BỘ GIÁO DỤC VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Trần Quang

ĐỘ PHỨC TẠP CỦA BÀI TOÁN BIẾN ĐỔI ĐỒ THỊ VỀ ĐỒ THỊ ĐẦY ĐỦ

LUẬN VĂN THẠC SĨ TOÁN HỌC

Hà Nội - 2019

BỘ GIÁO DỤC
VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC
VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Trần Quang

ĐỘ PHỨC TẠP CỦA BÀI TOÁN BIẾN ĐỔI ĐỒ THỊ VỀ ĐỒ THỊ ĐẦY ĐỦ

Chuyên ngành: Toán ứng dụng

Mã số: 8 46 01 12

LUẬN VĂN THẠC SĨ TOÁN HỌC

NGƯỜI HƯỚNG DẪN: PGS.TSKH. Phan Thị Hà Dương

Hà Nội - 2019

LỜI CAM ĐOAN

Tôi xin cam đoan những gì viết trong luận văn là do sự tìm tòi, học hỏi của bản thân và sự hướng dẫn tận tình của cô Phan Thị Hà Dương. Mọi kết quả nghiên cứu cũng như ý tưởng của tác giả khác, nếu có đều được trích dẫn cụ thể. Đề tài luận văn này cho đến nay chưa được bảo vệ tại bất kỳ một hội đồng bảo vệ luận văn thạc sĩ nào và cũng chưa hề được công bố trên bất kỳ một phương tiện nào. Tôi xin chịu trách nhiệm về những lời cam đoan trên.

Hà Nội, ngày 1 tháng 12 năm 2018
Người cam đoan

Trần Quang

LỜI CẢM ƠN

Luận văn này được hoàn thành dưới sự hướng dẫn tận tình của cô Phan Thị Hà Dương. Tôi xin bày tỏ lòng biết ơn sâu sắc đến cô, người đã giúp đỡ và chỉ bảo tôi một cách tận tình trong suốt quá trình thực hiện luận văn. Xin cảm ơn các thầy cô các anh chị trong Seminar Cơ sở Toán-Tin đã góp ý và giúp đỡ tôi trong quá trình làm luận văn. Xin cảm ơn các thầy cô, các anh chị ở Học viện Khoa học-Công nghệ nói chung và ở Viện Toán học nói riêng đã tạo điều kiện cho tôi hoàn thành luận văn. Cảm ơn chị Thúy, chị Vân Anh đã hết mình tạo điều kiện và sắp xếp cho tôi thời gian bảo vệ luận văn sớm nhất có thể.

Đặc biệt tôi xin gửi lời cảm ơn chân thành và biết ơn vô tận đối với cha mẹ tôi, những người đã luôn sát cánh và tạo động lực để tôi hoàn thành luận văn này.

Trần Quang

MỤC LỤC

LỜI NÓI ĐẦU

Trong khoa học ngày nay, ngoài việc tìm ra lời giải của một bài toán, thì chúng ta còn chú trọng đến việc cải thiện và phát triển để tạo nên một lời giải hiệu quả, tiết kiệm thời gian giải. Nội dung của luận văn này là đề cập đến các vấn đề về độ phức tạp thuật toán, đặc biệt sẽ tập trung nói về bài toán Clique Editing và một số kết quả đã có.

Bài toán Clique Editing, là bài toán chỉnh sửa đồ thị về đồ thị đầy đủ. Nội dung của bài toán như sau, làm cách nào để biến đổi (bằng cách thêm hoặc bớt cạnh) một đồ thị cho trước thành một đồ thị đầy đủ sao cho số các phép thêm bớt cạnh ấy là ít nhất có thể. Trong thực tế hay trong khoa học thì mô hình về đồ thị đầy đủ khá là phổ biến và thông dụng, đó là lí do tại sao bài toán trên dành được nhiều sự quan tâm.

Những bài toán chỉnh sửa đồ thị được quan tâm từ những năm 80 của thế kỷ XIX, điển hình như Garey, Johnson,...[2] . Bài toán Clique Editing là bài toán thuộc lớp các bài toán chỉnh sửa đồ thị được quan tâm gần đây, nó đã được chứng minh thuộc lớp FPT (Fixed-Parameter Tractable) bởi Flum và Grobe 2006 [3]. Tính NP-đầy đủ của bài toán Clique Editing được nêu ra như một câu hỏi mở bởi Peter Damaschke 2013 và được chứng mình bởi Ivan Kováč 2014 [4].

Chúng tôi sẽ tập trung trình bày về bài toán Clique Editing, và chứng minh tính NP-đầy đủ của bài toán, sau đó sẽ tìm hiểu lớp bài toán FPT và chứng minh bài toán Clique Editing thuộc lớp FPT. Luận văn được chia làm ba chương như sau:

Chương 1: Các khái niệm cơ bản. Trong phần này chúng tôi trình bày một số khái niệm đồ thị và độ phức tạp tính toán về các lớp P, lớp NP, lớp NP-đầy

đủ và một số bài toán nằm trong lớp NP-đầy đủ có liên quan đến bài toán Clique Editing.

Chương 2: Bài toán Clique Editing. Ở chương này, chúng tôi sẽ giới thiệu về bài toán Clique Editing, sau đó nêu một số tính chất liên quan đến bài toán, cuối cùng là chứng minh bài toán Clique Editing thuộc lớp NP-đầy đủ.

Chương 3: Bài toán Clique Editing thuộc lớp FPT. Trong chương này chúng tôi sẽ trình bày định nghĩa lớp FPT và thuật toán FPT, sau đó chứng minh bài toán Clique Editing thuộc lớp FPT.

Mặc dù bản thân tác giả đã hết sức cố gắng, xong luận văn này sẽ không tránh khỏi những thiếu sót. Rất mong nhận được sự góp ý của quý thầy cô và các bạn để luận văn được hoàn thiện hơn.

Trần Quang

DANH MỤC CÁC HÌNH VẼ VÀ ĐỒ THỊ

Chương 1

Các kiến thức cơ bản

Trước khi đi đến vấn đề chính của luận văn thì chúng tôi sẽ trình bày các kiến thức cơ bản của Đồ thị và giới thiệu về bài toán Clique. Và đồng thời sẽ trình bày về độ phức tạp tính toán của thuật toán.

Các khái niệm này đươc tham khảo tại một số tài liệu [1],[7].

1.1 Đồ thị, các khái niệm cơ bản

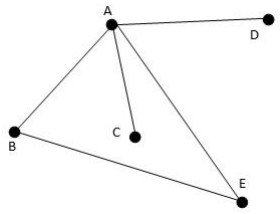
1.1.1 Khái niệm đồ thị

Định nghĩa 1.1.1. *Đồ thị vô hướng hoặc đồ thị G là một cặp không có thứ tự $G := (V, E)$, trong đó*

- *V, tập các đỉnh.*
- *E, tập các cặp đỉnh (không thứ tự), được gọi là cạnh. Hai đỉnh thuộc một cạnh được gọi là các đầu mút của cạnh đó.*

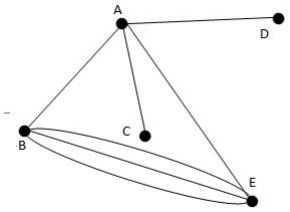
Cạnh của đồ thị mà có 2 điểm đầu mút trùng nhau thì được gọi là khuyên. Đồ thị vô hướng có thể có một hoặc nhiều khuyên. Các cạnh mà có cùng cặp đầu mút thì được gọi là các cạnh song song.

Định nghĩa 1.1.2. *Đơn đồ thị vô hướng là một đồ thị không có khuyên và không có cặp cạnh nào song song.*



Hình 1.1: Đơn đồ thị vô hướng

Định nghĩa 1.1.3. *Đa đồ thị vô hướng là một đồ thị vô hướng mà không phải là đơn đồ thị.*



Hình 1.2: Đa đồ thị vô hướng

Trong luận văn này, chúng tôi chỉ đề cập đến đơn đồ thị vô hướng. Viết "đồ thị $G = (V, E)$", thì có thể hiểu là đơn đồ thị vô hướng $G = (V, E)$. Nhiều khi có thể viết gọn là "đồ thị G", kí hiệu $V(G), E(G)$ lần lượt là tập đỉnh, tập cạnh của đồ thị G.

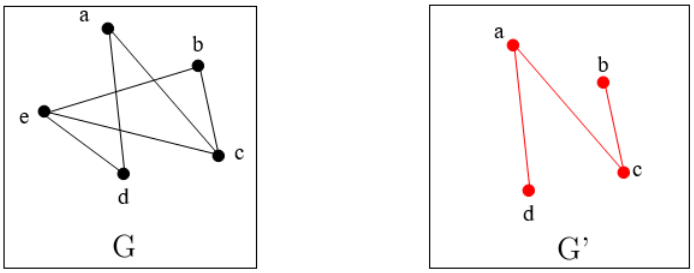
1.1.2 Đồ thị con, đồ thị cảm sinh từ tập đỉnh

Định nghĩa 1.1.4. *Cho đơn đồ thị vô hướng $G = (V, E)$. Khi đó $G' = (V', E')$ được gọi là đồ thị con của G nếu $V' \subset V$ và $E' \subset E$.*

Định nghĩa 1.1.5. *Đồ thị con $G' = (V', E')$ của $G = (V, E)$ được gọi là đồ thị con bao trùm của G nếu $V = V'$.*

Định nghĩa 1.1.6. *Cho đồ thị $G = (V, E)$ và tập đỉnh $V' \subset V$. Đồ thị $G' = (V', E')$ thỏa mãn $E' \subset E$ và E' chứa tất cả các cả các cạnh của E mà*

có hai đầu mút là những đỉnh thuộc V', được gọi là đồ thị con của G cảm sinh bởi tập đỉnh V' hay có thể gọi là đồ thị con cảm sinh bởi G trên tập đỉnh V'. Khi đó G' được ký hiệu là $G' = G[V']$.

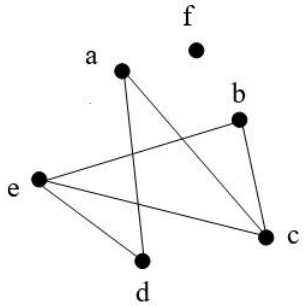


Hình 1.3: Đồ thị G' là đồ thị con của G cảm sinh bởi $\{a, b, c, d\}$

1.1.3 Bậc của đỉnh

Định nghĩa 1.1.7. *Hai đỉnh u và v trong đồ thị vô hướng $G = (V, E)$ được gọi là liền kề nếu $\{u, v\} \in E$. Khi đó $e = \{u, v\}$ gọi là cạnh liên thuộc với các đỉnh u, v. Cạnh e cũng có thể gọi là cạnh nối các đỉnh u, v.*

Định nghĩa 1.1.8. *Bậc của đỉnh v trong đồ thị $G = (V, E)$, ký hiệu $deg(v)$ hay $d_v(G)$, là số cạnh liên thuộc với nó.*



Hình 1.4: $deg(a) = deg(b) = deg(d) = 2$, $deg(c) = deg(e) = 3$, $deg(f) = 0$

Định nghĩa 1.1.9. *Đỉnh v được gọi là đỉnh cô lập nếu $deg(v) = 0$.*

Định lý 1.1.1. *Cho $G = (V, E)$ là một đơn đồ thị vô hướng, khi đó*

$$\sum_{v \in V} deg(v) = 2|E|.$$

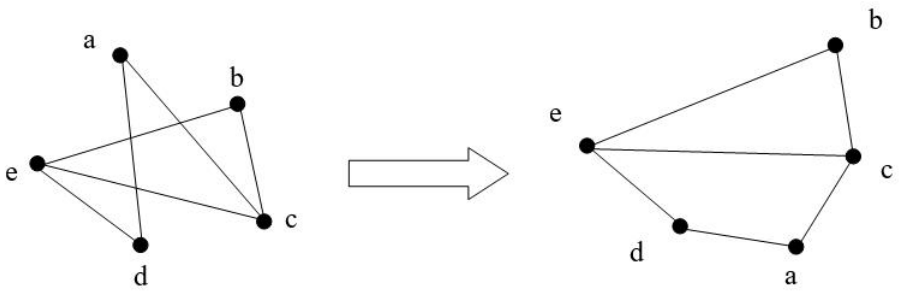
1.1.4 Hành trình

Định nghĩa 1.1.10. *Giả sử $G = (V, E)$ là một đồ thị vô hướng. Một hành trình trong G là một dãy các đỉnh $v_0v_1v_2...v_n$ sao cho với mọi $i = 0, 1, ..., n-1$, $\{v_i, v_{i+1}\}$ là một cạnh của G. Các cạnh $\{v_i, v_{i+1}\}$, $i = 1, 2, ..., n-1$, cũng được gọi là các cạnh của hành trình $v_1v_2...v_n$.*

- *n được gọi là độ dài, v_0 được gọi là đỉnh đầu, v_n được gọi là đỉnh cuối của hành trình nói trên.*
- *Một hành trình được gọi là khép kín nếu đỉnh đầu và đỉnh cuối của nó trùng nhau.*
- *Một hành trình được gọi là đường nếu các đỉnh của hành trình đó đều khác nhau.*
- *Một hành trình được gọi là vết nếu tất cả các cạnh của hành trình đó đều khác nhau.*
- *Một hành trình khép kín được gọi là chu trình nếu nó có độ dài ít nhất là 3 và khi xóa đi đỉnh cuối thì trở thành đường.*
- *Một hành trình khép kín được gọi là mạch nếu các cạnh của hành trình ấy đều khác nhau.*

1.1.5 Đồ thị phẳng

Định nghĩa 1.1.11. *Đồ thị vô hướng $G = (V, E)$ được gọi là đồ thị phẳng nếu nó có thể biểu diễn được ở trên mặt phẳng sao cho các đường cong biểu diễn các cạnh hoặc không giao nhau hoặc chỉ giao nhau ở các đỉnh chung. Biểu diễn nói trên của đồ thị phẳng được gọi là biểu diễn phẳng. Ta sẽ đồng nhất đồ thị phẳng với một biển diễn phẳng của nó.*



Hình 1.5: Đồ thị bên phải là một biểu diễn phẳng của đồ thị bên trái

Định nghĩa 1.1.12. *Độ dài của chu trình ngắn nhất trong một đồ thị được gọi là chu vi nhỏ nhất của đồ thị đó, ký hiệu chu vi nhỏ nhất của đồ thị G là $g(G)$ hay g. Độ dài của chu trình lớn nhất trong một đồ thị được gọi là chu vi lớn nhất của đồ thị đó, ký hiệu chu vi lớn nhất của đồ thị G là $c(G)$ hay c. Đồ thị không có chu trình thì quy ước g và c bằng ∞.*

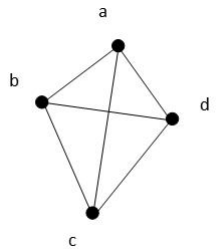
Định lý 1.1.2 (**Công thức Euler**). *Nếu đồ thị phẳng liên thông $G = (V, E)$ có v đỉnh, e cạnh, f miền, thì $v - e + f = 2$.*

Định lý 1.1.3 (**Bất đẳng thức cạnh đỉnh**). *Trong đồ thị phẳng liên thông $G = (V, E)$ bất kỳ với chu vi nhỏ nhất g thỏa mãn $3 \leq g < \infty$, ta luôn có*

$$|E| \leq \frac{g}{g-2}(|V| - 2).$$

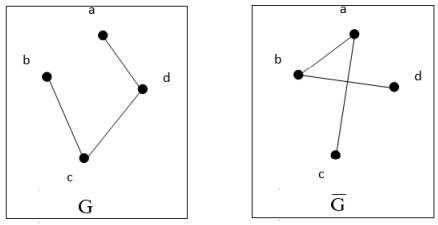
1.1.6 Đồ thị đầy đủ, đồ thị bù, và đồ thị hai phía

Định nghĩa 1.1.13. *Đồ thị $G = (V, E)$ được gọi là đồ thị đầy đủ nếu mọi cặp đỉnh phân biệt trong V đều kề nhau. Với số nguyên dương n, đồ thị đầy đủ n đỉnh có thể được kí hiệu là K_n.*



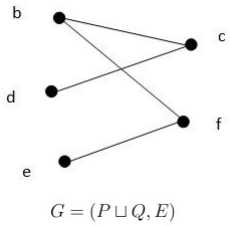
Hình 1.6: Đồ thị đầy đủ K_4

Định nghĩa 1.1.14. *Cho đồ thị $G = (V, E)$ có n đỉnh, đồ thị $G' = (V', E')$ được gọi là đồ thị bù của $G = (V, E)$ nếu $V' = V$ và $E' = E(K_n) \setminus E$. Kí hiệu $G' = \overline{G}$.*



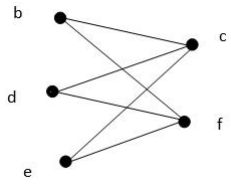
Hình 1.7: Đồ thị bên phải là đồ thị bù của đồ thị bên trái

Định nghĩa 1.1.15. *Đồ thị $G = (V, E)$ được gọi là đồ thị hai phía nếu $V = U \sqcup W$ (kí hiệu "$\sqcup$" nghĩa là hợp rời của 2 tập hợp), trong đó mọi $u_1, u_2 \in U$, $w_1, w_2 \in W$ và $u_1 \neq u_2$, $w_1 \neq w_2$ thì $\{u_1, u_2\} \notin E$, $\{w_1, w_2\} \notin E$. Ta viết đồ thị hai phía này ở dạng $G = (U \sqcup W, E)$.*



Hình 1.8: Đồ thị 2 phía G trong đó $P = \{b, d, e\}, Q = \{c, f\}$

Định nghĩa 1.1.16. *Đồ thị hai phía $G = (U \sqcup W, E)$ được gọi là đồ thị hai phía đầy đủ nếu mỗi đỉnh của U đều kề với tất cả các đỉnh của W. Gọi m, n là số đỉnh của U và V, ta có thể kí hiệu đồ thị hai phía đầy đủ $G = (U \sqcup W, E)$ ở dạng $K_{m,n}$.*

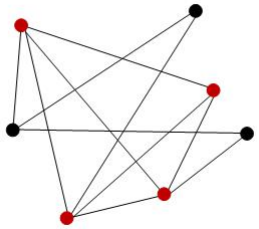


Hình 1.9: Đồ thị 2 phía đầy đủ $K_{3,2}$

1.2 Bài toán Clique

1.2.1 Khái niệm Clique

Định nghĩa 1.2.1. *Clique trong đồ thị $G = (V, E)$ là 1 tập đỉnh $C \subset V$ sao cho $G[C]$ là một đồ thị đầy đủ. Số đỉnh của Clique C được gọi là kích thước của Clique C.*



Hình 1.10: Tập các đỉnh màu đỏ là một Clique của đồ thị có kích thước 4

Định nghĩa 1.2.2. *Clique cực đại của đồ thị $G = (V, E)$ là Clique không được chứa trong bất cứ Clique nào khác nó. Hay nói cách khác là không thể thêm bất kỳ đỉnh nào của G vào Clique cực đại để tạo thành Clique khác.*

Định nghĩa 1.2.3. *Clique lớn nhất của một đồ thị vô hướng G là Clique có số định lớn nhất của G. Chỉ số clique của đồ thị G là số đỉnh của clique lớn nhất của G, ký hiệu $\omega(G)$.*

Chúng ta có thể thấy rằng, trong đồ thị G, Clique lớn nhất là Clique cực đại, nhưng Clique cực đại chưa chắc đã là Clique lớn nhất.

1.2.2 Bài toán Clique

Vấn đề được nãy sinh một cách tự nhiên đó là làm thế nào để tìm ra Clique lớn nhất trong đồ thị G cho trước. Việc tìm kích thước của Clique lớn nhất trong một đồ thị chính là bài toán Clique. Và muốn giải bài toán Clique thì trước hết ta đến với bài toán sau:

Định nghĩa 1.2.4. *Cho đồ thị vô hướng $G = (V, E)$ và một số nguyên dương k. Có hay không một Clique có kích thước k của đồ thị.*

Chúng ta thấy rằng, nếu giải quyết được bài toán trên thì việc giải quyết bài toán Clique hoàn toàn không khó bằng việc cho k tăng dần từ 1 đến giá trị p đầu tiên mà bài toán trên cho ra kết quả là "không". Khi đó $p - 1$ là kết quả của bài toán Clique.

1.3 Độ phức tạp tính toán

1.3.1 Độ phức tạp thuật toán là gì

Định nghĩa 1.3.1. *Độ phức tạp thuật toán (thời gian chạy thuật toán) là một hàm số $f(n)$, trong đó n là kích thước của dữ liệu đầu vào, $f(n)$ chính là số phép toán tối đa mà thuật toán thực hiện trong suốt quá trình chạy của thuật toán.*

Định nghĩa 1.3.2. *Một thuật toán được gọi là có độ phức tạp đa thức, hay còn gọi là có thời gian đa thức, nếu số các phép tính cần thiết khi thực hiện thuật toán không vượt quá $O(n^k)$, với k nguyên dương nào đó, còn n là kích thước của dữ liệu đầu vào.*

1.3.2 Bài toán quyết định

Để giải quyết một bài toán tương đối phức tạp, chúng ta có thể chuyển bài toán ấy về dạng đơn giản hơn nhưng kết quả vẫn tương đương với bài toán ban đầu. Trong phần này chúng tôi sẽ nói về việc chuyển đổi các bài toán về dạng quyết định.

Một bài toán bất kỳ bao giờ cũng có 2 phần chính là *đầu vào (input)* và *đầu ra (output)*.

Định nghĩa 1.3.3. *Bài toán quyết định là bài toán mà đầu ra của nó (output) chỉ nhận kết quả là "yes" hoặc "no".*

Bài toán quyết định thoạt đầu nghe có vẻ không có tính tổng quát cao vì đầu ra chỉ có 1 bit. Tuy nhiên trong thực tế toán học, khi chúng ta tìm ra được lời giải của bài toán quyết định thì cũng thường đồng nghĩa với việc tìm ra lời giải cho bài toán không quyết định ban đầu tương ứng.

Ví dụ 1.3.1. *Bài toán ở định nghĩa* 1.2.4 *chính là dạng quyết định của bài toán Clique.*

1.3.3 Lớp P, lớp NP

Trong thực tế có rất nhiều bài toán có thể được giải trong thời gian đa thức. Nhưng cũng không ít bài toán mà con người chưa tìm ra thuật toán để giải nó trong thời gian đa thức, trong số đó lại có những bài toán mà có thể kiểm tra lời giải trong thời gian đa thức. Dựa vào đó, các bài toán được phân thành 2 lớp như sau:

Lớp P

Định nghĩa 1.3.4. *Một bài toán quyết định được gọi là thuộc lớp P nếu tồn tại một thuật toán thời gian đa thức giải được bài toán ấy.*

Chúng ta có thể xem lớp P là lớp các bài toán đơn giản dễ giải. Có khá nhiều bài toán mà ta đã biết nằm trong lớp P như bài toán sắp xếp một dãy hữu hạn các số, tìm đường đi ngắn nhất trong đồ thị có trọng số, bài toán Euler, nửa Euler,...

Lớp NP

Chúng ta có thể hiểu, lớp NP là lớp các bài toán có thể kiểm tra lời giải trong thời gian đa thức. Sau đây là một định nghĩa tương đối chặt chẽ về lớp NP.

Xét bài toán quyết định $\mathcal{A}$, với mỗi dữ liệu đầu vào x thì đầu ra sẽ là "Yes" nếu x thỏa mãn tính chất $T_{\mathcal{A}}$, là "No" nếu x không thỏa tính chất $T_{\mathcal{A}}$. Đặt $\mathcal{A}_{input}$ là tập tất cả các bộ dữ liệu đầu vào x của bài toán $\mathcal{A}$.

Bài toán 1. *Bài toán* $\mathcal{A}$

Input: $x \in \mathcal{A}_{input}$.

Output: *"Yes" nếu* x *thỏa mãn tính chất* $T_{\mathcal{A}}$.
"No" nếu x *không thỏa tính chất* $T_{\mathcal{A}}$.

Giả sử có bài toán 2 tương đương với bài toán 1 như sau:

Bài toán 2.

Input: $x \in \mathcal{A}_{input}$.

Output: *"Yes" nếu tồn tại* $y(x)$ *để* $(x, y(x))$ *thỏa tính chất* T'.
"No" trong trường hợp còn lại.

"Tương đương" ở đây nghĩa là nếu dữ liệu đầu vào x tương ứng với đầu ra "Yes" ở bài toán 1 khi và chỉ khi x tương ứng với đầu ra "Yes" ở bài toán 2.

Bây giờ chúng ta xét bài toán 3 như sau:

Bài toán 3.

Input: $x \in \mathcal{A}_{input}$, $y(x)$.

Output: *"Yes" nếu* $(x, y(x))$ *thỏa mãn tính chất* T'.
"No" trong trường hợp còn lại.

Ta gọi M là một "Thuật toán kiểm chứng" của bài toán $\mathcal{A}$ nếu M là thuật toán giải bài toán 3 trong thời gian đa thức.

Định nghĩa 1.3.5. *Bài toán quyết định* $\mathcal{A}$ *được gọi là thuộc lớp NP nếu tồn tại một thuật toán kiểm chứng của* $\mathcal{A}$.

Ví dụ 1.3.2. *Về bài toán Clique, ở đây ta xét dạng quyết định của nó.*

Input: $G = (V, E)$, $k \in \mathbb{N}$.

Output: *"Yes" nếu tồn tại một Clique* C *của* G *và* $|C| \geq k$.
"No" trong trường hợp còn lại.

Để chứng tỏ bài toán này thuộc lớp NP ta xét bài toán sau:

Input: $G = (V, E)$, $k \in \mathbb{N}$, $C \subset V$.

Output: *"Yes" nếu* C *là một Clique của* G *và* $|C| \geq k$.
"No" trong trường hợp còn lại.

Thuật toán M để giải bài toán thứ hai này là trước hết kiểm tra số phần tử của C, nếu $|C| < k$ thì cho kết quả "No". Nếu $|C| \geq k$ thì làm bước tiếp theo, duyệt từng cặp đỉnh u, v của C, nếu $\{u, v\} \in E$ thì duyệt tiếp cho tới khi hết các cặp đỉnh của C và cho ra kết quả "Yes", ngược lại phép duyệt sẽ dừng và cho kết quả "No" nếu $\{u, v\} \notin E$.
Thuật toán M này được thực hiện với thời gian tối đa là n^2, trong đó n là số đỉnh của đồ thị.

Như vậy theo định nghĩa thì bài toán *Clique* thuộc lớp NP.

Tóm lại lớp P bao gồm những bài toán dễ giải, nghĩa là có thể được giải quyết trong thời gian đa thức. Còn lớp NP bao gồm những bài toán dễ kiểm tra lời giải. Về mặt trực quan ta có thể thấy bài toán dễ giải quyết thì lời giải của nó cũng sẽ dễ kiểm tra, do đó tập các bài toán của lớp NP đã bao hàm cả những bài toán thuộc lớp P, nghĩa là P $\subset$ NP.
Câu hỏi đặt ra rằng, liệu những bài toán dễ kiểm tra lời giải thì có dễ giải không, hay nói cách khác liệu "P = NP?". Câu hỏi này cho đến nay vẫn chưa có câu trả lời và là một trong những vấn đề quan trọng nhất của Tin học. Người ta tin rằng P $\neq$ NP.

Để phân loại các bài toán thì việc sắp xếp các bài toán từ dễ đến khó dần là một suy nghĩ hoàn toàn tự nhiên. Trong đó lớp P là lớp bài toán được xem là dễ nhất, bao gồm những bài toán mà tồn tại thuật toán giải nó trong thời gian đa thức. Lớp NP là lớp bài toán mà ta hiểu rằng lời giải của nó có thể được kiểm chứng trong thời gian đa thức, và chúng ta đã biết lớp P được chứa trong lớp NP, và người ta tin rằng lớp P là con thực sự của NP. Có nhiều bài toán mà cho đến nay người ta vẫn chưa biết liệu có hay không một thuật toán giải nó trong thời gian đa thức. Mặc dù có thể chúng ta chưa tìm ra câu trả lời ấy nhưng có một công cụ cũng như thước đo để so sánh bài toán này khó hơn hay dễ hơn bài toán kia, đó là phép Quy dẫn.

1.3.4 Quy dẫn

Cho một bài toán quyết định $\mathcal{A}$, một đầu vào x của bài toán $\mathcal{A}$ được gọi là thỏa mãn $\mathcal{A}$ nếu đầu ra tương ứng với x là "Yes". Còn x được gọi là không thỏa mãn $\mathcal{A}$ nếu đầu ra tương ứng với x là "*No*".

Định nghĩa 1.3.6. *(Quy dẫn Karp) Một bài toán $\mathcal{A}$ được gọi là quy được về*

bài toán $\mathcal{B}$ trong thời gian đa thức nếu tồn tại một thuật toán M:

$$\begin{aligned} M : \mathcal{A}_{input} &\longrightarrow \mathcal{B}_{input} \\ x &\longmapsto M(x) \end{aligned}$$

Thỏa mãn:

+ M biến x thành $M(x)$ trong thời gian đa thức.

+ x thỏa mãn $\mathcal{A}$ khi và chỉ khi $M(x)$ thỏa mãn $\mathcal{B}$.

Ký hiệu: $\mathcal{A} \preccurlyeq_{Karp} \mathcal{B}$, vì trong bài viết này ta chỉ sử dụng quy dẫn Karp nên có thể viết gọn lại là $\mathcal{A} \preccurlyeq \mathcal{B}$.

Richard Karp là người đã đưa ra định nghĩa trên (1972) [5].

Ta có các tính chất quan trọng sau:

1. Nếu $\mathcal{A} \preccurlyeq \mathcal{B}$ và $\mathcal{B} \preccurlyeq \mathcal{C}$ thì $\mathcal{A} \preccurlyeq \mathcal{C}$.
2. Nếu $\mathcal{A} \preccurlyeq \mathcal{B}$ và $\mathcal{B} \in P$ thì $\mathcal{A} \in P$.

1.3.5 NP-khó, NP-đầy đủ

Định nghĩa 1.3.7. *Một bài toán quyết định $\mathcal{A}$ được gọi là thuộc lớp NP-khó nếu với mọi bài toán $\mathcal{B} \in NP$, ta có $\mathcal{B} \preccurlyeq \mathcal{A}$.*

Định nghĩa 1.3.8. *Một bài toán quyết định $\mathcal{A}$ được gọi là thuộc lớp NP-đầy đủ nếu $\mathcal{A} \in NP$ và $\mathcal{A} \in$ NP-khó. Hay:*

$$\textit{NP-đầy đủ} = \textit{NP-khó} \cap NP$$

Nếu $\mathcal{A} \preccurlyeq \mathcal{B}$ thì ta nói $\mathcal{B}$ khó hơn $\mathcal{A}$, nghĩa là nếu giải được $\mathcal{B}$ trong thời gian đa thức thì cũng sẽ giải được $\mathcal{A}$ trong thời gian đa thức.

Nếu $\mathcal{A} \preccurlyeq \mathcal{B}$ và $\mathcal{B} \preccurlyeq \mathcal{A}$ thì ta nói $\mathcal{A}$ và $\mathcal{B}$ tương đương với nhau, nghĩa là giải được $\mathcal{A}$ trong thời gian đa thức khi và chỉ khi giải được $\mathcal{B}$ trong thời gian

đa thức.

Như vậy theo định nghĩa chúng ta thấy các bài toán thuộc lớp NP-đầy đủ tương đương với nhau, do đó nếu có một bài toán thuộc lớp NP-đầy đủ nào đó được giải quyết trong thời gian đa thức thì mọi bài toán thuộc lớp NP-đầy đủ cũng sẽ được giải quyết trong thời gian đa thức, và nếu điều đó xảy ra thì $P = NP$. Trên thực tế chúng ta vẫn tin rằng $P \subsetneq NP$.

1.4 Một số bài toán thuộc lớp NP-đầy đủ

Những bài toán thuộc lớp NP-đầy đủ tương đương với nhau, do đó để mô tả lớp NP-đầy đủ, ta cần tìm ra ít nhất một phần tử nào đó thuộc lớp NP-đầy đủ. Một trong những bài toán thuộc lớp NP-đầy đầu tiên được tìm ra là bài toán SAT(Satisfiability). Phần tiếp sau đây, chúng tôi sẽ trình bày một số bài toán NP-đầy đủ, đồng thời những bài toán ấy đều có liên quan đến việc quy dẫn để chứng minh bài toán CLIQUE EDITING (được giới thiệu ở chương sau) là NP-đầy đủ.

1.4.1 Bài toán SAT, 3SAT

Định nghĩa 1.4.1. *Bài toán SAT (Satisfiability):*

Cho một biểu thức logic $\phi(m, n)$ gồm m biến $x_1, x_2, ..., x_m$. $\phi(m, n)$ có dạng là hội của n mệnh đề, mỗi mệnh đề là tuyển của các kí tự phân biệt có dạng x_i hoặc $\bar{x}_i$. Một phép gán của $\phi(m, n)$ là một cách gán cho mỗi biến x_i giá trị True hoặc False. Giá trị của phép gán là giá trị của biểu thức $\phi(m, n)$ khi ta thay giá trị tương ứng của các biến vào $\phi(m, n)$. Có tồn tại hay không một phép gán của $\phi(m, n)$ sao cho giá trị của biểu thức $\phi(m, n)$ là True.

Ví dụ 1.4.1.

$$\phi(4,2) = (x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (\bar{x_2} \vee x_3)$$
$$\phi(2,4) = (x_1 \vee x_2) \wedge (\bar{x_1} \vee x_2) \wedge (x_1 \vee \bar{x_2}) \wedge (\bar{x_1} \vee \bar{x_2})$$

Ở biểu thức $\phi(4,2)$, nếu gán $(x_1, x_2, x_3, x_4) = (true, true, true, true)$ thì ta có kết quả $\phi(4,2) = true$.
Ở biểu thức $\phi(2,4)$, không có phép gán nào để biểu thức đó nhận giá trị true.

Định lý 1.4.1 (Cook-Levin 1971 [6])**.** *SAT(Satisfiability) là bài toán NP-đầy đủ.*

Vì khuôn khổ bài viết nên chúng tôi không trình bày lại chứng minh của định lý này. Bài toán đầu tiên được chứng minh thuộc lớp NP-đầy đủ là *CIRCUITSAT*, một phiên bản dạng mạch điện tử của SAT, bài toán này do Cook-Levin đã chứng minh. Richard Karp chính là người đã chứng minh bài toán SAT mà ta đã định nghĩa ở trên thuộc lớp NP-đầy đủ (1973) .

Sau đây là một dạng đặc biệt của bài toán SAT:

Định nghĩa 1.4.2. *Bài toán 3SAT:*

Cho một biểu thức logic $\phi(m,n)$, trong đó mỗi mệnh đề bao gồm không quá 3 kí tự. Hỏi có tồn tại hay không một phép gán để biểu thức $\phi(m,n)$ nhận giá trị True.

Định lý 1.4.2. *3SAT là bài toán NP-đầy đủ.*

Chứng minh. Xét một biểu thức logic $\phi(m,n)$ là đầu vào của bài toán SAT, bây giờ ta sẽ quy dẫn biểu thức $\phi(m,n)$ về biểu thức $\phi(m',n')$ sao cho $\phi(m',n')$ là một đầu vào của bài toán 3SAT:

Biểu thức $\phi(m,n)$ có n mệnh đề, ta chỉ chuyển đổi các mệnh đề có nhiều hơn 3 kí tự, những mệnh đề còn lại giữ nguyên. Không mất tính tổng quát, gọi $C = (x_1 \vee x_2 \vee ... \vee x_k)$ là một mệnh đề của $\phi(m,n)$ gồm k ký tự ($k \geq 4$). Khi đó ta thêm biến z và thay C thành mệnh đề mới như sau:

$$C' = (x_1 \vee x_2 \vee z) \wedge (\bar{z} \vee x_3 \vee x_4 \vee ... \vee x_k) \tag{1.1}$$

Nhận xét rằng, nếu có một phép gán làm cho C nhận giá trị True thì sẽ tồn tại i ($i \in \{1, 2, ..., k\}$) sao cho x_i được gán giá trị True, nếu $i \in \{3, 4, ..., k\}$ thì ta gán z là True để C' nhận giá trị True, nếu i là 1 hoặc 2 thì ta gán z bởi False để C' nhận giá trị True, điều này nghĩa là nếu có phép gán làm C nhận giá trị True thì sẽ tồn tại phép gán để C' nhận giá trị True. Ngược lại nếu C' nhận giá trị True thì cả hai mệnh đề $(x_1 \vee x_2 \vee z)$ và $(\bar{z} \vee x_3 \vee x_4 \vee ... \vee x_k)$ đều nhận giá trị True, do đó nếu z được gán True thì $\bar{z}$ là False và tồn tại $i \in \{3, 4, ..., k\}$ sao cho x_i là True, nếu z được gán False thì x_1 hoặc x_2 là True, tóm lại nếu C' nhận giá trị True thì sẽ tồn tại $i \in \{1, 2, ..., k\}$ để x_i là True hay nói cách khác C nhận giá trị True. Như vậy tồn tại phép gán để $C = True$ khi và chỉ khi tồn tại phép gán để $C' = True$.

Ta tiếp thực hiện phép biến đổi kiểu (1.1) cho $(\bar{z} \vee x_3 \vee x_4 \vee ... \vee x_k)$, và cứ tiếp tục như vậy cho đến khi thu được biểu thức logic là hội của những mệnh đề có không quá 3 kí tự. Làm điều đó cho những mệnh đề còn lại nhiều hơn 3 ký tự của $\phi(m, n)$. Khi đó ta sẽ thu được biểu thức logic $\phi(m', n') \in 3SAT_{input}$, ta có những nhận xét sau:

1. Phép chuyển đổi từ $\phi(m, n)$ sang $\phi(m', n')$ được thực hiện trong thời gian đa thức.

2. Ta đã chứng minh ở trên, có phép gán làm $C = True$ khi và chỉ khi có phép gán làm $C' = True$. Bằng phương pháp quy nạp với điểm xuất phát đó, ta rút ra $\phi(m, n)$ thỏa mãn SAT khi và chỉ khi $\phi(m', n')$ thỏa mãn 3SAT.

Như vậy $SAT \preccurlyeq 3SAT$.

Việc chứng minh 3SAT thuộc lớp NP không khó, giả sử $\phi(m, n)$ được gán bởi phép gán nào đó, ta duyệt từng ký tự của từng mệnh đề từ trái qua phải, việc duyệt sẽ dừng và trả lời "No" nếu có một mệnh đề mà tất cả các ký tự đều là False. Việc duyệt sẽ dừng và trả lời "Yes" nếu tất cả các mệnh đề đều có ít nhất một kí tự được gán True. Việc duyệt này được thực hiện với thời

gian $o(mn)$.

Như vậy $3SAT \in$ NP-đầy đủ. □

SAT và 3SAT là những bài toán NP-khó đầu tiên được tìm ra, từ đó người ta đã tìm và chứng minh được khá nhiều bài toán thuộc lớp NP-khó.

1.4.2 Bài toán về tập độc lập (IndependentSet)

$I \subset V$ được gọi là tập độc lập của $G = (V, E)$ nếu với mọi cặp đỉnh u, v trong I thì $\{u, v\} \notin E$. Số đỉnh của I được gọi là kích thước của I.
Bài toán INDEPENDENTSET được phát biểu như sau: Cho đồ thị $G = (V, E)$. Tìm tập độc lập có kích thước lớn nhất trong đồ thị G.
Dạng quyết định của bài toán:

Input: $G = (V, E)$, $k \in \mathbb{N}$.

Output: "Yes" nếu tồn tại một tập độc lập X của G và $|X| \geq k$.
"No" trong trường hợp còn lại.

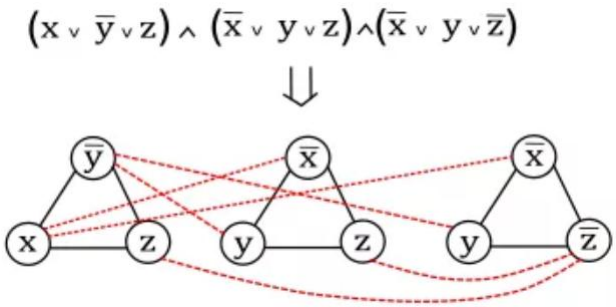
Mệnh đề 1.4.1. *INDEPENDENTSET thuộc lớp NP-đầy đủ.*

Chứng minh. Việc chứng minh bài toán INDEPENDENTSET thuộc lớp NP là không khó (tương tự ví dụ (1.3.2)), chúng tôi không trình bày ở đây. Bây giờ ta sẽ chứng tỏ bài toán này thuộc lớp NP-khó bằng cách quy dẫn từ bài toán 3SAT.

Xét biểu thức logic $\phi(m, n) \in 3SAT_{input}$, ta sẽ xây dựng đồ thị $G = (V, E)$ qua các bước như sau:

B1. Ta viết lại biểu thức $\phi(m, n) = C_1 \vee C_2 \vee ... \vee C_n$, trong đó mỗi mệnh đề C_i ($i \in \{1, 2, ..., n\}$) có không quá 3 kí tự. Sau đó xây dựng n đồ thị con đầy đủ $G_1, G_2, ..., G_n$ sao cho G_i có số đỉnh bằng số kí tự của mệnh đề C_i, nhãn các đỉnh của G_i được dán bởi tên các ký tự của mệnh đề C_i.

B2. Nối các đồ thị $G_1, G_2, ..., G_n$ lại bằng cách nối các cặp đỉnh có dạng x_i và $\bar{x_i}$ lại với nhau. Từ đó ta thu được đồ thị G cần xây dựng.



Hình 1.11: Ảnh mô tả phép quy dẫn từ Clique sang Independent Set

Sau cùng ta chọn $k = n$, như vậy ta đã thực hiện xong phép quy dẫn từ $\phi(m, n) \in 3SAT_{input}$ sang $(G, n) \in INDEPENDENTSET_{input}$ trong thời gian đa thức.

Bây giờ ta sẽ chứng minh rằng $\phi(m, n)$ thỏa mãn 3SAT khi và chỉ khi (G, n) thỏa mãn INDEPENDENTSET. Thật vậy:

Giả sử $\phi(m, n)$ thỏa mãn 3SAT, khi đó tồn tại một phép gán các biến $x_1, x_2, ..., x_m$ để $\phi(m, n)$ nhận giá trị True, tức là với phép gán ấy thì tất cả các mệnh đề $C_1, C_2, ..., C_n$ đều nhận giá trị True, do đó ở mỗi mệnh đề $C_i (i = \overline{1, n})$ luôn tồn tại một kí tự z_i nhận giá trị True. Với mỗi i như vậy, từ tập đỉnh V_{G_i} ta chọn một đỉnh có nhãn là z_i cho vào tập X, từ đó ta thu được tập đỉnh $X = \{z_1, z_2, ..., z_n\}$, từ cách xây dựng đồ thị G thì ta dễ thấy X chính là tập độc lập của G. Cho nên (G, n) thỏa mãn INDEPENDENTSET.

Giả sử (G, n) thỏa mãn INDEPENDENTSET, khi đó tồn tại tập độc lập X của G và X có n đỉnh. Ta có nhận xét sau: 2 đỉnh bất kỳ của X không thể cùng thuộc V_{G_i} nào đó tức là nhãn của 2 đỉnh ấy không cùng thuộc một mệnh đề C_i $i \in \{1, 2, ..., n\}$; hơn nữa theo cách xây dựng G thì với mọi i, X không thể chứa đồng thời 2 đỉnh có nhãn x_i và $\bar{x_i}$. Từ 2 nhận xét vừa rồi ta có thể gán giá trị True, False vào các biến x_i của $\phi(m, n)$ sao cho các kí tự

nhãn của những đỉnh trong X đều mang giá trị True, do đó với mỗi tập đỉnh V_{G_i} đều có một đỉnh có nhãn mang giá trị True, hay nói cách khác thì mỗi mệnh đề C_i của $\phi(m,n)$ có một kí tự mang giá trị True, suy ra với cách gấn ấy ta được $\phi(m,n)$ mang giá trị True.
Như vậy $3SAT \preccurlyeq INDEPENDENTSET$, và INDEPENDENTSET thuộc lớp NP-đầy đủ. □

1.4.3 Bài toán Clique

Bài toán CLIQUE được phát biểu như sau: *Cho đồ thị $G = (V, E)$ tìm Clique có kích thước lớn nhất trong đồ thị G.*

Dạng quyết định của bài toán:

Input: $G = (V, E)$, $k \in \mathbb{N}$.

Output: *"Yes" nếu tồn tại một Clique có không ít hơn k đỉnh.*
"No" trong trường hợp còn lại.

Mệnh đề 1.4.2. *CLIQUE là bài toán NP-dầy đủ.*

Chứng minh. Dễ dàng chứng tỏ $Clique \in NP$, thật vậy ta lấy tập con X bất kỳ của V, trước hết kiểm tra số phần tử của X, trong trường hợp X có không dưới k phần tử thì ta thực hiện việc duyệt từng cặp đỉnh của X, việc duyệt sẽ dừng và trả lời Yes nếu tất cả các cặp đỉnh đều là 2 đầu mút của một cạnh của G, việc duyệt sẽ dừng và trả lời No nếu có 2 đỉnh nào đó của X mà không là 2 đầu mút của bất kỳ cạnh nào của G. Việc kiểm duyệt từng cặp đỉnh này mất thời gian tối đa là $|V|^2$.

Bây giờ ta sẽ chứng minh CLIQUE là NP-khó bằng việc quy dẫn từ bài toán INDEPENDENTSET, Thật vậy:

Xét phép chuyển đổi đơn giản biến $(G, k) \in INDEPENDENTSET_{input}$ thành $(\overline{G}, k) \in CLIQUE_{input}$. Giả sử (G, k) thỏa mãn INDEPENDENTSET,

suy ra tồn tại tập độc lập X của G có không dưới k phần tử, khi đó X chính là Clique trong $\overline{G}$ và X có không dưới k phần tử. Giả sử ngược lại $\overline{G}$ thỏa mãn CLIQUE, suy ra tồn tại $X \subset V_G$ có không dưới k đỉnh, khi đó X chính là tập độc lập của G.

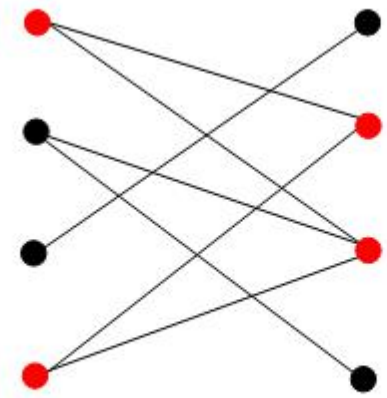
Như vậy CLIQUE là bài toán NP-đầy đủ. □

1.4.4 Bài toán Balanced Complete Biartite Subgraph (BCBS)

BCBS là một bài toán hay, điển hình về tính NP-đầy đủ của đồ thị 2 phía. Bài toán này dược David S. Johnson chứng minh và đưa vào bộ sưu tập những bài toán NP-đầy đủ của mình (1987) [4].
Bài toán được phát biểu như sau: *Đồ thị hai phía* $G = (V_1 \sqcup V_2, E)$ *và* $k \in \mathbb{N}^*$. *Có hay không một đồ thị con* 2 *phía đầy đủ* $K_{k,k}$ *của* G.

Input: *Đồ thị hai phía* $G = (V_1 \sqcup V_2, E)$, $k \in \mathbb{N}^*$.

Output: *"Yes" Nếu tồn tại một đồ thị con hai phía* $K_{k,k}$ *của* G.
"No" trong trường hợp còn lại.



Hình 1.12: Đồ thị con $K_{2,2}$

Mệnh đề 1.4.3. *BCBS là bài toán NP-đầy đủ.*

Chứng minh. Trước hết, để kiểm tra BCBS thuộc lớp NP là hoàn toàn không khó. nên chúng tôi sẽ không trình bày ở đây.

Bây giờ chúng ta sẽ chứng minh BCBS thuộc lớp NP-khó bằng việc quy dẫn từ bài toán CLIQUE ở trên.

Xét đồ thị $G = (V, E)$ và số k là đầu vào của bài toán Clique, không mất tính tổng quát giả sử $k = \frac{|V|}{2}$, ta xây dựng đồ thị hai phía $G' = (V_1' \sqcup V_2', E')$ và số tự nhiên k' thuộc đầu vào của BCBS như sau:

$V_1' = E$

$V_2' = V \cup W$ trong đó W là tập gồm $\dfrac{k(k-1)}{2} - k$ phần tử mới.

$E' = \{\{e, w\} : e \in E, w \in W\} \cup \{\{e, v\} : e \in E$, v không phải là đầu mút của e trong G$\}$

Chọn $k' = \dfrac{k(k-1)}{2}$

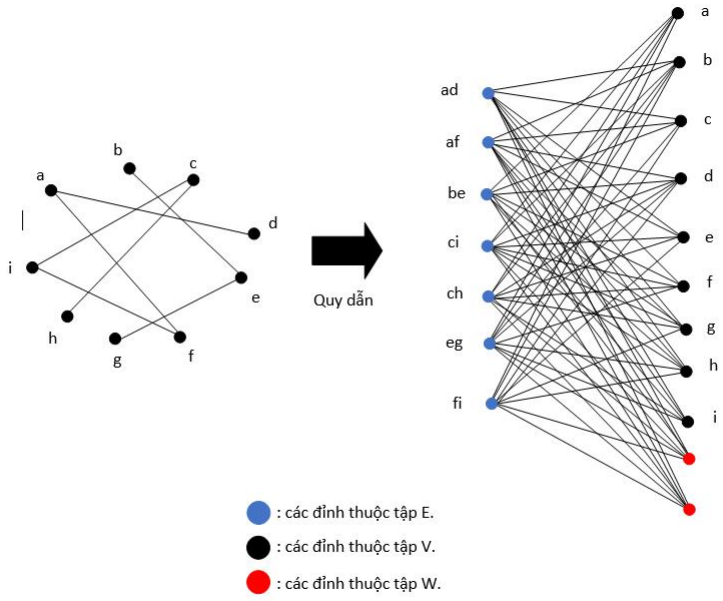
Phép chuyển đổi này được thực hiện hoàn toàn trong thời gian đa thức.

Giả sử tồn tại Clique C của G và có kích thước k, đồ thị cảm sinh của C trong G là $G[C] = (V_C, E_C)$, với $|V_C| = k, |E_C| = \dfrac{k(k-1)}{2} = k'$. Khi đó $V \setminus V_C$ là tập đỉnh không phải là đầu mút của bất cứ cạnh nào của E_C, do trong G' thì đồ thị con cảm sinh từ $(V \setminus V_C) \sqcup E_C$ là đồ thị hai phía đầy đủ. Hơn nữa theo các xây dựng tập cạnh của G' thì đồ thị con hai phía cảm sinh bởi $(V \setminus V_C \cup W) \sqcup E_C$ là đồ thị con hai phía đầy đủ $K_{k',k'}$, trong đó $|(V \setminus V_C \cup W)| = k'$ và $|E_C| = k'$.

Ngược lại giả sử tồn tại đồ thị con hai phía đầy đủ $K_{k',k'}$ của G', gọi đồ thị ấy là $G'[E_A \sqcup V_A]$, trong đó $E_A \subset E, V_A \subset (W \cup V)$, theo cách xây dựng tập cạnh của G' thì ta có thể chọn $V_A = W \cup V_A'$, với $|V_A'| = k' - |W| = k$,

khi đó V'_A bao gồm các đỉnh mà không phải là đầu mút của bất cứ cạnh nào của E_A, suy ra tập hợp những đầu mút của E_A sẽ được chứa trong $V \setminus V'_A$. Vì $|V \setminus V'_A| = |V| - |V'_A| = k$ và $|E_A| = \dfrac{k(k-1)}{2}$ nên $V \setminus V'_A$ chính là Clique của G.

Như vậy $CLIQUE \preccurlyeq BCBS$ và BCBS là bài toán NP-đầy đủ. □



Hình 1.13: Quy dẫn từ Clique sang BCBS

Như vậy chúng ta thấy việc chứng minh các bài toán thuộc lớp NP-khó là hoàn toàn không dễ. Muốn chứng tỏ một bài toán thuộc lớp NP-khó thì ta cần tìm một bài toán NP-khó nào đó để quy dẫn về nó. Khi ta tìm thấy thêm một bài toán NP-khó thì cũng đồng nghĩa với việc có thêm một công cụ mới cho việc tìm những bài toán NP-khó khác. Do đó việc sưu tập những bài toán NP-khó là cần thiết và quan trọng, điều này đã được một số nhà khoa học quan tâm và làm từ những năm 1974 trở đi.

Chương 2

Bài toán Clique Editing

Những bài toán chỉnh sửa đồ thị là một trong những vấn đề cổ điển trong lý thuyết đồ thị và khoa học máy tính. Các bài toán đó có nội dung như sau, cho đồ thị G bất kỳ và một lớp đồ thị $\mathcal{G}$ có tính chất T nào đó, câu hỏi đặt ra là với số lượng chỉnh sửa ít nhất là bao nhiêu để biến đồ thị G thành đồ thị G^* mới thuộc vào lớp $\mathcal{G}$, những chỉnh sửa ấy bao gồm việc thêm hoặc bớt cạnh, thêm đỉnh, bớt đỉnh... . Các bài toán chỉnh sửa đồ thị có ứng dụng quan trọng trong nhiều lĩnh vực như Sinh học phân tử, Đại số, thiết kế vi mạch..., vấn đề này đã được quan tâm vào những năm 80 của thế kỷ 19, cho đến nay đã có rất nhiều bài báo và những kết quả có liên quan được công bố bởi các nhà khoa học. Trong chương này, chúng tôi sẽ trình bày về bài toán Clique Editing, trước tiên là giới thiệu bài toán, sau đó tìm hiểu một số tính chất của bài toán trên đồ thi tổng quát, đồ thị 2 phía, đồ thị phẳng, việc chính là trình bày cách quy dẫn và chứng minh bài toán Clique Editing thuộc vào lớp NP-đầy đủ. Tuy nhiên trên đồ thị phẳng thì bài toán thuộc lớp P. Chương này được tham khảo tại [4].

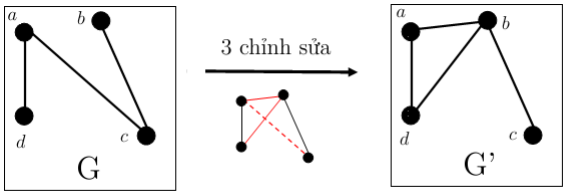
2.1 Giới thiệu bài toán Clique Editing

2.1.1 Bài toán chỉnh sửa (thêm, bớt cạnh) và Clique

Có nhiều hình thức để chỉnh sửa đồ thị tùy vào nhu cầu của chúng ta. Tuy nhiên chúng ta quy ước kể từ đây cho đến hết luận văn này, khi nhắc đến hoạt động chỉnh sửa thì chúng ta hiểu đó là hoạt động thêm bớt cạnh.

Định nghĩa 2.1.1. *Cho đồ thị $G = (V, E)$, với cặp đỉnh u, v nào đó của G*

- *Nếu u, v không kề nhau mà ta thêm $\{u, v\}$ vào E thì đây được gọi là một hoạt động thêm cạnh, và được tính là một hoạt động chỉnh sửa trên G hay cụ thể hơn là hoạt động chỉnh sửa giữa u và v.*
- *Nếu u, v kề nhau mà ta bỏ đi cạnh $\{u, v\}$ ra khỏi E thì đây được gọi là một hoạt động bớt cạnh, và được tính là một hoạt động chỉnh sửa trên G hay cụ thể hơn là hoạt động chỉnh sửa giữa u và v.*
- *Nếu ta không thêm hoặc bớt cạnh giữa u, v thì không được tính là một hoạt động chỉnh sửa.*



Hình 2.1: G được chỉnh sửa thành G' bởi việc thêm và bớt cạnh

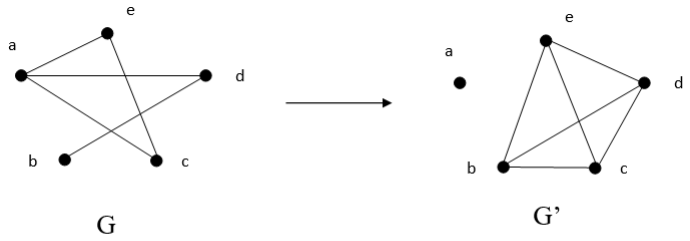
Ở hình trên $G = (V, E)$, $V = \{a, b, c, d\}$ và $E = \{\{a, d\}, \{a, c\}, \{b, c\}\}$. Ta đã thực hiện 3 phép chỉnh sửa đối với G để thành đồ thị $G' = (V', E')$. Cụ thể đồ thị G đã được bỏ đi cạnh $\{a, c\}$ và thêm vào 2 cạnh $\{a, b\}, \{b, d\}$.

Mệnh đề 2.1.1. *Cho đồ thị $G = (V, E)$ và tập đỉnh $C \subset V$. Số phép chỉnh sửa tối thiểu trên đồ thị G để được đồ thị mới $G' = (C \sqcup I, E')$ sao cho C là một Clique của G' và I là tập các đỉnh cô lập là:*

$$|E(\overline{G}[C])| + |\{\{u, v\} \in E(G) : u \in V \setminus C \vee v \in V \setminus C\}|$$

Biểu thức trên được kí hiệu là $cost_G(C)$, nếu không có gì nhầm lẫn thì ta có thể viết $cost(C)$.

Chứng minh. Để số chỉnh sửa là ít nhất thì ứng với mỗi cặp đỉnh u, v của G, ta thực hiện tối đa một lần chỉnh sửa giữa u và v.
Từ G, để tạo thành G' như thế ta cần thêm các cạnh vào $G[C]$ để $G[C]$ trở thành một đồ thị đầy đủ, đồng thời bỏ đi tất cả các cạnh của G mà có đầu mút là đỉnh trong $V \setminus C$.
Như thế ta có điều phải chứng minh. □



Hình 2.2: $C = \{b, c, d, e\}$ và $cost_G(C) = 7$

2.1.2 Bài toán Clique Editing là gì

Bây giờ chúng ta xét lớp đồ thị $\mathcal{G}$ bao gồm các đồ thị có dạng $G' = (C \sqcup I, E')$ trong đó C là một Clique của G' và I là tập các đỉnh cô lập. Mục tiêu của chúng ta là chỉnh sửa một đồ thị G bất kỳ thành một đồ thị G' thuộc lớp $\mathcal{G}$.

Định nghĩa 2.1.2 (**Bài toán Clique Editing**). *Cho đồ thị $G = (V, E)$, tìm số chỉnh sửa tối thiểu trên G, để được một đồ thị $G' \in \mathcal{G}$.*

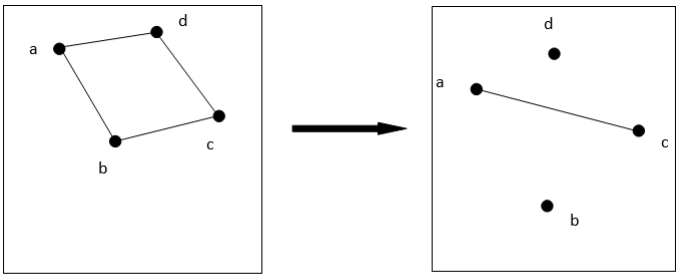
Input: Đồ thị $G = (V, E)$.

Output: Tính $\min\{cost(C) : C \subset V\}$.

Ví dụ 2.1.1. *Cho đồ thị* G *gồm* 4 *đỉnh* a, b, c, d *và* 4 *cạnh* $\{a, b\}, \{b, c\}, \{c, d\}, \{d, a\}$. *Tìm số chỉnh sửa tối thiểu của* G *để được đồ thị* $G' \in \mathcal{G}$.

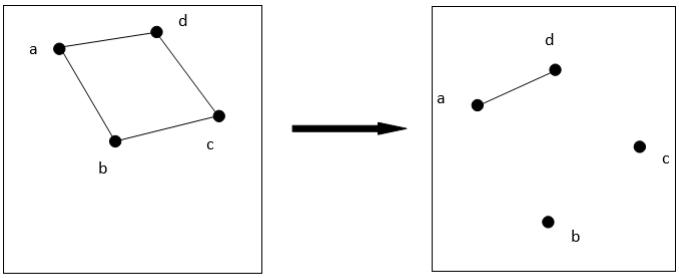
+ Nếu tập $C \in V(G)$ có đúng 1 phần tử thì $cost(C) = 4$.

+ $cost(\{a, c\}) = cost(\{b, d\}) = 5$



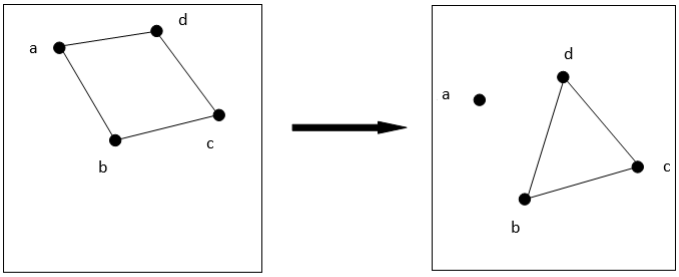
G $\quad$ G_2

+ $cost(\{a, b\}) = cost(\{b, c\}) = cost(\{c, d\}) = cost(\{d, a\}) = 3$



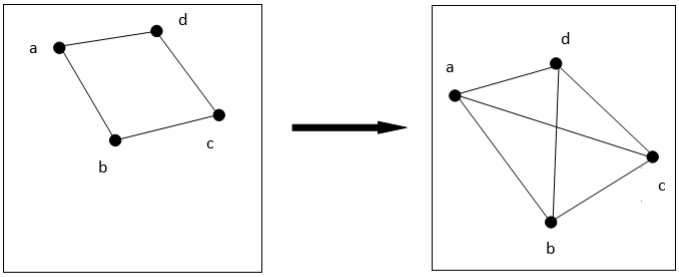
G $\quad$ G_3

+ $cost(\{a,b,c\}) = cost(\{a,b,d\}) = cost(\{a,c,d\}) = cost(\{b,c,d\}) = 3$



G G_4

+ $cost(\{a,b,c,d\}) = 2$



G G_5

Vậy số chỉnh sửa tối thiểu của để biến đổi G thành $G' \in \mathcal{G}$ là 2.

Chúng ta thấy rằng số tập con C của V là hữu hạn nên luôn tồn tại giá trị nhỏ nhất của $cost(C)$.
Kí hiệu: C_{OPT} là "nghiệm Clique tối ưu" của bài toán Clique Editing trên đồ thị G, nghĩa là $C_{OPT} \subset V$ là một trong những tập đỉnh mà $cost(C)$ đạt giá trị nhỏ nhất tại đó.

Định nghĩa 2.1.3. *Dạng quyết định của bài toán Clique Editing*

Input: $G = (V, E)$ *và* $k \in \mathbb{N}$.

Output: *"Yes" nếu tồn tại* $C \subset V$ *để* $cost(C) \leq k$.
"No" trong trường hợp còn lại.

Ta sẽ tập trung xét dạng quyết định của bài toán Clique Editing. Khi nói (G, k) là bộ dữ liệu đầu vào của bài toán Clique Editing thì ta hiểu đây là một bộ dữ liệu đầu vào gồm đồ thị $G = (V, E)$ và số tự nhiên k. Khi ta nói đồ thị G_0 là nghiệm của bài toán Clique Editing với đầu vào (G, k) thì ta hiểu G_0 chính là kết quả của việc chỉnh sửa đồ thị G bởi tối đa k phép chỉnh sửa, và G_0 là đồ thị như mong muốn đó là tập đỉnh của G' là hợp giữa một Clique và một tập đỉnh cô lập.

2.2 Bài toán Clique Editing là bài toán NP-đầy đủ

2.2.1 Các tính chất liên quan bài toán Clique Editing

Mệnh đề 2.2.1. *Cho đồ thị* $G = (V, E)$, *gọi* C_{OPT} *là nghiệm Clique tối ưu của bài toán CLIQUE EDITING trên* G. *Khi đó với mỗi đỉnh* $v \in C_{OPT}$ *thì* $|E_v(G[C_{OPT}])| \geq |E_v(\overline{G}[C_{OPT}])|$, *hơn nữa* $|E_v(G[C_{OPT}])| \geq \dfrac{|C_{OPT}| - 1}{2}$.

Chứng minh. Vì C_{OPT} là nghiệm tối ưu của G nên $cost(C_{OPT}) \leq cost(C_{OPT} \setminus \{v\})$, hay $|E(\overline{G}[C_{OPT}])| + |\{\{u, w\} \in E(G) : u \in V \setminus C_{OPT} \vee w \in V \setminus C_{OPT}\}| \leq |E(\overline{G}[C_{OPT} \setminus \{v\}])| + |\{\{u, w\} \in E(G) : u \in V \setminus (C_{OPT} \setminus \{v\}) \vee w \in V \setminus (C_{OPT} \setminus \{v\})\}|$, từ đây suy ra $|E(\overline{G}[C_{OPT}])| - |E(\overline{G}[C_{OPT} \setminus \{v\}])| \leq |E_v(G[C_{OPT}])|$, do đó $|E_v(\overline{G}[C_{OPT}])| \leq |E_v(G[C_{OPT}])|$.
Từ đó ta suy ra $|E_v(G[C_{OPT}])| \geq \dfrac{|E_v(\overline{G}[C_{OPT}])| + |E_v(G[C_{OPT}])|}{2}$ hay
$|E_v(G[C_{OPT}])| \geq \dfrac{|C_{OPT}| - 1}{2}$. □

Mệnh đề 2.2.2. *Cho* $G = (P \sqcup Q, E)$ *là đồ thị hai phía. Trong đó* C_{OPT} *chứa* p *đỉnh của* P *và* q *đỉnh của* Q. *Giả sử* $p \leq q$, *khi đó* $p \geq q - 1$.

Chứng minh. Gọi v là một đỉnh của C_{OPT} và $v \in Q$. Khi đó $|E_v(G[C_{OPT}])| \geq |E_v(\overline{G}[C_{OPT}])|$, mà ta lại thấy rằng $p \geq |E_v(G[C_{OPT}])|$ và $|E_v(\overline{G}[C_{OPT}])| \geq q - 1$ cho nên $p \geq q - 1$. □

Hệ quả 2.2.1. *Cho đồ thị hai phía $G = (P \sqcup Q, E)$. Khi đó C_{OPT} chứa p đỉnh của nhánh này và p đỉnh của nhánh còn lại hoặc C_{OPT} chứa p đỉnh của nhánh này và $p + 1$ đỉnh của nhánh còn lại, với p là số tự nhiên nào đó.*

Mệnh đề 2.2.3. *Cho đồ thị hai phía $G = (P \sqcup Q, E)$. Luồn tồn tại nghiệm tối ưu C_{OPT} sao cho $G[C_{OPT}]$ có dạng $K_{k,k}$ hoặc $K_{k,k+1}$ (k là số tự nhiên nào đó).*

Chứng minh. Xét C_{OPT} chứa p đỉnh thuộc P và q đỉnh thuộc Q ($p \leq q$). Giả sử tồn tại $u \in P$ và $v \in Q$ sao cho $\{u, v\} \notin E$, khi đó

$$p - 1 \geq E_v(G_[C_{OPT}]) \geq E_v(\overline{G}_[C_{OPT}]) \geq q$$

Điều này suy ra $p > q$ (mâu thuẫn), do đó $G[C_{OPT}]$ là đồ thị con hai phía đầy đủ $K_{p,q}$. Kết hợp với mệnh đề trên ta suy ra điều phải chứng minh. □

Mệnh đề 2.2.4. *Cho đồ thị hai phía G. Luôn tồn tại một nghiệm tối ưu C_{OPT} sao cho $G[C_{OPT}]$ là $K_{p,p}$, với $p \in \mathbb{N}$*

Chứng minh. Với mệnh đề (2.2.3) trên thì $G[C_{OPT}]$ có dạng $K_{p,p}$ hoặc $K_{p,p+1}$. Nếu $G[C_{OPT}]$ có dạng $K_{p,p}$ thì mệnh đề là đúng. Nếu $G[C_{OPT}]$ có dạng $K_{p,p+1}$ thì ta bỏ đi một phần tử v trong C_{OPT} để $G[C_{OPT} \setminus \{v\}]$ là $K_{p,p}$, mà $|E_v(G[C_{OPT}])| = |E_v(\overline{G}[C_{OPT}])|$ cho nên $cost(G[C_{OPT}]) = cost(G[C_{OPT} \setminus \{v\}])$, do vậy $C_{OPT} \setminus \{v\}$ cũng chính là một nghiệm tối ưu của bài toán và đồ thị cảm sinh của nó trong G chính là $K_{p,p}$.
Như vậy ta luôn tìm được một nghiệm tối ưu của bài toán mà đồ thị cảm sinh của nó có dạng $K_{p,p}$. □

Mệnh đề 2.2.5. *Cho đồ thị hai phía $G = (P \sqcup Q, E)$, với mọi đồ thị con bất kỳ có dạng $K_{p,p}$ của G, gọi C là tập đỉnh của $K_{p,p}$, khi đó*

$$cost_G(C) = |E(G)| - p$$

Chứng minh. Đặt $C = P_0 \sqcup Q_0$ trong đó $P_0 \subset P$, $Q_0 \subset Q$. Vì $G[C] = K_{p,p}$ là đồ thị hai phía đầy đủ, nên để chỉnh sửa G thành đồ thị $G' = (P_0 \sqcup Q_0 \sqcup I, E')$ trong đó $P_0 \sqcup Q_0$ là một Clique và I là tập các đỉnh cô lập, thì cần bỏ đi tất cả các cạnh của G nhưng không nằm trong $G[C]$ đồng thời thêm tất cả các cạnh giữa hai đầu mút bất kỳ trong P_0 và trong Q_0. Khi đó số chỉnh sửa sẽ là:

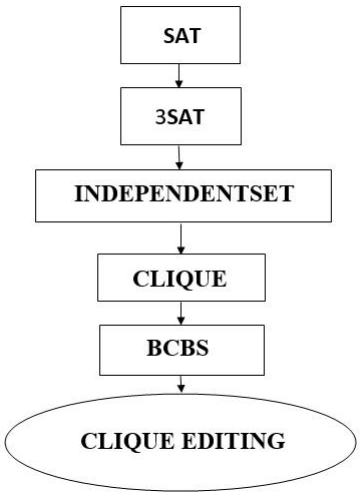
$$|E(G)| - p^2 + 2.\frac{p(p-1)}{2} = |E(G)| - p$$

□

Mệnh đề 2.2.6. *Cho đồ thị hai phía G. Nghiệm C_{OPT} mà $G[C_{OPT}]$ dạng $K_{p,p}$ thì $cost(C_{OPT}) = |E(G)| - p$.*

Chứng minh. Mệnh đề này là hệ quả trực tiếp từ mệnh đề trên. □

2.2.2 Bài toán Clique Editing là NP-đầy đủ



Hình 2.3: Sơ đồ quy dẫn

Ở chương trước ta đã chứng tỏ được bài toán BCBS là NP-đầy đủ. Bây giờ ta sẽ chứng minh bài toán CLIQUE EDITING là NP-đầy đủ bởi việc quy dẫn trực tiếp từ bài toán BCBS.

Định lý 2.2.1. *Bài toán CLIQUE EDITING là bài toán NP-đầy đủ.*

Chứng minh. Chúng ta sẽ thực hiện phép quy dẫn sau:
Gọi x là dữ liệu đầu vào của bài toán BCBS, dữ liệu x cụ thể là: đồ thị hai phía $G = (P \sqcup Q, E)$ và số nguyên dương k_1.
Xét dữ liệu đầu vào $M(x)$ của bài toán CLIQUE EDITING phụ thuộc vào x như sau: đồ thị hai phía $G = (P \sqcup Q, E)$ và số nguyên dương $|E| - k_1$.

Giả sử x thỏa mãn bài toán BCBS, nghĩa là tồn tại đồ thị con hai phía đầy đủ K_{k_1,k_1} của G. Gọi C là tập đỉnh của K_{k_1,k_1} nói trên, theo mệnh đề (2.2.5) thì $cost(C) = |E(G)| - k_1$, suy ra $M(x)$ bao gồm G và số nguyên dương $|E(G)| - k_1$ thỏa mãn CLIQUE EDITING.

Giả sử $M(x)$ thỏa mãn bài toán CLIQUE EDITING, khi đó tồn tại $C \subset V(G)$ sao cho $cost(C) \leq |E(G)| - k_1$. Theo Mệnh đề (2.2.4) thì tồn tại nghiệm tối ưu $C_{OPT} = K_{k_2,k_2}$, nên $cost(C_{OPT}) \leq cost(C) \leq |E(G)| - k_1$, suy ra $|E(G)| - k_2 \leq |E(G)| - k_1$, do đó $k_2 \geq k_1$, mà K_{k_2,k_2} là một đồ thị con đầy đủ của G, nên K_{k_1,k_1} cũng là một đồ thị con đầy đủ của G. Tức là x thỏa mãn bài toán BCBS.

Như vậy có một phép quy dẫn trong thời gian đa thức từ bài toán BCBS sang bài toán CLIQUE EDITING, và đồng thời chúng ta cũng dễ dàng chứng minh được CLIQUE EDITING thuộc lớp NP. Do đó bài toán CLIQUE EDITING là NP-đầy đủ. □

2.3 Bài toán Clique Editing trong đồ thị phẳng

Như chúng ta đã biết, bài toán Clique Editing trên đồ thị tổng quát là bài toán NP-đầy đủ. Tuy nhiên đối với một số lớp đồ thị khá đặc biệt như đồ thị phẳng thì bài toán Clique Editing có thể hoàn toàn được giải quyết trong thời gian đa thức, tức là bài toán Clique Editing trên đồ thị phẳng thuộc lớp P. Chúng ta sẽ đến với 2 mệnh đề sau đây.

Mệnh đề 2.3.1. *Cho C_{OPT} là nghiệm tối ưu của bài toán Clique Editing trên đồ thị phẳng G. Khi đó $cost(C_{OPT}) \leq 11$.*

Chứng minh. Gọi số đỉnh của G là n, số cạnh của G là m, một tính chất cơ bản trong đồ thị phẳng cho ta $m \leq 3n - 6$.
Ta có $\Sigma_{v \in V} deg(v) = 2m \leq 6n - 12$, nên $\dfrac{\Sigma_{v \in V} deg(v)}{n} < 6$, hay nói cách khác là bậc trung bình của các đỉnh trong đồ thị phẳng luôn nhỏ hơn 6.
Do G là đồ thị phẳng nên $G[C_{OPT}]$ cũng là một đồ thị phẳng, cho nên bậc trung bình của $G[C_{OPT}]$ nhỏ hơn 6, do đó tồn tại $v \in C_{OPT}$ để v có bậc nhỏ hơn 6 trong $G[C_{OPT}]$, hay $|E_v(G[C_{OPT}])| \leq 5$. Mặt khác theo mệnh đề (2.1.1) ta có $|E_v(G[C_{OPT}])| \geq \dfrac{|C_{OPT}| - 1}{2}$, từ đó suy ra $\dfrac{|C_{OPT}| - 1}{2} \leq 5$ hay $|C_{OPT}| \leq 11$. □

Mệnh đề 2.3.2. *Bài toán Clique Editing trên đồ thị phẳng G có thể giải được trong thời gian đa thức.*

Chứng minh. Ta xét duyệt tất cả các tập con C của V_G với $|C| \leq 11$. Trong đó tập C nào có giá trị $cost(C)$ nhỏ nhất thì chọn tập ấy làm nghiệm tối ưu. Mà số tập C như vậy là $\Sigma_{i=1}^{11} \binom{n}{i} = O(n^{11})$, nên ta có điều phải chứng minh. □

Như vậy trong chương 2 này ta đã tìm hiểu khái niệm về bài toán chỉnh sửa đồ thị (cụ thể hơn là bài toán Clique Editing). Trong đó bài toán Clique Editing đối với đồ thị 2 phía nói riêng, đồ thị tổng quát nói chung là bài toán NP-đầy đủ. Tuy nhiên đối với lớp đồ thị phẳng thì bài toán Clique Editing là bài toán thuộc lớp P.

Chương 3

Clique Editing là bài toán FPT

Như chúng ta đã biết, bài toán Clique Editing là bài toán NP-đầy đủ, nên việc tìm ra một thuật toán để giải nó trong thời gian đa thức là rất khó hoặc không thể. Tuy nhiên, chúng ta có thể tìm ra những thuật toán với thời gian được cải thiện đáng kể cho một lớp đồ thị đầu vào G nào đó. Ở đây ta muốn nói đó là thuật toán FPT (Fixed Parameter Tractable).

Chương này được tham khảo từ [3]

3.1 Giới thiệu về lớp FPT, bài toán FPT

Ở đây ta chỉ đề cập các bài toán liên quan đến đồ thị mà thuộc lớp NP-khó.

Định nghĩa 3.1.1. *Cho bài toán quyết định $\mathcal{A}$ với đầu vào gồm đồ thị $G = (V, E)$ kích thước n và số nguyên dương k ($1 \leq k \leq n$). Ta nói bài toán $\mathcal{A}$ ấy thuộc lớp FPT nếu tồn tại một thuật toán M giải được bài toán $\mathcal{A}$ trong thời gian $f(k).n^{O(1)}$ (f là hàm chỉ phụ thuộc vào k, $O(1)$ là một hằng số).*
Bài toán $\mathcal{A}$ như thế gọi là bài toán FPT, thuật toán M gọi là thuật toán FPT cho bài toán $\mathcal{A}$.

Chương này chúng tôi sẽ đi chứng minh bài toán Clique Editing là bài toán FPT. Xét đầu vào của bài toán là (G, k), ta tìm được một thuật toán giải bài toán Clique Editing với đầu vào như trên trong thời gian $2^{O(k)} + n^{O(1)}$. Có thể thấy, trong trường hợp k tương đối nhỏ thì bài toán trên được giải

quyết trong thời gian nhỏ đáng kể. Trong thực tế con người cũng có thể trực quan và chọn những đầu vào bao gồm đồ thị tương đối hoàn chỉnh như mong muốn, và k nhỏ.

3.2 Thuật toán Nhân tử hóa

Để tìm một thuật toán FPT cho một bài toán $\mathcal{A}$ với đầu vào (G, k) người ta thường sử dụng phương pháp nhân tử hóa, ý tưởng như sau: biến đổi đầu vào (G, k) thành đầu vào $(G', \phi(k))$ của bài toán $\mathcal{A}$ nhưng trong đó G' là đồ thị có kích thước phụ thuộc vào k, $\phi(k)$ là hàm số phụ thuộc vào k.

3.2.1 Khái niệm

Gọi $\mathcal{A}$ là một bài toán chỉnh sửa đồ thị, $\mathcal{A}_{input}$ là tập hợp các bộ dữ liệu đầu vào mà bài toán $\mathcal{A}$ có thể nhận.

Định nghĩa 3.2.1. *Cho bài toán chỉnh sửa đồ thị $\mathcal{A}$, và $(G, k) \in \mathcal{A}_{input}$ (G có n đỉnh). Thuật toán Nhân tử hóa là một thuật toán M biến (G, k) thành $(G', k') \in \mathcal{A}_{input}$ sao cho:*

- *Độ phức tạp của thuật toán M là $(n + k)^{f(\frac{1}{\epsilon})}$, với hằng số $\epsilon > 0$ và hàm số f phụ thuộc k nào đó.*
- *$n \leq \epsilon.g(k)$ và $k' \leq k$, với g là một hàm số nào đó phụ thuộc k.*
- *(G, k) thỏa mãn bài toán $\mathcal{A}$ khi và chỉ khi (G', k') thỏa mãn bài toán $\mathcal{A}$.*

(G', k') được gọi là một "nhân tử" của (G, k).

Chúng ta thấy rằng n phụ thuộc vào k, do đó trong trường hợp một đồ thị cho trước gần hoàn chỉnh thì số thao tác chỉnh sửa ít đi. Như thế k sẽ bé

và việc giải quyết bài toán sẽ nhanh hơn đáng kể. Trên thực tế chúng ta cũng thường chỉ giải quyết các bài toán chỉnh sửa với hình dạng gần hoàn chỉnh.

3.2.2 Thuật toán Nhân tử hóa cho bài toán Clique Editing

Ý tưởng ban đầu: cho một đồ thị $G = (V, E)$, giả sử chúng ta có một quá trình biến đổi G thành $G_0 \in \mathcal{G}$ như mong muốn (tập đỉnh của G_0 là hợp giữa một Clique và một tập các đỉnh cô lập). Với một đỉnh v nào đó của G, trong quá trình biến đổi từ G sang G_0 chúng ta chỉ xét những chỉnh sửa cạnh có đầu mút v. Khi đó nếu cuối quá trình ta thấy v không còn cạnh nối nào thì rõ ràng v là một đỉnh cô lập trong G_0, còn nếu cuối quá trình ta thấy v được nối với những những đỉnh $v_1, v_2, ..., v_t$ nào đó thì không khó để chúng ta khẳng định rằng $C = \{v, v_1, v_2, v_3, ..., v_t\}$ chính là một Clique trong G_0 và còn $V \setminus C$ là tập những đỉnh cô lập. Như vậy, ý tưởng chính ở đây là xét các trường hợp chỉnh sửa cạnh tại mỗi đỉnh trong G. Sau đây là việc xây dựng thuật toán một cách cụ thể.

Xây dựng quy tắc 1

Cho bài toán chỉnh sửa $\mathcal{A}$ và bộ dữ liệu đầu vào (G, k), giả sử thông qua tối đa k phép chỉnh sửa ta biến đổi G thành G_0 như mong muốn (tức G_0 là nghiệm của bài toán $\mathcal{A}$ đối với bộ dữ liệu đầu vào (G, k)). Với đồ thị G ta đặt $|V(G)| = n_G = n, |E(G)| = m_G = m$.

Định nghĩa 3.2.2. *Bài toán $\mathcal{A}$ được gọi là "có thể xây dựng từ lân cận" trong thời gian $p(n)$ (p là đa thức), nếu cho trước đỉnh v trong G và tập hợp không rỗng tất cả các lân cận của v trong nghiệm G_0, thì ta có thể giải được bài toán $\mathcal{A}$ đối với bộ đầu vào (G, k) trong thời gian $p(n)$.*

Mệnh đề 3.2.1. *Bài toán Clique Editing là "có thể xây dựng từ lân cận" trong thời gian tuyến tính theo số cạnh của đồ thị.*

Chứng minh. Gọi (G, k) là bộ dữ liệu đầu vào của bài toán Clique Editing, giả sử thông qua tối đa k hoạt động chỉnh sửa ta biến đổi được đồ thị G

thành đồ thị $G_0 = (C_0 \sqcup I_0, E_0)$ (C_0 là Clique của G_0 và I_0 là tập đỉnh cô lập). Gọi v là một đỉnh của G, để tập tất cả lân cận của v trong G_0 khác rỗng thì $v \in C_0$, và khi đó $N_v(G_0) = C_0 \setminus \{v\}$.
Giả sử ta có đỉnh v trong G và biết trước tập đỉnh $N_v(G_0) = C_0 \setminus \{v\}$. Khi đó ta sẽ chỉnh sửa G thành G_0 như sau: đầu tiên trong đồ thị con $G[C_0]$ ta thêm tất cả cạnh giữa các cặp đỉnh không kề nhau, sau đó bỏ đi tất cả các cạnh mà có ít nhất một đầu mút nằm trong I_0. Khi đó ta được số phép chỉnh sửa là

$$(m - m_{G[C]}) + \Big(\frac{|C_0|.(|C_0|-1)}{2} - m_{G[C]}\Big) = m + \frac{|C_0|.(|C_0|-1)}{2} - 2m_{G[C]}$$

□

Định nghĩa 3.2.3. *Cho đồ thị $G = (V, E)$ và một đỉnh v của G. Ta nói việc "xây dựng từ lân cận" của v trong G là một loạt các phép chỉnh sửa trên G để tạo thành G' sao cho $V(G')$ là hợp của một Clique $N_v(G) \cup \{v\}$ và tập đỉnh cô lập.*

Xét (G, k) là bộ dữ liệu đầu vào của bài toán Clique Editing, và cho trước số nguyên dương c. Ta đến với quy tắc 1.
Quy tắc 1. Ta xét duyệt từng đỉnh của G. Với mỗi đỉnh v của G, ta duyệt từng tập con $N \subset V(G)$ không chứa v và $1 \leq |N| \leq c - 1$. Ứng với mỗi tập con N ấy ta thực hiện các hoạt động chỉnh sửa giữa các cặp đỉnh (v, v') với $(v' \in N)$, sau khi thực hiện đúng $|N|$ chỉnh sửa như thế, ta được đồ thị G_1, tiếp tục "xây dựng từ lân cận" của v trong G_1 ta được đồ thị G_2, nếu hoạt động "xây dựng từ lân cận" ấy có độ phức tạp không vượt quá k thì ta dừng việc xét duyệt và cho ra kết quả YES (bài toán được giải), còn nếu độ phức tạp vượt quá k thì việc xét duyệt vẫn tiến hành tiếp.
Nếu toàn bộ các bước xét duyệt trên đều không có trường hợp nào cho kết quả YES thì ta đi đến quy tắc 2.

Mệnh đề 3.2.2. *Thời gian thực hiện quy tắc 1 là $O(mn^c)$, trong đó $V(G) = n, E(G) = m$*

Chứng minh. Với mỗi v, thì số tập hợp N có p phần tử như thế là $\binom{n-1}{p}$, cho p chạy từ 1 đến $c-1$ ta được tổng số chỉnh sửa:

$$\sum_{p=1}^{c-1} p\binom{n-1}{p}$$

Mà đồ thị G có n đỉnh nên số trường hợp chỉnh sửa tối đa là

$$n\sum_{p=1}^{c-1} p\binom{n-1}{p} = O(n^c)$$

Theo mệnh đề (3.2.1), mỗi lần "xây dựng từ lân cận" là tốn thời gian $o(m)$. Hơn nữa G có n đỉnh cho nên tổng độ phức tạp của quy tắc 1 là $o(mn^c)$. □

Xây dựng quy tắc 2

Giả sử sau khi thực hiện quy tắc 1 mà ta chưa tìm thấy nghiệm (tức không có trường hợp xét duyệt nào cho kết quả "YES"). Thì khi đó, muốn chỉnh sửa đồ thị G thành nghiệm G_0 như mong muốn (Clique C và tập đỉnh cô lập I) thì tại mỗi đỉnh của G nằm trong C đều phải thực hiện không ít hơn c chỉnh sửa (nghĩa là với mỗi đỉnh $v \in C$, luôn tồn tại tập $N \subset V(G) \setminus \{v\}$, $|N| \geq c$, sao cho giữa các cặp đỉnh (v, v'), $\forall v' \in N$ đều có đúng một phép chỉnh sửa). Khi đó ta thực hiện tiếp quy tắc sau:

Quy tắc 2. Với mỗi đỉnh v của G mà có bậc nhỏ hơn c thì ta bỏ đi tất cả các cạnh liên thuộc với v và thay số nguyên k thành $k' = k - deg(v)$, để được bộ dữ liệu đầu vào mới (G', k'), khi đó v trở thành điểm cô lập của G' và mọi hoạt động chỉnh sửa sau đó trên G' không liên quan đến v, lúc này ta có thể xem như G' có tập đỉnh, tập cạnh là

$$V(G') = V(G) \setminus \{v \in V(G) \mid deg(v) < c\}$$
$$E(G') = E(G) \setminus \{\{v, v'\} \in E(G) \mid deg(v) < c\}$$

Mệnh đề 3.2.3. *(G', k') thỏa mãn bài toán Clique Editing khi và chỉ khi (G, k) thỏa mãn bài toán Clique Editing.*

Chứng minh. Giả sử G chỉ có 1 đỉnh v thỏa mãn $deg_G(v) < c$.

Bây giờ giả sử (G', k') thỏa mãn bài toán Clique Editing, tức là có thể chỉnh sửa G' thành nghiệm G'_0 bởi tối đa $k' = k - deg_G(v)$ phép chỉnh sửa. Khi đó ta thực hiện chỉnh sửa G bằng cách bỏ đi các cạnh liên thuộc với v, rồi sau đó chỉnh sửa phần đồ thị $G[V(G) \setminus \{v\}]$ như cách mà chỉnh sửa từ G' thành G'_0, lúc này số chỉnh sửa tối đa là $deg_G(v) + (k - deg_G(v)) = k$. Như vậy (G, k) thỏa mãn bài toán Clique Editing.

Giả sử (G, k) thỏa mãn bài toán Clique Editing, tức là có thể chỉnh sửa G thành nghiệm G_0 bởi tối đa k phép chỉnh sửa. Gọi δv là số chỉnh sửa đồ thị G tại đỉnh v để tạo thành G_0. Có 2 trường hợp, tại v có không ít hơn c chỉnh sửa để tạo ra một tập các lân cận không rỗng, hoặc tại v có một số chỉnh sửa nào đó để tạo thành đỉnh cô lập v trong G_0. Cho nên ta có thể giả sử hoạt động chỉnh sửa G thành G_0 bao gồm việc xóa hết tất cả các cạnh liên thuộc với v, những hoạt động chỉnh sửa còn lại trên G như thế nào thì ta làm như thế ấy đối với G'. Lúc này số chỉnh sửa tối đa trên G' là $k - deg_G(v)$. Như vậy (G', k') thỏa mãn bài toán Clique Editing.

□

Thời gian thực hiện quy tắc 2 là $O(m)$.

Xây dựng quy tắc 3

Chúng ta thấy rằng sau khi thực hiện quy tắc 2 thì được bộ dữ liệu đầu vào mới (G', k') có kết quả tương đương với dữ liệu đầu vào (G, k) lúc ban đầu. Trong đó G' là một đồ thị con của G, và muốn chỉnh sửa G' thành nghiệm thì tại mỗi đỉnh của G' đều được thực hiện tối thiểu c phép chỉnh sửa. Như vậy bây giờ ta sẽ thực hiện tiếp quy tắc 3 đối với bộ đầu vào (G', k').

Quy tắc 3. Nếu đồ thị G' có nhiều hơn $\frac{2k}{c}$ đỉnh thì (G', k') không thỏa mãn

bài toán Clique Editing, nghĩa là không thể chỉnh sửa G' thành nghiệm với tối đa k' phép chỉnh sửa. Lúc này bài toán Clique Editing với bộ đầu vào (G, k) trả lời NO.

Mệnh đề 3.2.4. *Quy tắc 3 là đúng.*

Chứng minh. Giả sử có thể chỉnh sửa G' thành nghiệm G'_0 bởi tối đa k' phép chỉnh sửa. Trong quá trình chỉnh sửa đó, gọi δv là số chỉnh sửa tại định $v \in V(G')$, và gọi t là tổng số chỉnh sửa trên G'. Khi đó ta có

$$2k \geq 2t = \sum_{v \in V(G')} \delta v \geq V(G').c$$

Suy ra $V(G') \leq \dfrac{2k}{c}$.

□

Kết luận: Nếu (G, k) được biến đổi thông qua quy tắc 1, và quy tắc 2 để được (G', k') thỏa mãn điều kiện $V(G') \leq \dfrac{2k}{c}$, thì ta dễ thấy rằng:

+ Thuật toán để biến đổi (G, k) thành (G', k') là $O(n^{c+2})$.
+ $k' \leq k$ và $n \leq \dfrac{2k}{c} + k$.
+ (G, k) thỏa mãn bài toán Clique Editing khi và chỉ khi (G', k') thỏa mãn bài toán Clique Editing.

Như vậy ta có thể nói (G', k') lúc này là một Kernel của (G, k).

Định nghĩa 3.2.4. *Việc thực hiện lần lượt các quy tắc* 1*, quy tắc* 2*, quy tắc* 3 *như trên được gọi là "thuật toán nhân tử hóa" cho bài toán Clique Editing đối với bộ dữ liện đầu vào* (G, k) *và một số nguyên dương* c*.*

3.3 Thuật toán FPT cho bài toán Clique Editing

3.3.1 Thuật toán

Xét bộ dữ liệu đầu vào (G, k) của bài toán Clique Editing.

Bước 1. Thực hiện thuật toán nhân tử hóa cho (G, k) với $c = 2$. Nếu trước khi kết thúc quy tắc 3 mà thuật toán đã dừng thì bài toán Clique Editing đối với bộ đầu vào (G, k) được giải quyết, khi đó ta không cần thực hiện các bước sau. Ngược lại thuật toán nhân tử hóa vẫn chạy và tạo ra nhân tử (G', k') với $|V_{G'}| \leq 2k/2 = k$, $k' \leq k$.

Bước 2. Thực hiện thuật toán nhân tử hóa cho (G', k') với $c = \sqrt{k/log(k)}$. Nếu trước khi kết thúc quy tắc 3 mà thuật toán đã dừng thì bài toán Clique Editing đối với bộ đầu vào (G, k) được giải quyết, khi đó ta không cần thực hiện các bước sau. Ngược lại, thuật toán vẫn chạy và tạo ra một nhân tử (G'', k''), với $|V_{G''}| \leq \frac{2k'}{\sqrt{k/log(k)}} \leq \frac{2k}{\sqrt{k/log(k)}} = \sqrt{2k.log(k)}$.

Bước 3. Từ nhân tử (G'', k'') ta thực hiện việc xét duyệt hết tất cả các tập con của $V_{G''}$ để tìm ra nghiệm. Thuật toán sẽ dừng và cho kết quả YES nếu tìm ra nghiệm, ngược lại thì trả lời NO.

3.3.2 Độ phức tạp

Bước 1 được thực hiện với thời gian tối đa là n^4.

Bước 2 được thực hiện với thời gian tối đa là $|V_{G'}|^{\sqrt{k/log(k)}}.|E_{G'}| \leq k^{\sqrt{k/log(k)}}.k^2 = 2^{\sqrt{k/log(k)}.log(k)}.k^2 = 2^{O(\sqrt{k.log(k)})}$

Bước 3 được thực hiện trong thời gian $2^{|V_{G''}|} = 2^{O(\sqrt{k.log(k)})}$

Do đó tổng thời gian tối đa mà thuật toán thực hiện là:

$$n^{O(1)} + 2^{O(\sqrt{k.log(k)})}$$

Mệnh đề 3.3.1. *Bài toán Clique Editing là một bài toán thuộc lớp FPT.*

Trong khi câu hỏi "$P = NP$?" chưa có trả lời thì thuật toán FPT có vai trò khá quan trọng trong khoa học tính toán. Trong thực tế, ta thường chọn được những lớp đồ thị đầu vào gần hoàn chỉnh so với mong muốn, khi đó thời gian $n^{O(1)} + 2^{O(\sqrt{k.log(k)})}$ tốt hơn nhiều so với một hàm mũ hoặc một hàm lớn hơn đa thức đối với biến n.

Luận văn này có các phần chính đó là: tìm hiểu bài toán Clique Editing và tính NP-đầy đủ của nó, lớp FPT và bài toán Clique Editing thuộc lớp FPT. Việc chứng minh bài toán Clique Editing là tương đối phức tạp, nó thông qua một bài toán khá hay đó là BCBS về đồ thị 2 phía được trình bày ở chương 1. Còn việc sử dụng phương pháp nhân tử hóa và xây dựng thuật toán FPT cho bài toán Clique Editing cũng là việc khá tinh vi và phức tạp.

Ở luận văn này là chúng ta mong muốn xây dựng một nghiệm đồ thị bao gồm chỉ một Clique từ đồ thị ban đầu. Từ đó, câu hỏi đặt ra hoàn toàn tự nhiên là liệu đối với nghiệm là một đồ thị bao gồm 2,3... Clique thì sao? Sự thật là bài toán đối với 2 Clique cũng thuộc lớp FPT (điều đấy đã được chứng minh cách đây không lâu). Do đó chúng ta hoàn toàn có thể tin rằng bài toán $3, 4, 5...$ Clique cũng thuộc lớp FPT.

KẾT LUẬN

Trong luận văn "Độ phức tạp của bài toán biến đổi đồ thị về đồ thị đầy đủ", tôi đã trình bày được một số vấn đề như sau:

1. Trình bày sơ lược về độ phức tạp thuật toán và các khái niệm lớp P, NP. Trình bày về phép quy dẫn và thế nào là lớp NP-khó, NP-đầy đủ. Sau đó đã nêu một số bài toán nằm trong lớp NP-đầy đủ, đặc biệt các bài toán liên quan đến bài toán Clique, Biclique.

2. Giới thiệu bài toán chỉnh sửa nói chung, bài toán Clique Editing nói riêng.

3. Trình bày một số tính chất xung quanh bài toán Clique Editing đối với đồ thị 2 phía. Đồng thời cũng chứng minh được bài toán Clique Editing trên đồ thị 2 phía và đồ thị tổng quát là NP-đầy đủ. Bài toán Clique Editing trên đồ thị phẳng là thuộc lớp P.

4. Giới thiệu về thuật toán FPT và xây dựng thuật toán FPT cho bài toán Clique Editing. Trình bày cụ thể thuật toán FPT với những kỹ thuật rất tinh tế và với phương pháp nhân tử hóa hữu hiệu.

5. *Đóng góp:* phương pháp nhân tử hóa, thuật toán FPT là rất mới mẻ với những học viên Toán như chúng tôi. Trong luận văn này, các chứng minh đều có nhiều việc phân tích rất kỹ cấu trúc đồ thị và việc xây dựng các thuật toán rất tinh vi và phức tạp.

Bài báo "Editing Graphs into Few Cliques: Complexity, Approximation, and Kernelization Schemes" mà tôi tham khảo được viết khá vắn tắt và cô đọng. Chúng tôi đã trình bày lại có hệ thống về thuật toán FPT, phương pháp nhân tử hóa, từng quy tắc, từng chi tiết bé của thuật toán, từng phân tích độ phức tạp của thuật toán.

Luận văn này có thể xem là một tài liệu khá kỹ về bài toán Clique và Clique Editing.

Tài liệu tham khảo

Tài liệu Tiếng Việt

[1] Ngô Đắc Tân, *Lý thuyết Tổ hợp và Đồ thị*, Viện Toán học, Hà Nội, 2003.

Tài liệu Tiếng Anh

[2] David S. Johnson, *The NP-Completeness Column: An Ongoing Guide*, Journal of Algorithms 8(3), 438–448, 1987.

[3] Falk Hüffner, Christian Komusiewicz, André Nichterlein, *Editing Graphs into Few Cliques: Complexity, Approximation, and Kernelization Schemes*, F. Dehne et al. (Eds.): WADS 2015, LNCS 9214, pp. 410–421, 2015.

[4] Ivan Kováč, Ivana Selečéniová, Monika Steinová, *On the Clique Editing Problem*, MFCS 2014: Mathematical Foundations of Computer Science, pp 469-480, 2014.

[5] Richard Karp, *Reducibility Among Combinatorial Problems*, New York: Plenum. pp. 85–103, 1972.

[6] Stephen Cook, *The complexity of theorem proving procedures*, Proceedings of the Third Annual ACM Symposium on Theory of Computing. pp. 151–158, 1971.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, *Introduction to Algorithms*, The MIT Press, 2001.