Kỷ yếu Hội nghị Khoa học Quốc gia lần thứ IX "Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR'9)"; Cần Thơ, ngày 4-5/8/2016
DOI: 10.15625/vap.2016.00079

RÚT GỌN ĐỒ THỊ CHO BÀI TOÁN CÂY STEINER NHỎ NHẤT

Phan Tấn Quốc
Trường Đại học Sài Gòn
quocpt@sgu.edu.vn

TÓM TẮT— *Cây Steiner nhỏ nhất (Steiner Minimal Tree - SMT) là bài toán tối ưu tổ hợp có nhiều ứng dụng quan trọng trong khoa học và kỹ thuật; đây là bài toán thuộc lớp NP-hard và hiện đang được nghiên cứu rộng rãi. Các tiếp cận giải bài toán SMT - dù là giải đúng hay giải gần đúng - thì công đoạn rút gọn đồ thị là rất quan trọng; công đoạn này càng có ý nghĩa đối với các tiếp cận giải đúng bài toán SMT. Trong hàng chục năm qua, đã có hàng loạt thuật toán rút gọn đồ thị cho bài toán SMT đã được công bố; các kết quả này đã hỗ trợ tích cực cho việc giải bài toán SMT với kích thước lớn hơn. Bài báo này đề xuất thuật toán rút gọn đồ thị cho bài toán SMT và kết quả thực nghiệm là thông tin hữu ích cho việc nghiên cứu các thuật toán giải đúng cũng như các thuật toán giải gần đúng cho bài toán SMT.*

Từ khóa— *Steiner minimal tree, graph reduction, branch and bound algorithm, exact algorithms for steiner minimal tree.*

I. GIỚI THIỆU

A. Một số định nghĩa

Cho n điểm $P_1,P_2,\ldots,P_n$. Giả sử cần tìm một mạng giao thông nối k điểm (trong số n điểm đã cho) với nhau, mạng giao thông này có thể sử dụng thêm một số điểm khác - cũng trong số n điểm đã cho và ngoài k điểm đã chọn - sao cho tổng độ dài của các đoạn thẳng nối các điểm là nhỏ nhất; đây là bài toán cây Steiner nhỏ nhất. Mục này bài báo sẽ trình bày một số định nghĩa và tính chất liên quan.

Định nghĩa 1. *Cây Steiner* [2]

Cho $G = (V(G), E(G))$ là một đơn đồ thị vô hướng liên thông và có trọng số không âm trên cạnh; trong đó $V(G)$ là tập gồm n đỉnh, $E(G)$ là tập gồm m cạnh, $w(e)$ là trọng số của cạnh e, $e \in E(G)$. Cho Y là tập con các đỉnh của $V(G)$, cây T đi qua tất cả các đỉnh trong Y được gọi là cây Steiner của Y.

Tập Y được gọi là tập *terminal*, các đỉnh thuộc tập Y được gọi là các đỉnh *terminal*, các đỉnh thuộc cây T mà không thuộc tập Y được gọi là các đỉnh Steiner. Khác với các bài toán cây khung, cây Steiner chỉ cần đi qua tất cả các đỉnh thuộc tập *terminal* Y và có thể thêm một số đỉnh khác nữa thuộc tập $V(G)$.

Định nghĩa 2. Chi phí cây Steiner [2]

Cho $T = (V(T), E(T))$ là một cây Steiner của đồ thị G, chi phí của cây T, ký hiệu là $C(T)$, là tổng trọng số của các cạnh thuộc cây T, tức là $C(T) = \sum_{e \in E(T)} w(e)$.

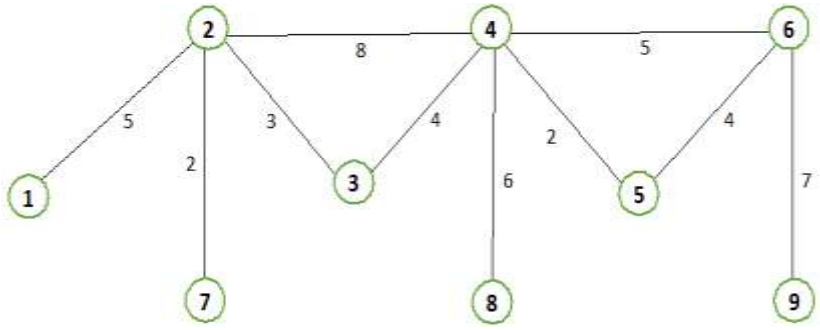
Định nghĩa 3. *Cây Steiner nhỏ nhất* [2]

Cho đồ thị G được mô tả như trên, bài toán tìm cây Steiner có chi phí nhỏ nhất được gọi là bài toán cây Steiner nhỏ nhất (Steiner Minimal Trees problem – *SMT*).

SMT là bài toán tối ưu tổ hợp trên lý thuyết đồ thị. Trong trường hợp tổng quát, *SMT* đã được chứng minh là bài toán thuộc lớp bài toán *NP-hard* [16,18]. Có hai trường hợp đặc biệt đối với bài toán *SMT* có thể giải được bằng thời gian đa thức; đó là khi $Y=V(G)$ và khi $|Y|=2$ ($|Y|$ là ký hiệu số lượng đỉnh của tập Y): Khi $Y=V$ thì bài toán *SMT* có thể giải bằng các thuật toán tìm cây khung nhỏ nhất; chẳng hạn như các thuật toán Prim, Kruskal; khi $|Y|=2$ thì bài toán *SMT* có thể giải được bằng các thuật toán tìm đường đi ngắn nhất giữa hai đỉnh; chẳng hạn thuật toán Dijkstra (để ngắn gọn, trong bài báo này từ *đồ thị* sẽ được hiểu là đơn đồ thị, vô hướng, liên thông, có trọng số không âm).

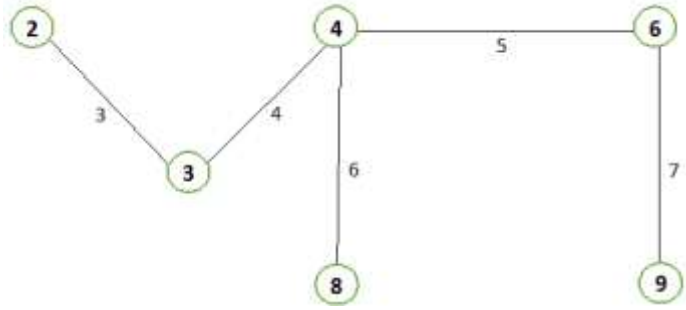
Ví dụ:

Cho một đồ thị G có 9 đỉnh và 10 cạnh như hình vẽ 1 và tập $Y=\{2,8,9\}$.



Hình 1. Đồ thị vô hướng liên thông có trọng số G

Khi đó cây Steiner nhỏ nhất tìm được ứng với tập *Y* trên đồ thị *G* là *T* có *V*(*T*)={2,3,4,6,8,9} và *E*(*T*) = {(2,3), (3,4), (4,6), (4,8), (6,9)} như được minh họa ở hình vẽ 2; *T* có tập đỉnh Steiner là {3,4,6} và có chi phí là 25.



Hình 2. Cây Steiner nhỏ nhất của tập *Y* trên đồ thị *G*

Định lý. *Định lý về số đỉnh Steiner* [30]

Cho đồ thị *G* và tập *terminal Y*, cây Steiner *T* của *Y* có *p* đỉnh thì số đỉnh Steiner của *T* không vượt quá *p*–2.

Tiếp theo chúng tôi đưa ra các khái niệm về *cạnh cầu Steiner* và *đồ thị rút gọn Steiner* để diễn đạt cho các nội dung ở phần tiếp theo.

Định nghĩa 4. *Cạnh cầu Steiner*

Cho đồ thị *G* và tập *terminal Y*, cạnh e_{uv} được gọi là cạnh cầu Steiner của *G* nếu khi loại cạnh e_{uv} thì tập các đỉnh *terminal Y* không cùng thuộc về một thành phần liên thông.

Định nghĩa 5. *Đồ thị rút gọn Steiner*

Cho đồ thị *G* và tập *terminal Y*, *G*' được gọi là đồ thị rút gọn Steiner của *G* nếu số đỉnh và số cạnh của *G*' nhỏ hơn hoặc bằng số đỉnh và số cạnh của *G* và trong *G*' tồn tại ít nhất một *SMT* ứng với tập *terminal Y* của đồ thị *G*.

B. Ứng dụng của bài toán SMT

Bài toán *SMT* có thể được tìm thấy trong các ứng dụng quan trọng trong các lĩnh vực khoa học và kỹ thuật; chẳng hạn như bài toán thiết kế mạng truyền thông [2], bài toán định tuyến trong *VLSI* [2, 8], các bài toán liên quan đến hệ thống mạng với chi phí nhỏ nhất [7, 32],…

C. Một số nghiên cứu liên quan bài toán SMT và vấn đề đặt ra cần giải quyết

Do có tính khoa học và tính ứng dụng rộng rãi, bài toán *SMT* đã thu hút được sự quan tâm nghiên cứu liên tục, sâu rộng của nhiều nhà khoa học trên thế giới trong hàng chục năm qua. Các mô hình của bài toán cây Steiner như bài toán Steiner với khoảng cách Euclide (*Euclidean Steiner Tree problem*) [29], bài toán Steiner với khoảng cách chữ nhật (*Rectilinear Steiner Tree problem*) [14], bài toán Steiner cho đồ thị vô hướng,… Hiện tại đã có hàng loạt thuật toán giải bài toán *SMT* được đề xuất và có thể chia chúng làm bốn hướng tiếp cận sau [3, 17, 19, 22]:

Hướng thứ nhất là các thuật toán tìm lời giải đúng. Chẳng hạn thuật toán quy hoạch động của Dreyfus và Wagner [24], thuật toán dựa trên phép nới lỏng lagrange của Beasley [13], thuật toán nhánh cận của Koch và Martin [28],… Ưu điểm của hướng tiếp cận này là tìm được lời giải chính xác, nhược điểm của hướng tiếp cận này là chỉ giải được các bài toán có kích thước nhỏ. Hướng tiếp cận này là cơ sở quan trọng để đánh giá mức độ chính xác của các thuật toán giải gần đúng. Việc giải đúng bài toán *SMT* thực sự là một thách thức lớn trong lý thuyết tối ưu tổ hợp [4, 6, 21, 23, 26, 31].

Hướng thứ hai là các thuật toán tìm lời giải gần đúng cận tỉ lệ. Ưu điểm của các thuật toán này là có sự đảm bảo về mặt toán học theo nghĩa lời giải tìm được gần đúng một cận tỉ lệ α nào đó so với lời giải tối ưu, nhược điểm của thuật toán này là cận tỉ lệ tìm được của nhiều thuật toán đề xuất trong thực tế thường là kém hơn rất nhiều so với chất lượng lời giải tìm được bởi nhiều thuật toán gần đúng khác dựa trên thực nghiệm. Thuật toán *MST-Steiner* có cận tỉ lệ 2 (MST-Minimum Spanning Trees) [2], thuật toán *Zelikovsky-Steiner* có cận tỉ lệ 11/6 [2] là các thuật toán điển hình của hướng tiếp cận này [12].

Hướng thứ ba là các thuật toán heuristic. Thuật toán heuristic chỉ những kinh nghiệm riêng biệt để tìm kiếm lời giải cho một bài toán tối ưu cụ thể. Thuật toán heuristic thường tìm được lời giải có thể chấp nhận được trong thời gian cho phép nhưng không chắc đó là lời giải tốt nhất; thậm chí các thuật toán heuristic không chắc hiệu quả trên mọi loại dữ liệu đối với một bài toán cụ thể. Ưu điểm của các thuật toán heuristic là cho thời gian chạy nhanh. Các thuật toán Floyd tìm đường đi ngắn nhất giữa mọi cặp đỉnh của đồ thị, các thuật toán Kruskal, thuật toán Prim để tìm cây khung nhỏ nhất của đồ thị, thuật toán Distance Network Heuristic của Kou, Markowsky và Berman [15],… có thể được xem là các heuristic cho bài toán *SMT*; tuy nhiên chi phí tìm được các heuristic này không hiệu quả bằng các thuật toán metaheuristic trong việc giải bài toán *SMT* [27].

Hướng thứ tư là các thuật toán metaheuristic. Thuật toán metaheuristic sử dụng nhiều heuristic kết hợp với các kỹ thuật phụ trợ nhằm khai phá không gian tìm kiếm; metaheuristic thuộc lớp các thuật toán tìm kiếm tối ưu. Hiện đã có nhiều công trình sử dụng thuật toán metaheuristic giải bài toán *SMT*; chẳng hạn như thuật toán tìm kiếm tabu [5], thuật toán di truyền [1], thuật toán di truyền song song [29],…. Cho đến hiện tại, hướng tiếp cận metaheuristic cho kết quả tốt nhất trong số các thuật toán giải gần đúng.

Bài toán *SMT* có hệ thống dữ liệu thực nghiệm chuẩn [10] và nhiều công bố liên quan đã tiến hành thực nghiệm trên hệ thống dữ liệu này. Các đồ thị trong hệ thống dữ liệu này có tối đa là 2500 đỉnh, 62500 cạnh. Một số công trình đã công bố kết quả thực nghiệm trên các bộ dữ liệu chuẩn này [2, 28].

Rút gọn đồ thị là một trong những chủ đề nghiên cứu liên quan đến bài toán *SMT* hiện nay. Hầu hết các đồ thị gặp trong thực tế ứng dụng là đồ thị thưa. Một số công trình liên quan bài toán *SMT* đã trình bày các kỹ thuật nhằm giảm thiểu kích thước của đồ thị; chẳng hạn công trình của Jeffrey H.Kingston và Nicholas Paul Sheppard [11], công trình của Thorsten Koch, Alexander Martin [28],…Trong bài báo này, chúng tôi đề xuất một thuật toán rút gọn đồ thị cho giải bài toán *SMT*; chúng tôi thực nghiệm các thuật toán này với 18 bộ dữ liệu được lấy trong hệ thống dữ liệu thực nghiệm chuẩn. Đề xuất này có thể xem là một bước tiền xử lý quan trọng trong việc giải bài toán *SMT*.

II. THUẬT TOÁN RÚT GỌN ĐỒ THỊ CHO BÀI TOÁN CÂY STEINER

Trong phần này, trước hết chúng tôi đề xuất một thuật toán rút gọn đồ thị cho bài toán *SMT* dựa trên một số bài toán và tính chất kinh điển của lý thuyết đồ thị như *đuòng đi ngắn nhất* (tính chất 1), *cạnh cầu Steiner* (tính chất 2), *phân rã đồ thị* (tính chất 3). Cũng trong phần này, chúng tôi đề xuất thuật toán nhánh cận và hai thuật toán heuristic giải bài toán *SMT*.

A. Một số tính chất rút gọn đồ thị

Cho đồ thị *G* và tập *terminal Y* như định nghĩa 1, ý tưởng chung của các tính chất nhằm rút gọn đồ thị là làm *gia tăng* các đỉnh cho tập *terminal* (các đỉnh này sẽ thuộc về cây Steiner nhỏ nhất cần tìm) và *loại bỏ* các đỉnh của đồ thị mà nó chắc chắn không thuộc về cây Steiner nhỏ nhất cần tìm; nghĩa là khi đó cũng đã xác định được các cạnh kề với nó cần loại bỏ. Tương tự, việc *gia tăng* các đỉnh cho tập *terminal* còn có thể được thực hiện qua việc xác định một cạnh nào đó của đồ thị phải thuộc về cây Steiner nhỏ nhất cần tìm (nghĩa là hai đỉnh kề của cạnh đó phải thuộc về tập *terminal)*. Các tính chất rút gọn đồ thị cũng tập trung vào việc *loại bỏ* một cạnh nào đó của đồ thị mà nó chắc chắn không thuộc cây Steiner nhỏ nhất cần tìm. Chất lượng các thuật toán giải bài toán *SMT* phụ thuộc vào độ lớn của hệ số $n-|Y|$; do vậy mục đích của các thuật toán rút gọn đồ thị là làm giảm thiểu tối đa hệ số $n-|Y|$.

Để thuận tiện cho việc mô tả các thuật toán bằng ngôn ngữ mã giả, chúng tôi sẽ định nghĩa hai mảng một chiều edge_select và vertex_select để đánh dấu một cạnh hoặc một đỉnh nào đó có thuộc về cây Steiner nhỏ nhất cần tìm hay không. Các biến mảng của đồ thị ban đầu được khởi tạo như sau: edge_select[e]=1, $\forall e \in E(G)$ và vertex_select [u]=1, $\forall u \in V(G)$. Khi xác định một cạnh hoặc một đỉnh nào đó chắc chắn không thuộc về cây Steiner nhỏ nhất cần tìm thì các biến mảng này sẽ được gán bằng giá trị 0.

1. Tính chất 1 (Shortest Paths - SP)

Cho đồ thị *G* và tập *terminal Y*, đỉnh $x \notin Y$ và x không nằm trên ít nhất một đường đi ngắn nhất (Shortest Path) nào nối hai đỉnh u, v với mọi $u, v \in$ Y thì đỉnh x và các cạnh kề với đỉnh x sẽ không thuộc về *SMT* cần tìm.

Đoạn mã giả cho tính chất này được minh họa như sau (gọi là thuật toán *SP*):

1. Cho mảng một chiều visit[u], đặt visit[u]=0, $\forall u \in V(G)$;
2. Tìm ma trận khoảng cách (d) và ma trận đường đi (p) giữa mọi cặp đỉnh u, v$\in V(G)$ (sử dụng thuật toán Floyd [25]);
3. **for** ($\forall u, v \in Y$)
4. Dựa vào ma trận đường đi (p), đánh dấu các đỉnh thuộc về đường đi ngắn nhất nối u, v; tức *visit*[z]=1 với mọi đỉnh z thuộc đường đi đó;
5. **for** ($\forall u \in V(G)$)
6. **if** ($u \notin Y$ và *visit*[u] = 0)
7. vertex_select[u]=0; // nghĩa là xác định được đỉnh u không thuộc về *SMT* cần tìm
8. **for** ($e_{uv} \in E(G)$)
9. **if** (vertex_select[u]=0 hoặc vertex_select[v]=0)
10. edge_select[e_{uv}]=0; // nghĩa là xác định được các cạnh không thuộc về *SMT* cần tìm

Thuật toán rút gọn đồ thị *SP* có độ phức tạp thời gian tính là $O(n^3)$; đây cũng là độ phức tạp thời gian tính của thuật toán Floyd (dòng thứ 2).

Hàng loạt tính chất rút gọn đồ thị khác đã được bao hàm trong tính chất 1; chẳng hạn các tính chất "Cạnh kề với một đỉnh treo không thuộc tập *terminal Y* của đồ thị *G* đều không thuộc về *SMT* cần tìm", "Cạnh e=(u,v) kề với đỉnh treo u, mà u thuộc tập *terminal Y* thì v cũng thuộc *terminal Y*", "Nếu *G* có cạnh cầu e; khi đó *G* được tách thành hai thành phần liên thông theo cạnh cầu e và nếu thành phần liên thông nào không chứa ít nhất một đỉnh nào thuộc tập

terminal Y thì tất cả các đỉnh thuộc thành phần liên thông đó đều không thuộc về *SMT* cần tìm", "Xét từng bộ ba cạnh của đồ thị *G* đôi một chung đỉnh, nếu một cạnh có trọng số lớn hơn tổng trọng số của hai cạnh còn lại thì cạnh đó không thuộc về *SMT* cần tìm", "Một đường đi (*p*) nối hai đỉnh *u,v* mà không đi qua một đỉnh *terminal* nào và nếu tổng trọng số các cạnh trên đường đi (*p*) lớn hơn w_{uv} thì các đỉnh thuộc (*p*) sẽ không thuộc về *SMT* cần tìm; ngược lại, nếu tổng trọng số các cạnh trên đường đi (p) nhỏ hơn w_{uv} thì cạnh (*u,v*) không thuộc về *SMT* cần tìm",…

2. Tính chất 2 (Bridge Steiner-BS)

Cho đồ thị *G* và tập *terminal Y*. Các đỉnh kề với các cạnh cầu Steiner (định nghĩa 4 ở trên) đều thuộc về *SMT* cần tìm.

Đoạn mã giả cho tính chất này được minh họa như sau (gọi là thuật toán *BS*):

```
1. for (∀e_uv ∈ E(G))
2. if (e_uv là cạnh cầu Steiner) {
3.    if (u ∉ Y )
4.          Y=Y ∪ u;
5.    if (v ∉ Y )
6.          Y=Y ∪ v;
7. }
```

Thuật toán rút gọn đồ thị *BS* có độ phức tạp thời gian tính là $O(mn|Y|)$.

3. Tính chất 3 (Decomposition Graph-DG)

Cho đồ thị *G* và tập *terminal Y*. *G* được tách thành các thành phần liên thông theo các cạnh cầu Steiner thì khi đó *SMT* cần tìm là hợp thành của các cạnh cầu steiner và của các cây steiner nhỏ nhất tìm được ứng với mỗi thành phần liên thông tương ứng.

Đoạn mã giả cho tính chất này được minh họa như sau (gọi là thuật toán *DG*):

1. Tìm các cạnh cầu Steiner của đồ thị *G*; giả sử đó là $e_1,e_2,\ldots,e_k$;
2. Phân rã *G* thành *k*+1 thành phần liên thông $G_1,G_2,\ldots,G_{k+1}$ (mỗi thành phần liên thông G_i sẽ xác định được một *SMT* tương ứng; giả sử đó là SMT_1, $SMT_2,\ldots$, SMT_{k+1}; khi đó *SMT* cần tìm của đồ thị là $\cup_{i=1..k} e_i \cup \cup_{i=1..k+1} SMT_i$).

Thuật toán rút gọn đồ thị *DG* có độ phức tạp thời gian tính là $O(nm)$.

Tiếp theo, bài báo sẽ đề xuất một thuật toán rút gọn đồ thị có tên là *SP-BS* cho bài toán *SMT*; *SP-BS* sử dụng các thuật toán *SP* và *BS* tương ứng với tính chất 1 và tính chất 2.

B. Thuật toán rút gọn đồ thị SP-BS

Input: Đồ thị *G* và tập terminal *Y;*

Output: Đồ thị sau rút gọn;

```
1. condition_loop=1;
2. k=n_terminal; // n_terminal là số đỉnh của tập terminal
3. while (condition_loop){
4.    condition_loop=0;
5.    SP(); // cập nhật đồ thị G; cập nhật tập terminal
6.    BS(); // cập nhật đồ thị G; cập nhật tập terminal
7.    if (n_terminal>k){
8.          k=n_terminal;
9.          condition_loop=1;
10.   }
11. }
```

C. Thuật toán rút gọn đồ thị SP-DG

Từ tính chất 3, chúng tôi đề xuất một thuật toán khác cho kết quả tương đương với thuật toán *SP-BS* trên; và gọi là thuật toán *SP-DG*; thuật toán *SP-DG* được cho thực hiện thuật toán *SP* trước và tiếp theo là *DG*.

Các thuật toán *SP-BS* và *SP-DG* tương đương theo nghĩa có hiệu số $n-|Y|$ như nhau; ở đây, hiệu số $n-|Y|$ của *SP-DG* được chọn là số $n-|Y|$ lớn nhất trong số các đồ thị thu được sau phép tách. Rõ ràng dù có $n-|Y|$ như nhau; nhưng các đồ thị thu được bằng thuật toán *SP-DG* sẽ thuận lợi hơn trong việc tìm kiếm *SMT* nhỏ nhất tương ứng (điều này đã thể hiện qua các thực nghiệm được chúng tôi trình bày chi tiết ở phần tiếp theo); đặc biệt là đối với các thuật toán tiếp cận theo hướng tìm lời giải đúng.

THỰC NGHIỆM VÀ ĐÁNH GIÁ

Phần này trình bày thực nghiệm thuật toán rút gọn đồ thị và một số thuật toán tìm cây Steiner nhỏ nhất.

A. Dữ liệu thực nghiệm

Chúng tôi sử dụng tổng cộng 18 bộ dữ liệu đồ thị thưa trong hệ thống dữ liệu thực nghiệm chuẩn cho bài toán cây Steiner: URL http://people.brunel.ac.uk/~mastjjb/jeb/orlib/steininfo.html [10]. Thông tin chi tiết về 18 bộ dữ liệu này được trình bày ở bảng 1; trong đó các cột lần lượt ghi các thông tin về tên tập tin trong hệ thống dữ liệu thực nghiệm chuẩn, số đỉnh, số cạnh và số đỉnh thuộc tập *terminal* của từng đồ thị (các đồ thị này gọi là các đồ thị gốc, các đồ thị *steinc*1, *steind*1, *steine*1 gọi là các đồ thị có kích thước lớn); trọng số của các cạnh là các số nguyên ngẫu nhiên trong phạm vi [1..10].

Bảng 1. *Thông tin các bộ dữ liệu thực nghiệm* [10]

Test	Original problem		
	n	*m*	|*Y*|
steinb1.txt	50	63	9
steinb2.txt	50	63	13
steinb3.txt	50	63	25
steinb4.txt	50	100	9
steinb5.txt	50	100	13
steinb6.txt	50	100	25
steinb7.txt	75	94	13
steinb8.txt	75	94	19
steinb9.txt	75	94	38
steinb10.txt	75	150	13
steinb11.txt	75	150	19
steinb12.txt	75	150	38
steinb13.txt	100	125	17
steinb14.txt	100	125	25
steinb15.txt	100	125	50
steinc1.txt	500	625	5
steind1.txt	1000	1250	5
steine1.txt	2500	3125	5

B. Môi trường thực nghiệm

Các thuật toán được cài đặt bằng ngôn ngữ C++ sử dụng môi trường DEV C++ 5.9.2, CPU(2) Intel(R) Xeon(R) E5- 2660 2.20GHz, RAM 8GB, hệ điều hành Windows 10, 64 bit.

C. Kết quả thực nghiệm và đánh giá

1. Chất lượng các thuật toán

Kết quả thực nghiệm các thuật toán rút gọn đồ thị *SP*, *SP-BS, SP-DG* trên 18 bộ dữ liệu được ghi nhận chi tiết ở bảng 2; thông tin về mỗi thuật toán được ghi nhận ở mỗi ba cột *n, m*, |*Y*|, *reduced* lần lượt ứng với số đỉnh của đồ thị, số cạnh của đồ thị, số đỉnh *terminal* của đồ thị sau rút gọn và tỉ lệ rút gọn được của thuật toán đó. Tỉ lệ này được tính theo công thức $1-\frac{n-|Y|}{n_0-|Y_0|}$; trong đó tử số là thông tin của đồ thị sau rút gọn, mẫu số là thông tin ứng với đồ thị gốc. Cột *b* trong thuật toán *SP-DG* là số cạnh cầu Steiner sau khi phân rã đồ thị.

Dựa vào kết quả ở bảng 2, ta có một số đánh giá sau:

Thuật toán rút gọn đồ thị *SP*, các đồ thị 50 đỉnh có hệ số *n*–|*Y*| giảm xuống không ít hơn 16%, các đồ thị 75 đỉnh có hệ số *n*–|*Y*| giảm xuống không ít hơn 11%, các đồ thị 100 đỉnh có hệ số *n*–|*Y*| giảm xuống không ít hơn 36% và với các đồ thị có kích thước lớn có hệ số *n*–|*Y*| giảm xuống không ít hơn 95%.

Thuật toán rút gọn đồ thị *SP-BS* cho kết quả tốt hơn *SP*; trong đó các đồ thị 50 đỉnh có hệ số *n*–|*Y*| giảm xuống không ít hơn 24%, các đồ thị 75 đỉnh có hệ số *n*–|*Y*| giảm xuống không ít hơn 50%, các đồ thị 100 đỉnh có hệ số *n*–|*Y*| giảm xuống không ít hơn 50% và với các đồ thị có kích thước lớn có hệ số *n*–|*Y*| giảm xuống không ít hơn 96%.

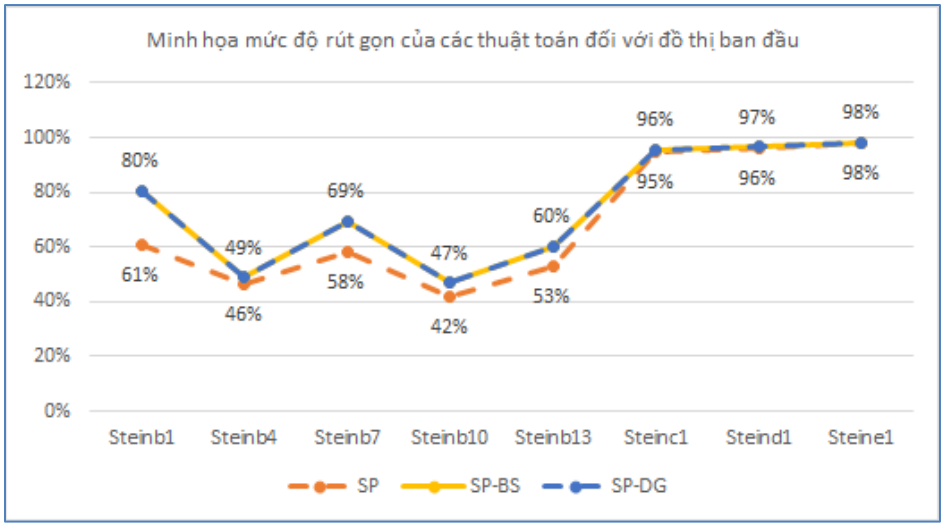
Thuật toán rút gọn đồ thị *SP-DG* cho kết quả tương đương với thuật toán *SP-BS (*đây là lý do chúng tôi xem hai thuật toán này như là một đóng góp).

Bảng 2. Kết quả thực nghiệm các thuật toán *SP*, *SP-BS*, *SP-DG*

Test	*SP*			*SP-BS*				*SP-DG*				
	n	|*Y*|	*reduced*	*n*	*m*	|*Y*|	*reduced*	*n*	*m*	|*Y*|	*b*	*reduced*
steinb1	25	9	61%	25	30	17	80%	16	21	8	9	80%
steinb2	28	13	59%	28	34	18	73%	21	27	11	7	73%
steinb3	41	25	36%	41	52	30	56%	27	38	16	14	56%
steinb4	31	9	46%	31	57	10	49%	30	56	9	1	49%
steinb5	32	13	49%	32	56	14	51%	31	55	13	1	51%
steinb6	46	25	16%	46	92	27	24%	43	89	24	3	24%
steinb7	39	13	58%	39	47	20	69%	30	38	11	9	69%
steinb8	37	19	68%	37	47	21	71%	31	41	15	6	71%
steinb9	62	38	35%	62	81	45	54%	46	65	29	16	54%
steinb10	49	13	42%	49	86	16	47%	46	83	13	3	47%
steinb11	44	19	55%	44	71	20	57%	43	70	19	1	57%
steinb12	71	38	11%	71	142	40	16%	68	139	37	3	16%
steinb13	56	17	53%	55	71	22	60%	47	63	14	8	60%
steinb14	70	25	40%	70	93	35	53%	56	79	21	14	53%
steinb15	82	50	36%	82	107	57	50%	66	91	41	16	50%
steinc1	32	5	95%	32	36	10	96%	27	31	5	5	96%
steind1	43	5	96%	43	48	11	97%	37	42	5	6	97%
steine1	49	5	98%	49	53	8	98%	46	50	5	3	98%

2. Hình vẽ minh họa chất lượng các thuật toán

Hình vẽ 3 minh họa mức độ rút gọn đồ thị của các thuật toán *SP*, *SP-BS*, *SP-DG* (dựa trên thông tin các cột *reduced* của các thuật toán ở bảng 2).



Hình 3. Minh họa mức độ rút gọn được của các thuật toán *SP*, *SP-BS*, *SP-DG*

KẾT LUẬN

Bài báo này đề xuất một thuật toán rút gọn đồ thị cho bài toán *SMT*; đề xuất này đã được thực nghiệm trên một số bộ dữ liệu là đồ thị thưa trong hệ thống dữ liệu thực nghiệm chuẩn. Thuật toán rút gọn đồ thị đề xuất có hiệu quả rút gọn lên đến 98% đối với một số đồ thị lớn. Bài báo này không nhằm mục đích đưa ra một thuật toán rút gọn đồ thị tốt hơn các thuật toán rút gọn đồ thị hiện biết (là vấn đề khó khăn); mà chỉ nhằm bổ sung thêm một cách thức mới để rút gọn đồ thị; càng nhiều cách thức rút gọn thì khả năng rút gọn số cạnh và số đỉnh cho bài toán *SMT* càng cao. Thuật toán rút gọn đồ thị đề xuất cùng với kết quả thực nghiệm của bài báo này là thông tin hữu ích cho các tiếp cận giải đúng và giải gần đúng bài toán *SMT*. Trong thực tế, với các tiếp cận giải gần đúng bài toán *SMT*, chỉ cần vài thuật toán rút gọn tốt thì đồ thị sẽ được rút gọn đáng kể và do đó chất lượng của thuật toán sẽ được nâng lên.

TÀI LIỆU THAM KHẢO

[1]. Ankit Anand, Shruti, Kunwar Ambarish Singh, "An efficient approach for steiner tree problem by genetic algorithm", International Journal of Computer Science and Engineering (SSRG-IJCSE), vol.2, pp.233-237, 2015.

[2]. Bang Ye Wu, Kun-Mao Chao, "Spanning trees and optimization problems", Chapman&Hall/CRC, pp.13-139, 2004.

[3]. Chin Lung Lu, Chuan Yi Tang, Richard Chia-Tung Lee, "The full Steiner tree problem", Elsevier, pp.55-67, 2003.

[4]. Chopra, S., Gorres, E., and Rao, M. R. "Solving a Steiner tree problem on a graph using branch and cut". ORSA Journal on Computing, vol 4, pp.320 – 335, 1992.

[5]. C. C. Ribeiro, M. C. Souza, "Tabu Search for the Steiner Problem in Graphs", Networks, vol.36, pp.138-146, 2000.

[6]. Diego de Una, Graeme Gange, Peter Schachte, Peter J. Stuckey, "Steiner tree problems with side constraints using constraint programming", Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (www.aaai.org), 2016.

[7]. Ding-Zhu Du, J. M.Smith, J.H. Rubinstein, "Advances in Steiner trees", pp.1-322, 2000.

[8]. Edmund Ihler, Gabriele Reich, Peter Widmayer, "Class Steiner trees and VLSI-design", Elsevier, pp.173-194, 1999.

[9]. Gabriel Robins, Alexander Zelikovsky, "Minimum Steiner Tree Construction", (Department of Computer Science, University of Virginia, Department of Computer Science, Georgia State University), pp.1-33, 2008.

[10]. J. E. Beasley, OR-Library: URL http://people.brunel.ac.uk/~mastjjb/jeb/orlib/steininfo.html

[11]. Jeffrey H. Kingston, Nicholas Paul Sheppard, "On reductions for the steiner problem in graphs", Basser Department of Computer Science the University of Sydney, Australia, pp.1-10, 2006.

[12]. Jaros law Byrka, Mateusz Lewandowski, Carsten Moldenhauer, "Approximation algorithms for node-weighted prize-collecting Steiner tree problems on planar graphs", University of Wroc law, Poland, pp.1-23, 2016.

[13]. J. E. Beasley, "An SST-Based Algorithm for the Steiner Problem in Graphs", Networks, vol.19, 1-16, l989.

[14]. Latha N. R, "A study on approximation algorithms for constructing rectilinear Steiner trees", IJCSIT, pp.765-769, 2015.

[15]. L. Kou, G. Markowsky, L. Berman," A Fast Algorithm for Steiner Trees",Acta Informatica,vol.15,pp.141-145, 1981.

[16]. Marcello Caleffi, Ian F. Akyildiz, Luigi Paura, "On the solution of the Steiner tree NP-Hard problem via physarum bionetwork", IEEE/ACM transactions on networking, vol. 23, No. 4, pp.1092-1106, 2015.

[17]. Matjaz Kovse, "Vertex decomposition of Steiner wiener index and steiner betweenness centrality", University of Wroc law, Poland, 2016.

[18]. M. R. Garey R. L. Graham, D. S. Johnson, "The complexity of computing steiner minimal trees", SIAM J. APPL. MATH, vol.32, No 4, pp.835-859, 1977.

[19]. M. Hauptmann, M. Karpinski, "A compendium on Steiner tree problems", Department of Computer Science and Hausdorff Center for Mathematics University of Bonn, pp.1-50, 2015.

[20]. Nguyen Viet Huy, Nguyen Duc Nghia, "Solving graphical Steiner tree problem using parallel genetic algorithm", RIVF, 2008.

[21]. Ondra Suchý, "Exact algorithms for Steiner tree", Faculty of Information Technology, Czech Technical University in Prague, Prague, Czech Republic, 2014.

[22]. Phan Tấn Quốc, "Proposing a new algorithm for solving the minimum routing cost spanning tree problem in sparse graphs", Tạp chí Khoa học Trường Đại học Cần Thơ, ISSN:1859-2333, Số chuyên đề Công nghệ thông tin, pp. 61-68, 2015.

[23]. Stefan Hougardy, Jannik Silvanus, Jens Vygen, "Dijkstra meets Steiner: a fast exact goal-oriented Steiner tree algorithm", Research Institute for Discrete Mathematics, University of Bonn, pp.1-59, 2015.

[24]. S. E. Dreyfus, R. A. Wagner, "The Steiner Problem in Graphs", Networks, vol.1, pp.195-207, 1971.

[25]. Robert Sedgewick, Kevin Wayne, "Algorithms", Fourth edition, Addsion-Wesley, pp.518-700, 2011.

[26]. SangWon Bae, Sunghee Choi, Chunseok Lee, Shin-ichi Tanigawa, "Exact algorithms for the bottleneck Steiner tree problem", Springer-Verlag Berlin Heidelberg, 2009.

[27]. Thomas Bosman, "A solution merging heuristic for the Steiner problem in graphs using tree decompositions", VU University Amsterdam, The Netherlands, pp.1-12, 2015.

[28]. Thorsten Koch, Alexander Martin,"Solving Steiner tree problems in graphs to optimality", Germany, pp.1-31, 1996.

[29]. Tom Decroos, Patrick De Causmaecker, Bart Demoen, "Solving euclidean Steiner tree problems with multi swarm optimization", ACM, GECCO Companion '15, pp.1379-1380, 2015.

[30]. Vũ Đình Hòa, "Bài toán Steiner", http://math.ac.vn

[31]. Xinhui Wang, "Exact algorithms for the Steiner tree problem", doctoral thesis, ISSN 1381-3617, 2008.

[32]. Xiuzhen Cheng, Ding-Zhu Du, "Steiner trees in industry", Kluwer Academic Publishers, vol.5, pp.193-216, 2004.

ON GRAPH REDUCTIONS FOR THE STEINER MINIMAL TREE PROBLEM

Phan Tan Quoc

ABSTRACT*—Steiner Minimal Tree (SMT) is a combinatorial optimization problem that has many important applications in science and engineering; the SMT problem is proved to be NP-hard and is currently being extensively research. For the approach in solving the SMT problem - whether it is the exact or approximate algorithms - the graph reduction stage is very important;* it *is especially meaningful for the exact algorithms of the SMT problem. Over the years, many graph reduction methods to the problem of SMT have been published; these are positive support for the resolution of SMT problems with the graph of larger size. This paper proposes a graph reduction algorithm for SMT problem and the experimental results are useful information for the study of the exact algorithms and approximate algorithms solutions to the SMT problem.*