và $i \leq i'$ là chỉ số nhỏ nhất sao cho mọi đường đi $p_i, ..., p_{i'}$ đều chứa y_3^k.

Theo cách xây dựng đồ thị, p_{i-1} chứa y_1^{k-1} và $p_{i'+1}$ chứa y_6^{k-1}. Do đó ta được, p_i chứa y_1^{k-1} và p_i' chứa y_6^{k-1}.

Theo giả thiết quy nạp, đoạn này có khoảng cách ngắn nhất là $i' - i + 1 = 9(2^{k-1} - 1)$.

Tương tự, gọi j' là chỉ số nhỏ nhất sao cho $p_{j'+1}$ chứa y_6^k, và $j \leq j'$ là chỉ số nhỏ nhất sao cho mọi đường đi từ $p_j, ..., p_{j'}$ đều chứa y_5^k.

Khi đó, theo cách xây dựng đồ thị, p_{j-1} chứa y_6^{k-1}, và $p_{j'+1}$ chứa y_1^{k-1}. Do đó, p_j chứa y_6^{k-1}, và p'_j chứa y_1^{k-1}.

Theo giả thiết quy nạp, đoạn thứ hai này có khoảng cách ngắn nhất là $j' - j + 1 = 9(2^{k-1} - 1)$.

Ngoài ra, do cấu trúc của đồ thị, ρ phải luôn đi qua y_x^k trước y_{x+1}^k nên $i' < j$. Do đó, khoảng cách của ρ ít nhất bằng tổng khoảng cách hai đoạn trên, và cộng thêm các bước chuyển cần thiết để chuyển từ y_1^k đến y_6^k, từ đó ta được điều phải chứng minh. □

Bổ đề 2.2.2. *Khoảng cách tái cấu hình giữa p_b^k và p_e^k là lớn nhất* $11(2^k - 1)$.

Chứng minh. Chứng minh bằng phương pháp quy nạp theo k, trường hợp cơ sở là điều hiển nhiên.

Ta có thể coi một chuỗi tái cấu hình không phải là một chuỗi các đường đi, mà là một chuỗi các đỉnh được thay đổi.

Áp dụng giả thiết quy nạp, ta có dãy ρ là dãy tái cấu hình ngắn nhất trong G^{k-1}, và $\text{rev}(\rho)$ là dãy đảo ngược của ρ (từ ρ_e^{k-1} về ρ_b^{k-1}).

Ta xây dựng dãy tái cấu hình ρ' như sau:

$$\rho' = x_2^k, y_2^k, x_3^k, y_3^k, \rho, x_4^k, y_4^k, x_5^k, y_5^k, \text{rev}(\rho), x_6^k, y_6^k, x_7^k$$

Dãy này là một dãy tái cấu hình từ p_b^k đến p_e^k có khoảng cách thỏa mãn bất đẳng thức. □

Định lý 2.2.2. *Khoảng cách tái cấu hình giữa p_b^k và p_e^k trong G^k là $\Theta(2^k)$.*

Định lý 2.2.2 đã xây dựng một chuỗi các trường hợp bài toán tái cấu hình Đường đi ngắn nhất sao cho khoảng cách giữa các trường hợp trong chuỗi tăng theo hàm mũ theo kích thước đầu vào. Sau đó, bằng cách lấy các trường hợp tái cấu hình đường đi ngắn nhất này, Định lý 2.2.1 cho ta một chuỗi các trường hợp khó cho bài toán Token Jumping

2.3. Trường hợp đầu vào xây dựng dựa trên bài toán tái cấu hình tô màu đồ thị (List Coloring Reconfiguration)

2.3.1. Bài toán tái cấu hình danh sách tô màu đồ thị

Định nghĩa 2.3.1. *Giả sử C là một tập hợp gồm k màu. Đối với đồ thị $G' = (V', E')$, giả sử mỗi đỉnh $v \in V'$ có một danh sách màu $L(v) \subseteq C$. Khi đó,* ***danh sách tô màu (list coloring)*** *của đồ thị G là việc gán một màu trong $L(v)$ cho mỗi đỉnh $v \in V'$ sao cho không có hai đỉnh kề nhau nhận cùng một màu.*

Bài toán tái cấu hình danh sách tô màu đồ thị (LCR) *được định nghĩa như sau: chúng ta được cho hai danh sách tô màu của G, và yêu cầu được đặt ra là xác định liệu có thể biến đổi danh sách tô màu này thành danh sách tô màu kia bằng cách tô màu lại một đỉnh sao cho tất cả các kết quả trung gian đều là các cách tô màu danh sách hợp lệ của G.*

2.3.2. Phép giảm từ bài toán LCR sang bài toán Token Jumping

Định lý 2.3.1. *Cho một trường hợp của bài toán LCR, giả sử $G' = (V', E')$ là một đồ thị với N đỉnh và M cạnh, $|C| = k$. Khi đó, có một trường hợp tương ứng trong bài toán Token Jumping sao cho kích thước đầu vào là $O(k^2N + kM)$ và khoảng cách của nó bằng khoảng cách của trường hợp LCR ban đầu.*

Chứng minh. Chúng ta xây dựng một đồ thị G cho bài toán Token Jumping như sau:

Đối với mỗi đỉnh v trong G', ta xây dựng một cụm mới gồm $|L(v)|$ đỉnh; mỗi đỉnh trong cụm tương ứng với một màu trong danh sách $L(v)$.

Đối với mỗi cạnh uv trong G', nếu $L(u) \cap L(v) \neq \emptyset$, thì ta nối hai đỉnh trong các cụm của u và v tương ứng với mỗi màu $c \in L(u) \cap L(v)$.

Mỗi cách tô màu danh sách của G' tương ứng với một tập độc lập cực đại của G. Do đó, khoảng cách tái cấu hình giữa hai cách tô màu trong G' bằng khoảng cách giữa hai tập độc lập trong G khi thực hiện bài toán Token Jumping. □

2.3.3. Xây dựng trường hợp đầu vào LCR

Định nghĩa 2.3.2. *Một cách tô màu (colouring) κ của một đường đi (u, v) được gọi là cách tô màu (c, d) nếu $\kappa(u) = c$ và $\kappa(v) = d$. Một đường đi P từ u đến v với các danh sách màu (colour list) L, trong đó $a \in L(u)$ và $b \in L(v)$, được gọi là* ***đường đi cấm tô màu (forbidding path)*** *(a, b) nếu thỏa mãn các điều kiện sau:*

Một phép tô màu (c, d) tồn tại nếu và chỉ nếu $c \in L(u)$, $d \in L(v)$ và $(c, d) \neq (a, b)$. Cặp (c, d) như vậy được gọi là ***chấp nhận được (admissible)*** *đối với P.*

Nếu cả hai cặp (c, d) và (c', d) đều chấp nhận được, thì với mọi cách tô màu (c, d), tồn tại một dãy các bước tô lại màu mà không thay đổi màu của đỉnh v, chỉ thay đổi màu ở đỉnh u ở bước cuối cùng để có được cách tô màu (c', d), không bao giờ tô lại đỉnh v, và chỉ tô lại đỉnh u trong bước cuối cùng. Tương tự, khẳng định cũng đúng với các cặp chấp nhận được (c, d) và (c, d').

Trong những phần sau, khi nói "thêm một đường đi cấm tô màu (a, b) giữa u và v", điều này có nghĩa là: Ta thêm vào đồ thị một đường đi (u_0, v_0) có tập màu là đường đi cấm tô màu (a, b), với $L(u_0) = L(u)$ và $L(v_0) = L(v)$, sau đó xác định $u = u_0$ và $v = v_0$.

Như vậy, với các phép tô màu và tô lại màu của các đỉnh u và v trong đồ thị kết quả, các tính chất ở trên vẫn được đảm bảo. Điều này có nghĩa là trong các chứng minh, ta không cần quan tâm chi tiết đến việc tô màu các đỉnh bên trong đường đi; ta chỉ cần giả sử rằng việc tô lại màu của u và v là khả thi, miễn là chúng không đồng thời mang màu a và b.

Bổ đề tiếp theo sẽ cho thấy rằng ta không cần mô tả cụ thể một đường đi cấm tô màu (a, b) mỗi lần; miễn là $L(u), L(v) \neq \{1, 2, 3, 4\}$, thì luôn tồn tại một đường đi như vậy.

Bổ đề 2.3.1. *Với mọi tập màu $L_u \subset \{1, 2, 3, 4\}$, $L_v \subset \{1, 2, 3, 4\}$ và các màu $a \in L_u$, $b \in L_v$, luôn tồn tại một đường cấm tô màu (a, b) nối đỉnh $u - v$ (ký hiệu P) thỏa mãn: $L(u) = L_u$, $L(v) = L_v$ và $L(w) \subseteq \{1, 2, 3, 4\}$, và độ dài chẵn không vượt quá 6.*

Chứng minh. Giả sử $c \in \{1, 2, 3, 4\} \setminus L(u)$ và $d \in \{1, 2, 3, 4\} \setminus L(v)$.

Nếu $c \neq d$, ta xây dựng một đường đi P gồm có độ dài 4 với danh sách màu dọc theo đường đi như sau: $L(u), \{a, c\}, \{c, d\}, \{d, b\}, L(v)$.

Ta chứng minh rằng đây là một đường đi cấm tô màu (a, b):

- Nếu trong một cách tô màu nào đó, đỉnh u có màu a, thì đỉnh thứ hai phải có màu c, đỉnh thứ ba có màu d, đỉnh thứ tư có màu b, do đó đỉnh v không thể có màu b.
- Tương tự, nếu đỉnh v có màu b, lập luận ngược lại cho thấy đỉnh u không thể có màu a.

Ngoài ra, ta có thể thấy rằng với mọi cặp màu (x, y) chấp nhận được, luôn tồn tại một cách tô màu (x, y) cho đường đi P. Cách tô màu này là duy nhất nếu $x = a$ hoặc $y = b$. Nếu không, tất cả các cách tô màu (x, y) đều có thể thu được từ nhau chỉ bằng cách tô lại các đỉnh bên trong của P.

Các phép tô màu kề nhau dạng (x, y) và (x, y') được xác định như sau:

- Nếu $x = a$, thì cả hai cách tô màu đều là duy nhất và kề nhau.
- Nếu $x \neq a$, ta có thể tìm các cách tô màu kề nhau bằng cách (nếu cần) tô màu đỉnh kề đỉnh u bằng màu a, đỉnh giữa là màu c, và đỉnh kề đỉnh v là màu d, trong cả hai cách tô màu.

Vì vậy, ta kết luận rằng đường đi P với các tập màu như trên là một đường đi cấm tô màu (a, b) với các tính chất cần thiết.

Nếu $c = d$, ta xây dựng một đường đi P có độ dài 6 với danh sách các màu như sau:

$$L(u), \{a, c\}, \{c, e\}, \{e, f\}, \{f, c\}, \{c, b\}, L(v),$$

với $e \in \{1, 2, 3, 4\} \setminus \{a, c\}$, $f \in \{1, 2, 3, 4\} \setminus \{b, c\}$, và $e \neq f$.

Tương tự như trên, ta có thể chứng minh rằng đây cũng là một đường đi cấm tô màu (a, b) với các tính chất cần thiết. □

Khẳng định 2.3.1. *Kích thước của tập đỉnh $V(G_N)$ và tập cạnh $E(G_N)$ đều bị chặn bởi một hàm bậc hai $O(N^2)$.*

Chứng minh. Đồ thị G_N được cấu thành từ các thành phần: N tam giác, $\frac{N(N-1)}{2}$ đường cấm tô màu loại (4,4), $\frac{N(N-1)}{2}$ đường cấm tô màu loại (1,4) hoặc (2,4).

Theo Bổ đề 2.3.1, ta giả định rằng tất cả các đường cấm tô màu (a, b) đều có độ dài tối đa là 6. Từ đó suy ra:

$$|V(G_N)| \leq 3N + 5N(N-1) \in O(N^2),$$
$$|E(G_N)| \leq 3N + 6N(N-1) \in O(N^2).$$

Để chứng minh rằng tồn tại một cặp tô màu của G_N sao cho để chuyển từ tô màu này sang tô màu kia cần số bước tô lại tinh theo hàm mũ (theo N), ta chỉ cần xét màu của các đỉnh v_i. Các màu này có thể coi như N bit nhận giá trị 1 hoặc 2.

Gọi một cách tô màu κ của G_N là cách tô màu dạng $(c_1, c_2, \ldots, c_N)$ nếu $\kappa(v_i) = c_i$ với mọi chỉ số i. Tập hợp tất cả các cách tô màu cùng dạng $(c_1, c_2, \ldots, c_N)$ tạo thành một lớp tô màu $(c_1, c_2, \ldots, c_N)$. □

1111 → 2111 → 2211 → 1211 →
1221 → 2221 → 2121 → 1121 →
1122 → 2122 → 2222 → 1222 →
1212 → 2212 → 2112 → 1112

Hình 2.3: Các lớp tô màu được đi qua trên một đường đi ngắn nhất giữa tô màu $(1, 1, 1, 1)$ và tô màu $(1, 1, 1, 2)$ của đồ thị G_4.

Quan sát 2.3.1. *Mọi lớp tô màu (colour class) $(c_1, \ldots, c_N)$ với $c_i \in \{1, 2\}$ đều không rỗng.*

Chứng minh. Xét cách tô màu κ thỏa mãn:

$\kappa(v_i) = c_i$, $\kappa(v_{0i}) = 3 - c_i$, và $\kappa(v_i^*) = 3$ với mọi i.

Do tất cả các tam giác đều bị khóa, phép tô màu này không vi phạm bất kỳ ràng buộc nào từ các đường cấm tô màu, do đó có thể mở rộng thành một phép tô màu hoàn chỉnh cho đồ thị G_N. □

Bổ đề 2.3.2. *Cho hai bộ $(x_1, \ldots, x_N)$ và $(y_1, \ldots, y_N)$ phân biệt với mọi $x_i, y_i \in \{1, 2\}$.*

- *Nếu hai bộ chỉ khác nhau tại vị trí i, và$x_{i-1} = 2$ và $x_j = 1$ với mọi $j < i - 1$, thì từ một cách tô màu bất kỳ thuộc lớp $(x_1, \ldots, x_N)$, ta có thể chuyển đến một cách tô màu thuộc lớp $(y_1, \ldots, y_N)$ thông qua một chuỗi đổi màu mà không cần rời khỏi lớp màu $(x_1, \ldots, x_N)$ trong các bước trung gian.*

- *Ngược lại, không tồn tại cách tô màu nào thuộc lớp* $(x_1, \ldots, x_N)$ *kề với cách tô màu thuộc lớp* $(y_1, \ldots, y_N)$.

Chứng minh. Giả sử các điều kiện trên được thỏa mãn. Ta chứng minh rằng mọi cách tô màu thuộc lớp $\kappa \in (x_1, \ldots, x_N)$ có thể chuyển thành cách tô màu thuộc lớp $\in (y_1, \ldots, y_N)$. Theo định nghĩa đường đi cấm tô màu (a, b), ta có thể bỏ qua việc đổi màu các đỉnh bên trong các đường đi này vì bất kỳ cách đổi màu nào cần thiết trên các đỉnh này đều luôn thực hiện được.

Trước tiên, ta chứng minh cách tô lại màu κ thành một phép tô màu thuộc lớp $(x_1, \ldots, x_N)$, trong đó chỉ có tam giác thứ i là đang mở khóa. Nếu tất cả các tam giác đều bị khóa trong κ, ta có thể lập tức tô lại v_i^* thành màu 4 mà không vi phạm bất kỳ ràng buộc nào từ các đường cấm tô màu.

Ngược lại, nếu đang có đúng một tam giác được mở khóa, giả sử tam giác này là tam giác thứ j, với $j \neq i$, ta sẽ khóa tam giác này lại. Nếu không thể ngay lập tức tô lại v_j^* thành màu 3, tức là $\kappa(v_j^0) = 3$. Khi đó, ta thay đổi màu của đỉnh này thành $\kappa(v_j^0) := 3 - \kappa(v_j)$, lúc này tam giác j có thể bị khóa lại.

Tiếp theo, tam giác i có thể được mở khóa: vì không có tam giác nào khác đang mở khóa nên các đường đi cấm tô màu $(4, 4)$ không gây ra hạn chế nào. Do $\kappa(v_{i-1}) = 2$ và $\kappa(v_j) = 1$ với mọi $j < i - 1$, các đường đi cấm $(4, 1)$ và $(4, 2)$ bắt đầu từ v_i^* cũng không gây ra hạn chế. Lúc này, đặt $\kappa(v_i^0) := 3$, rồi đặt $\kappa(v_i) := y_i$, để thu được một cách tô màu thuộc lớp $(y_1, \ldots, y_N)$. Điều này chứng minh cho phát biểu thứ nhất.

Giả sử α là một cách tô màu thuộc lớp $(x_1, \ldots, x_N)$, β là một cách tô màu thuộc lớp $(y_1, \ldots, y_N)$, và α với β là hai cách tô màu kề nhau. Điều này có nghĩa là chúng chỉ khác nhau tại một đỉnh duy nhất, và do hai bộ $(x_1, \ldots, x_N)$ và $(y_1, \ldots, y_N)$ khác nhau, nên α và β phải khác nhau chính xác tại một đỉnh v_i nào đó. Điều này có nghĩa là việc tam giác thứ i đang được mở khóa trong cả hai cách tô màu. Do các đường đi cấm tô màu $(4, 1)$ và $(4, 2)$ bắt đầu từ v_i^*, ta có $\alpha(v_{i-1}) = 2$ và $\alpha(v_j) = 1$ với mọi $j < i - 1$. Điều này là chứng minh cho phát biểu thứ hai. □

Từ Bổ đề 2.3.2 suy ra rằng mỗi lớp tô màu chỉ kề với nhiều nhất hai lớp tô màu khác. Trước hết, màu của đỉnh v_1 luôn có thể thay đổi. Ngoài ra, có tối đa một đỉnh v_i sao cho v_{i-1} có màu 2 và v_j có màu 1 với $j < i - 1$; đây là đỉnh duy nhất trong các đỉnh $v_1, \ldots, v_N$ mà ta có thể thay đổi màu mà không cần thay đổi màu của các đỉnh

khác trước. Hình 2.3 thể hiện tất cả các lớp tô màu của G_4 và thứ tự cần thự hiện để đi từ cách tô màu $(1,1,1,1)$ đến cách tô màu $(1,1,1,2)$ của G_4 — tất cả 16 lớp khác nhau đều cần được duyệt qua. Điều này được chứng minh một cách đầy đủ cho mọi N trong 2.3.2.

Định lý 2.3.2. *Mỗi đồ thị G_N có hai cách tô màu α và β nằm trong cùng một thành phần liên thông của $C(G_N, L)$, sao cho khoảng cách giữa chúng ít nhất là 2^{N-1}.*

Chứng minh. Với cách tô màu α, ta chọn một cách tô màu thuộc lớp $(1,\ldots,1)$; tồn tại một tô màu như vậy theo Quan sát 2.3.1. Cách tô màu β sẽ thuộc lớp $(1,\ldots,1,2)$. Trước hết, ta chứng minh bằng phương pháp quy nạp rằng có thể biến đổi cách tô màu này thành cách tô màu kia thông qua việc đổi màu các đỉnh.

Giả thuyết quy nạp: Tồn tại một đường đi trong $C(G_N, L)$ từ cách tô màu α' bất kỳ thuộc lớp $(1,\ldots,1,x_0,x_1,\ldots,x_{N-n})$ đến một tô màu β' thuộc lớp $(1,\ldots,1,3-x_0,x_1,\ldots,x_{N-n})$.

Hai cách tô màu khác nhau tại đỉnh v_n: ta có $\alpha'(v_n) = x_0$ và $\beta'(v_n) = 3 - x_0$, trong khi với mọi $i \neq n$, ta có $\alpha'(v_i) = \beta'(v_i)$.

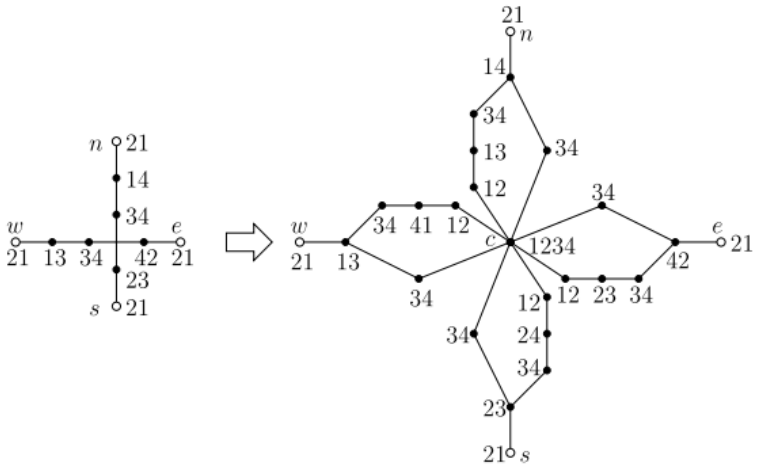
Nếu $n = 1$, mệnh đề được suy ra trực tiếp từ Bổ đề 2.3.2.

Nếu $n > 1$, ta thực hiện:

- Từ α', ta đổi màu để thu được một cách tô màu thuộc $(1,\ldots,1,2,x_0,x_1,\ldots,x_{N-n})$ (chỉ khác lớp ban đầu tại vị trí thứ $n-1$), dùng giả thuyết quy nạp.
- Sau đó ta lại đổi màu để thu một cách tô màu thuộc lớp $(1,\ldots,1,2,3-x_0,x_1,\ldots,x_{N-n})$, sử dụng Bổ đề 2.3.2.
- Cuối cùng, dùng lại giả thuyết quy nạp, ta có thể đổi màu để thu được một cách tô mùa thuộc lớp $(1,\ldots,1,1,3-x_0,x_1,\ldots,x_{N-n})$. Điều này chứng minh khẳng định.

Tiếp theo, ta chứng minh rằng để đi từ một cách tô màu thuộc lớp $(1,\ldots,1)$ đến một cách tô màu thuộc lớp $(1,\ldots,1,2)$, cần phải đi qua ít nhất $2^N - 2$ lớp tô màu khác.

Giả thuyết quy nạp: Để đi từ một tô màu $(1,\ldots,1,1,x_1,\ldots,x_{N-n})$ đến một tô màu $(1,\ldots,1,2,y_1,\ldots,y_{N-n})$, cần đi qua ít nhất $2^n - 2$ lớp tô màu trung gian.



Hình 2.4: Một thành phần giao cắt tương ứng với hai đường đi cấm tô màu dạng (1, 2)

Nếu $n = 1$, khẳng định hiển nhiên đúng.

Nếu $n > 1$, ta xét một đường đi ngắn nhất giữa hai cách tô màu thuộc hai lớp trên (nếu tồn tại). Tại một thời điểm nào đó trong quá trình tô lại, màu của v_n sẽ được thay đổi lần đầu tiên; trước đó, theo Bổ đề 2.3.2, ta phải có một cách tô màu thuộc lớp $(1, \ldots, 1, 2, 1, z_1, \ldots, z_{N-n})$ (trong cách tô màu này đỉnh v_{n-1} có màu 2).

Theo giả thuyết quy nạp, đã phải đi qua ít nhất $2^{n-1} - 2$ lớp tô màu trước khi đến được lớp này. Sau đó, khi đổi màu đỉnh v_n thành 2, ta được một cách tô màu thuộc lớp $(1, \ldots, 1, 2, 2, z_1, \ldots, z_{N-n})$. Theo giả thuyết quy nạp một lần nữa, cần đi qua ít nhất $2^{n-1} - 2$ lớp tô màu trước khi đến được lớp $(1, \ldots, 1, 2, y_1, \ldots, y_{N-n})$. Điều này có nghĩa là tổng cộng đã đi qua ít nhất $2^n - 4 + 2 = 2^n - 2$ lớp tô màu trung gian trong quá trình đổi màu các đỉnh. Ta được điều phải chứng minh. □

Định lý 2.3.2 đã xây dựng một chuỗi các trường hợp bài toán tái cấu hìnhdanh sách tô màu sao cho khoảng cách giữa các trường hợp trong chuỗi tăng theo hàm mũ theo kích thước đầu vào. Sau đó, bằng cách lấy các trường hợp tái cấu hình tô màu theo danh sách này, Định lý 2.3.1 cho ta một chuỗi các trường hợp khó cho bài toán Token Jumping

Chương 3

Giải thuật SAT cho bài toán Token Jumping

3.1. Phương pháp kiểm chứng mô hình có giới hạn (bounded model checking)

Ý tưởng của phương pháp kiểm tra mô hình có giới hạn (BMC) sử dụng SAT: Sử dụng các bộ giải SAT để kiểm chứng mô hình bằng cách tìm kiếm các phản ví dụ (counterexample) có độ dài giới hạn (gọi là giới hạn - bound k). Thay vì kiểm tra toàn bộ không gian trạng thái (rất lớn hoặc vô hạn), BMC tập trung kiểm tra xem có tồn tại một chuỗi các bước thực hiện dẫn đến việc vi phạm ràng buộc cần kiểm tra trong vòng k bước hay không. Nếu có, bộ giải SAT sẽ tìm ra phản ví dụ đó; nếu không, tiếp tục tăng k và kiểm tra.

Ví dụ: Mô hình máy trạng thái đơn giản M:

Xét một máy trạng thái đơn giản M bao gồm một thanh ghi dịch 3 bit x với các bit riêng lẻ được ký hiệu là $x[0]$, $x[1]$, và $x[2]$.

Tiên đề $T(x; x')$ biểu thị quan hệ chuyển trạng thái giữa các giá trị trạng thái hiện tại x và trạng thái kế tiếp x' và tương đương với:

$$(x'[0] = x[1]) \wedge (x'[1] = x[2]) \wedge (x'[2] = 1).$$

Ở trạng thái ban đầu, nội dung của thanh ghi x có thể là bất kỳ giá trị nào. Tiên đề $I(x)$ biểu thị tập các trạng thái ban đầu và luôn đúng.

Thanh ghi dịch này được thiết kế để trống (tất cả các bit đều bằng 0) sau ba lần dịch liên tiếp. Tuy nhiên, chúng ta đưa một lỗi vào quan hệ chuyển trạng thái cho giá trị trạng thái tiếp theo của $x[2]$, trong đó giá trị sai dược sử dụng là 1 thay vì 0. Do đó, tính chất cho rằng cuối cùng thanh ghi sẽ trống ($x = 0$) sau một số bước đủ lớn không còn hợp lệ.

Chúng ta chuyển bài toán kiểm tra mô hình "phổ quát" $\mathbf{AF}(x = 0)$ thành bài toán kiểm tra mô hình "tồn tại" $\mathbf{EG}(x \neq 0)$ bằng cách phủ định công thức. Sau đó,thya vì tìm kiếm một đường đi bất kỳ, ta chỉ xét các đường đi có tối đa $k + 1$ trạng thái, ví dụ, chọn $k = 2$.

Gọi ba trạng thái đầu tiên của đường đi này là x_0, x_1 và x_2, trong đó x_0 là trạng thái ban đầu (xem Hình 3.1). Vì nội dung ban đầu của thanh ghi x có thể là bất kỳ giá trị nào, nên ta không có ràng buộc gì với x_0.

Ta mở rộng quan hệ chuyển tiếp hai lần thu được công thức mệnh đề f_m được định nghĩa như sau:

$$f_m = I(x_0) \wedge T(x_0, x_1) \wedge T(x_1, x_2)$$

Sau khi mở rộng định nghĩa của T và I, ta thu được công thức sau:

Bước 1:

$$(x_1[0] = x_0[1]) \wedge (x_1[1] = x_0[2]) \wedge (x_1[2] = 1)$$

Bước 2:

$$(x_2[0] = x_1[1]) \wedge (x_2[1] = x_1[2]) \wedge (x_2[2] = 1)$$

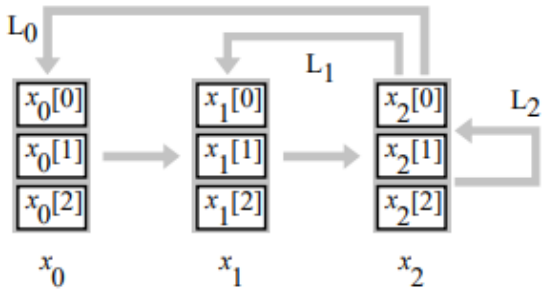
Bất kỳ đường đi nào gồm ba trạng thái và là một chuỗi trạng thái lặp vô hạn trong đó x không bao giờ bằng 0 cho công thức $G(x \neq 0)$ đều phải chứa một vòng lặp. Do đó, ta yêu cầu phải có một bước chuyển từ trạng thái x_2 quay trở lại trạng thái ban đầu x_0, trạng thái thứ hai x_1, hoặc chính nó x_2 (Hình 3.1).

Ta biểu diễn bước chuyển này là L_i, được định nghĩa là $T(x_2, x_i)$, tương đương với công thức sau:

$$(x_i[0] = x_2[1]) \wedge (x_i[1] = x_2[2]) \wedge (x_i[2] = 1)$$

Cuối cùng, ta cần đảm bảo rằng đường đi này phải thỏa mãn các ràng buộc được định nghĩa bởi công thức $G(x \neq 0)$. Trong trường hợp này, tính chất S_i, được định nghĩa là $x_i \neq 0$, phải đúng tại mỗi trạng thái. S_i tương đương với công thức sau:

$$(x_i[0] = 1) \vee (x_i[1] = 1) \vee (x_i[2] = 1)$$



Hình 3.1: Mở rộng quan hệ chuyển tiếp hai lần và thêm một vòng lặp ngược.

Kết hợp các phần trên, ta thu được công thức mệnh đề sau:

$$f_M \wedge \bigvee_{i=0}^{2} L_i \wedge \bigwedge_{i=0}^{2} S_i \tag{1}$$

Công thức này được thoả mãn nếu và chỉ nếu tồn tại một phản ví dụ có độ dài 2 cho công thức ban đầu $F(x = 0)$.

Trong ví dụ trên, ta tìm được một phép gán thỏa mãn cho công thức (1) bằng cách đặt:

$$x_i[j] := 1 \quad \text{với mọi } i = 0, 1, 2 \text{ và } j = 0, 1, 2.$$

3.2. Mã hóa bằng bộ đếm tuần tự

Trong phần này trình bày phương pháp mã hóa mệnh đề chuẩn CNF cho ràng buộc đếm (cardinality constraint) dạng $\leq k(x_1, \ldots, x_n)$, dựa trên mạch bộ đếm tuần tự (sequential counter circuit). Mạch này được minh họa trong Hình 3.2 và tính toán các tổng từng phần $s_i = \sum_{j=1}^{i} x_j$ với các giá trị i tăng dần đến n. Các giá trị s_i được biểu diễn dưới dạng số nguyên ở hệ đơn vị (unary numbers). Các bit tràn (overflow bits) v_i sẽ được đặt là đúng (true) nếu tổng từng phần s_i vượt quá k.

Để chuyển mạch này sang dạng CNF, trước tiên ta xây dựng các phương trình định nghĩa cho các bit tổng từng phần $s_{i,j}$ và các bit tràn v_i. Sau đó, đơn giản hóa các phương trình này với lưu ý rằng tất cả các bit tràn phải bằng 0 (vì tổng không được vượt quá k). Các phương trình kết quả sau đó được chuyển thành dạng CNF, và có thể lược bỏ một chiều của các phương trình nhờ vào tính chất cực tính (polarity) trong các mệnh đề (kỹ thuật cơ bản này do Tseitin giới thiệu[21], và sau đó được phát triển

thêm bởi nhiều tác giả, ví dụ như Jackson và Sheridan[22]).

Từ đó, ta thu được một tập mệnh đề, gọi là $LT^{n,k}_{SEQ}$, định nghĩa ràng buộc $\leq k(x_1, \ldots, x_n)$ dựa trên bộ đếm tuần tự, với $k > 0$ và $n > 1$:

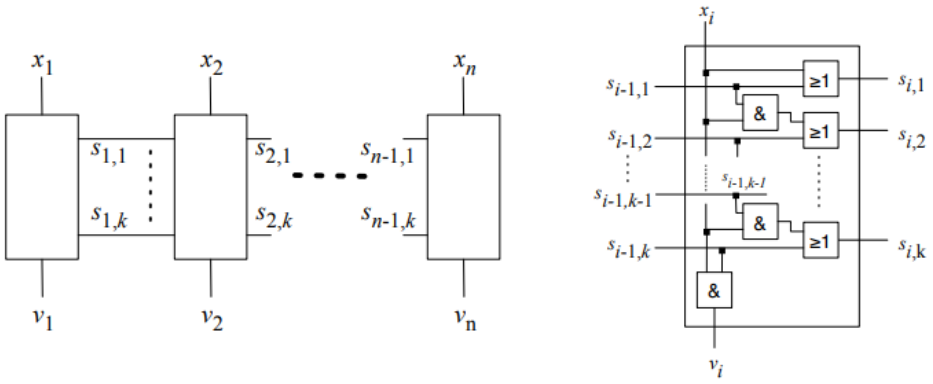
$$
\begin{aligned}
&(\neg x_1 \vee s_{1,1}) && \\
&(\neg s_{1,j}) && \text{với } 1 < j \leq k \\
&(\neg x_i \vee s_{i,1}) && \\
&(\neg s_{i-1,1} \vee s_{i,1}) && \\
&(\neg x_i \vee \neg s_{i-1,j-1} \vee s_{i,j}) && \\
&(\neg s_{i-1,j} \vee s_{i,j}) && \text{với } 1 < j \leq k \\
&(\neg x_i \vee \neg s_{i-1,k}) && \text{với } 1 < i < k \\
&(\neg x_n \vee \neg s_{n-1,k}) &&
\end{aligned}
$$

Tập mệnh đề $LT^{n,k}_{SEQ}$ bao gồm $2nk + n - 3k - 1$ mệnh đề và cần $(n-1) \cdot k$ biến phụ để mã hóa.

Dưới đây là tập mệnh đề $LT^{SEQ}_{n,1}$ cho trường hợp đặc biệt $k = 1$ dưới dạng công thức:

$$
\begin{aligned}
&(\neg x_1 \vee s_{1,1}) \wedge (\neg x_n \vee \neg s_{n-1,1}) \\
&\wedge \bigwedge_{1<i<n} [(\neg x_i \vee s_{i,1}) \wedge (\neg s_{i-1,1} \vee s_{i,1}) \wedge (\neg x_i \vee \neg s_{i-1,1})]
\end{aligned}
$$

Tập mệnh đề này có $3n - 4$ mệnh đề và $n - 1$ biến phụ.



Hình 3.2: **Bên trái:** Mạch để tính $\leq k$ $(x_1, \ldots, x_n)$. $s_{i,j}$ là chữ số thứ j của tổng một phần thứ i - s_i dưới dạng biểu diễn hệ đơn vị; các biến v_i là các bit tràn, biểu thị tổng một phần thứ i lớn hơn k.
Bên phải: Tiểu mạch để tính tổng một phần s_i dưới dạng biểu diễn hệ đơn vị.

Định lý 3.2.1. *$LT^{SEQ}_{n,k}$ là một mã hóa mệnh đề (clausal encoding) của ràng buộc $\leq k(x_1, \ldots, x_n)$, yêu cầu $\mathcal{O}(n \cdot k)$ mệnh đề và $\mathcal{O}(n \cdot k)$ biến phụ.*

Mã hóa $LT^{SEQ}_{n,k}$ còn thỏa mãn điều kiện:

- Nếu có hơn k biến x_i được gán bằng đúng (vi phạm ràng buộc $\leq k$), việc suy luận đơn vị (unit propagation) có thể phát hiện ra điều này trong thời gian tuyến tính.
- Ngoài ra, nếu một gán một phần (partial assignment) đã gán đúng cho k biến x_i, giá trị của tất cả các biến còn lại có thể được suy ra thông qua suy luận đơn vị.

3.3. Xây dựng công thức SAT và kỹ thuật giải tăng dần

Giả sử (G, I_s, I_t) là một bài toán Token Jumping, trong đó $G = (V, E)$. Bộ giải SAT trong phần này sử dụng phương pháp kiểm tra mô hình giới hạn. Cụ thể, với một giới hạn (số nguyên) ℓ, bộ giải xác định liệu có tồn tại một chuỗi tái cấu hình có độ dài ℓ giữa I_s và I_t; và sau đó, chúng tôi tăng dần giới hạn ℓ.

Đầu tiên, với một số nguyên $i \in \{0, 1, \ldots, \ell\}$ và một đỉnh $u \in V$, ta định nghĩa một biến mệnh đề $p_{i,u}$ sao cho nó đúng khi và chỉ khi một token được đặt lên đỉnh u tại bước i; ta định nghĩa tập độc lập ban đầu I_s, tướng ứng với bước 0.

Để biểu diễn một tập độc lập tại bước i, ta giới thiệu một hàm Token(i) như sau:

$$Token(i) = \begin{cases} \bigwedge_{u \in I_s} p_{i,u} \wedge \bigwedge_{u \notin I_s} \neg p_{i,u} & \text{nếu } i = 0, \\ \bigwedge_{\{u,v\} \in E} (\neg p_{i,u} \vee \neg p_{i,v}) & \text{nếu } 0 < i < \ell, \\ \bigwedge_{u \in I_t} p_{i,u} \wedge \bigwedge_{u \notin I_t} \neg p_{i,u} & \text{nếu } i = \ell. \end{cases}$$

Tiếp theo, ta sử dụng hai loại biến mệnh đề $q^{10}_{i,u}$ và $q^{01}_{i,u}$ để biểu diễn sự di chuyển của token tại một đỉnh $u \in V$ giữa bước i và bước $i+1$: $q^{10}_{i,u}$ (hoặc $q^{01}_{i,u}$) đúng khi và chỉ khi một token bị loại bỏ khỏi u (hoặc được đặt vào u) giữa bước i và bước $i+1$.

Khi đó, ràng buộc $Jump(i)$ biểu diễn sự di chuyển token giữa hai bước như sau:

$$Jump(i) = \bigwedge_{u \in V} (q^{10}_{i,u} \leftrightarrow (p_{i,u} \wedge \neg p_{i+1,u})) \wedge \bigwedge_{u \in V} (q^{01}_{i,u} \leftrightarrow (\neg p_{i,u} \wedge p_{i+1,u})) \wedge \sum_{u \in V} q^{10}_{i,u} = 1 \wedge \sum_{u \in V} q^{01}_{i,u} = 1.$$

Lưu ý rằng ràng buộc $\sum_i x_i = 1$ có thể được mã hóa thành các mệnh đề logic bằng cách sử dụng bộ đếm tuần tự.

Cuối cùng, để kiểm tra liệu có tồn tại một chuỗi tái cấu hình có độ dài ℓ bằng cách xét tính thỏa mãn của công thức sau Ψ_ℓ:

$$\Psi_\ell = \bigwedge_{i=0}^{\ell} Token(i) \wedge \bigwedge_{i=0}^{\ell-1} Jump(i).$$

Bộ giải SAT tăng dần ℓ và thực hiện kiểm tra tính khả thi của Ψ_ℓ. Phương pháp dừng lại khi công thức Ψ_ℓ trở nên khả thi và xuất ra mô hình của nó như một chuỗi tái cấu hình ngắn nhất.

Chương 4

Thực nghiệm và kết quả

4.1. Mô tả dữ liệu đầu vào

Để đánh giá hiệu năng của thuật toán SAT giải bài toán tái cấu hình tập độc lập, ta xây dựng và sử dụng tập dữ liệu đầu vào theo định dạng bao gồm hai tệp chính cho mỗi trường hợp đầu vào:

- Tệp đồ thị `.col`: mô tả cấu trúc đồ thị đầu vào.
- Tệp cấu hình `.dat`: chứa hai tập độc lập ban đầu và tập mục tiêu.

Định dạng tệp `.col`:

- Dòng đầu tiên có dạng: `p <số đỉnh> <số cạnh>`

  Ví dụ: p 13 66 cho biết đồ thị có 13 đỉnh và 66 cạnh.

- Các dòng sau có dạng: `e u v` biểu diễn một cạnh nối giữa hai đỉnh u và v

  Ví dụ: `e 1 2` cho biết tồn tại cạnh nối giữa đỉnh 1 và đỉnh 2.

Định dạng tệp `.dat`:
Tệp này mô tả hai tập độc lập:

- Dòng đầu tiên bắt đầu bằng `s` biểu diễn tập độc lập ban đầu.
- Dòng thứ hai bắt đầu bằng `t` biểu diễn tập độc lập mục tiêu.

4.2. Ngôn ngữ và môi trường thực thi

Để triển khai thuật toán SAT giải bài toán tái cấu hình các tập độc lập, ta sử dụng ngôn ngữ Python. Python là một ngôn ngữ bậc cao, phổ biến, dễ sử dụng và có nhiều thư viện hỗ trợ tính toán.

Trong quá trình xây dựng và giải bài toán tái cấu hình tập độc lập, một số thư viện Python được sử dụng để hỗ trợ như sau:

- `os` và `time`: Hỗ trợ quản lý tệp và đo thời gian chạy của thuật toán.
- `networkx`: Dùng để biểu diễn đồ thị đầu vào.
- `pysat.formula.CNF` và `pysat.solvers.Solver` (từ thư viện PySAT): Đóng vai trò trung tâm trong việc mã hóa bài toán thành công thức logic và giải bài toán SAT tương ứng.

Thuật toán được triển khai trên máy tính cá nhân với cấu hình phần cứng và phần mềm như sau:

- Phần cứng: CPU Intel(R) Core(TM) i5-1135G7 gồm 4 lõi và 8 luồng, tốc độ 2.4 GHz; RAM 16 GB.
- Phần mềm: Mã nguồn chương trình được phát triển trên phần mềm Visual Studio Code, sử dụng ngôn ngữ lập trình Python phiên bản 3.13.3.

4.3. Thực nghiệm thuật toán

Thuật toán nhận đầu vào gồm: đồ thị (đọc từ file `.col`), tập độc lập ban đầu và mục tiêu (đọc từ file `.dat`).

Các ràng buộc SAT được mã hóa bao gồm: ràng buộc không cho phép 2 token nằm trên các đỉnh kề nhau cùng một thời điểm, ràng buộc di chuyển token theo từng bước, ràng buộc tập độc lập khởi đầu và kết thúc, ràng buộc mỗi bước chỉ có một token di chuyển.

Quy trình thực nghiệm của thuật toán là: Đối với mỗi trường hợp đầu vào, thuật toán chạy tăng dần số bước k từ 1 đến max_k (mặc định ở đây là 50) để tìm ra lời giải

cho bài toán. Với mỗi bước k, nếu thuật toán tìm thấy lời giải sẽ dừng lại và lưu kết quả, nếu chưa tìm thấy lời giải tiếp tục tăng k và tìm kiếm lời giải.

4.4. Kết quả thực nghiệm

Thực nghiệm được tiến hành trên sáu trường hợp đại diện thuộc ba chuỗi SPR, LGC và SAT được với kích thước đồ thị đầu vào được mô tả trong Bảng. Kết quả thực nghiệm được tổng hợp trong Bảng

Trường hợp	**Số đỉnh**	**Số cạnh**
SPR_instance001	13	66
SPR_instance002	26	150
LGC_instance001	13	15
LGC_instance002	38	44
SAT_instance001	2	1
SAT_instance002	6	5

Bảng 4.1: Thời gian chạy và số mệnh đề với các trường hợp thuộc chuỗi IS, LGC và SAT

Trường hợp	**Thời gian (s)**	**Số mệnh đề**	**Số bước tái cấu hình**
SPR_instance001	0.078	3,414	11
SPR_instance002	2.797	31,816	33
LGC_instance001	0.000	1,296	5
LGC_instance002	6.594	45,480	27
SAT_instance001	0.000	22	1
SAT_instance002	0.016	236	3

Bảng 4.2: Kết quả thực nghiệm trên các trường hợp thuộc chuỗi IS, LGC và SAT

Từ kết quả thực nghiệm cho thấy, các trường hợp đầu vào được giải trong thời gian nhanh chóng và đường đi tái cấu hình ngắn. Ta thấy, từ các trường hợp đầu vào đang xét phù hợp trong việc kiểm tra tính đúng đắn của thuật toán. Tuy nhiên, các trường hợp đầu vào với kích thước nhỏ này (số đỉnh < 40) vẫn chưa thể đánh giá được hiệu năng của thuật toán.

KẾT LUẬN

Trong khóa luận này, chúng tôi đã tập trung nghiên cứu bài toán tái cấu hình các tập độc lập trong đồ thị thông qua mô hình *Token Jumping*. Nội dung được triển khai qua bốn chương chính, phản ánh quá trình tiếp cận từ lý thuyết đến thực nghiệm:

- **Chương 1** đã trình bày các kiến thức cơ sở về lý thuyết đồ thị, tập độc lập, bài toán tái cấu hình và mô hình Token Jumping. Các định nghĩa và kết quả nền tảng này nhằm tạo tiền đề cho các chương sau.
- **Chương 2** đã đi vào tìm hiểu cách xây dựng các trường hợp đầu vào khó có khả năng tùy chỉnh kích thước, dựa trên các bài toán gốc như *SAT Reconfiguration*, *Shortest Path Reconfiguration*, *List Coloring Reconfiguration.*
- **Chương 3** đi vào tìm hiểu và trình bày thuật toán giải bài toán tái cấu hình bằng cách sử dụng thuật toán SAT kết hợp phương pháp kiểm chứng mô hình có giới hạn BMC để kiểm tra khả năng chuyển đổi giữa hai tập độc lập.
- **Chương 4** trình bày các kết quả thực nghiệm chi tiết trên một số trường hợp của ba bài toán đầu vào *SAT*, *SPR*, *LCR*. Qua đó nhìn nhận được tính đúng đắn của thuật toán sử dụng.

Những kết quả đạt được mới chỉ là bước đầu trong quá trình tiếp cận lĩnh vực tái cấu hình các tập độc lập. Trong tương lai, đề tài mở ra một số hướng phát triển tiềm năng:

- Mở rộng mô hình giải SAT với các phiên bản hioeeuj quả hơn, hoặc sử dụng những ngôn ngữ và công nghệ có hiệu năng cao hơn, ví dụ việc kết hợp ngôn ngữ *Scala* và bộ giải *CaDiCal*
- Xây dựng thêm các chuỗi bài toán đầu vào, ví dụ từ các bài toán tổ hợp khác như *Matching*, *Dominating Set Reconfiguration.*

Tài liệu tham khảo

[1] Takehide Soh - Takumu Watanabe - Jun Kawahara - Akira Suzuki - Takehiro Ito, *Scalable Hard Instances for Independent Set Reconfiguration*, In *Proceedings of the 22nd International Symposium on Experimental Algorithms (SEA 2024)*, LIPIcs, Volume 301, pages 26:1–26:15, 2024.

[2] John Asplund and Brett Werner. *Classification of reconfiguration graphs of shortest path graphs with no induced 4-cycles*. Discrete Mathematics, 343(1):111640, 2020.

[3] Armin Biere - Alessandro Cimatti - Edmund M. Clarke - Yunshan Zhu, *Symbolic model checking without BDDs*, In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, trang 193–207, 1999.

[4] Paul S. Bonsma, *Rerouting shortest paths in planar graphs*, Discrete Applied Mathematics, 231:95–112, 2017.

[5] Paul S. Bonsma and Luis Cereceda. Finding paths between graph colourings: PSPACEcompleteness and superpolynomial distances. Theoretical Computer Science, 410(50):5215–5226, 2009.

[6] Luis Cereceda, Jan van den Heuvel, and Matthew Johnson. Finding paths between 3-colorings. Journal of Graph Theory, 67(1):69–82, 2011.

[7] Niklas Eén and Niklas Sörensson. Temporal induction by incremental SAT solving. Electronic Notes in Theoretical Computer Science, 89(4), 2003.

[8] Kshitij Gajjar, Agastya Vibhuti Jha, Manish Kumar, and Abhiruk Lahiri. Reconfiguring shortest paths in graphs. In Proceedings of AAAI 2022, pages 9758–9766. AAAI Press, 2022.

[9] Parikshit Gopalan, Phokion G. Kolaitis, Elitza N. Maneva, and Christos H. Papadimitriou. The connectivity of Boolean satisfiability: Computational and structural dichotomies. SIAM Journal on Computing, 38(6):2330–2355, 2009.

[10] Tatsuhiko Hatanaka, Takehiro Ito, and Xiao Zhou. The list coloring reconfiguration problem for bounded pathwidth graphs. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E98.A(6):1168–1178, 2015.

[11] Takehiro Ito, Erik D. Demaine, Nicholas J. A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. Theoretical Computer Science, 412(12–14):1054–1065, 2011.

[12] Mark Jerrum. Counting, Sampling and Integrating: Algorithms and Complexity. Birkhäuser Verlag, Basel, 2003.

[13] Marcin Kamiński, Paul Medvedev, and Martin Milanič. Shortest paths between shortest paths. Theoretical Computer Science, 412(39):5205–5210, 2011.

[14] Kazuhisa Makino, Suguru Tamaki, and Masaki Yamamoto. An exact algorithm for the Boolean connectivity problem for k-CNF. Theoretical Computer Science, 412(35):4613–4618, 2011.

[15] Amer E. Mouawad, Naomi Nishimura, Vinayak Pathak, and Venkatesh Raman. Shortest reconfiguration paths in the solution space of Boolean formulas. SIAM Journal on Discrete Mathematics, 31(3):2185–2200, 2017.

[16] Carsten Sinz. Towards an optimal CNF encoding of boolean cardinality constraints. In Peter van Beek, editor, Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings, volume 3709 of Lecture Notes in Computer Science, pages 827–831. Springer, 2005.

[17] Alice Joffard, *Graph domination and reconfiguration problems*, Thèse de doctorat en Informatique, Université Claude Bernard Lyon 1, École Doctorale InfoMaths (ED 512), soutenue le 25 novembre 2020, NNT : 2020LYSE1216.

[18] Volker Turau - Christoph Weyer, *Finding shortest reconfiguration sequences of independent sets*, In *Core Challenge 2022: Solver and Graph Descriptions*, trang 3–14, 2022.

[19] Paul S. Bonsma, *The complexity of rerouting shortest paths*, Theoretical Computer Science, Volume 510, Pages 1–12, 2013.

[20] Nicolas Bousquet, Bastien Durain, Théo Pierron, and Stéphan Thomassé, *Extremal independent set reconfiguration*, Electronic Journal of Combinatorics, Volume 30, Number 3, Paper P3.8, 2023.

[21] Tseitin, G.S.: On the complexity of derivation in propositional calculus. In Slisenko, A.O., ed.: *Studies in Constructive Mathematics and Mathematical Logic.* (1970) 115–125.

[22] Jackson, P., Sheridan, D.: The optimality of a fast CNF conversion and its use with SAT. Technical Report APES-82-2004, APES Research Group (2004).

[23] Marcin Kamiński, Paul Medvedev, and Martin Milanič. Complexity of independent set reconfigurability problems. *Theoretical Computer Science*, 439:9–15, 2012. doi:10.1016/j.tcs.2012.03.004.

[24] Marcin Kamiński, Paul Medvedev, and Martin Milanič. *Complexity of independent set reconfigurability problems.* In *Theoretical Computer Science*, 439:9–15, 2012.