Tạp chí Khoa học Trường Đại học Cần Thơ
website: sj.ctu.edu.vn

ĐỀ XUẤT THUẬT TOÁN MỚI GIẢI BÀI TOÁN CÂY KHUNG VỚI CHI PHÍ ĐỊNH TUYẾN NHỎ NHẤT TRONG TRƯỜNG HỢP ĐỒ THỊ THƯA

Phan Tấn Quốc[1]
[1] *Khoa Công nghệ Thông tin, Trường Đại học Sài Gòn*

Thông tin chung:
Ngày nhận: 19/09/2015
Ngày chấp nhận: 10/10/2015

Title:
Proposing a new algorithm for solving the minimum routing cost spanning tree problem in sparse graphs

Từ khóa:
Bài toán cây khung với chi phí định tuyến nhỏ nhất, metaheuristic, heuristic, MRCST

Keywords:
Minimum routing cost spanning tree, metaheuristic, heuristic, MRCST

ABSTRACT

Minimum routing cost spanning tree problem - MRCST is a graph optimization problem that has many applications in network communication design and bioinformatics. The MRCST problem is proved to be NP-hard. Most graphs in practical are sparse graphs while the most effective algorithm solving MRCST on sparse graph is not really effective - especially with sparse graphs in large size. This paper proposes a new algorithm called HCST to solve the MRCST problem for sparse graphs. The experimental results in sparse graphs taken from the experimental data system benchmark for optimal spanning tree problems show that the quality of HCST algorithm is equal or better than that of the best algorithms known to solve MRCST in time comparison. This is also the first publication which presents theexperimental results from solving MRCST for 20 large size sparse graphs instances.

TÓM TẮT

Bài toán cây khung với chi phí định tuyến nhỏ nhất (Minimum Routing Cost Spanning Tree - MRCST) là bài toán tối ưu đồ thị có nhiều ứng dụng trong lĩnh vực thiết kế mạng truyền thông và trong tin sinh học; đây là bài toán thuộc lớp NP-hard. Hầu hết các đồ thị gặp trong thực tế ứng dụng là đồ thị thưa, trong khi các thuật toán hiệu quả nhất hiện biết giải bài toán MRCST trên đồ thị thưa chưa thực sự hiệu quả - nhất là với các đồ thị thưa có kích thước lớn. Bài báo này đề xuất một thuật toán mới với tên gọi HCST để giải bài toán MRCST trong trường hợp đồ thị thưa. Kết quả thực nghiệm trên các đồ thị thưa trong hệ thống dữ liệu thực nghiệm chuẩn cho thấy thuật toán HCST cho chất lượng lời giải tương đương hoặc tốt hơn và với thời gian tính nhanh hơn khi so với các thuật toán tốt nhất hiện biết. Bài báo này cũng là công trình đầu tiên công bố kết quả thực nghiệm giải bài toán MRCST cho 20 bộ dữ liệu là các đồ thị thưa có kích thước lớn.

1 GIỚI THIỆU

1.1 Một số định nghĩa

Cho $G = (V, E)$ là một đồ thị vô hướng, liên thông, có trọng số không âm trên cạnh; trong đó V là tập gồm n đỉnh, E là tập gồm m cạnh, $w(e)$ là trọng số của cạnh e, $e \in E$ (trọng số trên cạnh e được hiểu như chi phí định tuyến của cạnh e).

Định nghĩa 1 (*Chi phí định tuyến giữa một cặp đỉnh* [4]). Cho T là một cây khung của G, ta gọi chi phí định tuyến (routing cost) của một cặp đỉnh (u,v) trên T, ký hiệu là $d_T(u,v)$, là tổng chi phí định tuyến của các cạnh trên đường đi đơn nối đỉnh u với đỉnh v trên cây T.

Định nghĩa 2 (*Chi phí định tuyến của một cây khung* [4]). Cho T là một cây khung của G, chi phí

định tuyến của T, ký hiệu là $C(T)$, là tổng chi phí định tuyến giữa mọi cặp đỉnh thuộc cây T, tức là:

$$C(T) = \sum_{u,v \in V(T)} d_T(u,v). \quad (1)$$

Bài toán đặt ra là trong số tất cả các cây khung của đồ thị G cần tìm cây khung có chi phí định tuyến nhỏ nhất. Bài toán này được đặt tên là bài toán cây khung với chi phí định tuyến nhỏ nhất (Minimum Routing Cost Spanning Tree-*MRCST*). Bài toán *MRCST* đã được chứng minh thuộc lớp bài toán *NP-hard* [4].

Trên cơ sở khái niệm tải định tuyến (routing load), công trình [4] đã chỉ ra cách tính chi phí định tuyến của một cây khung với độ phức tạp tuyến tính bằng cách sử dụng cấu trúc cây có gốc.

Định nghĩa 3 (*Tải định tuyến một cạnh của cây khung* [4]**)** Cho T là một cây khung của đồ thị G. Nếu loại khỏi cây T một cạnh e thì T sẽ được tách ra thành hai cây con T_1 và T_2 với hai tập đỉnh tương ứng ký hiệu là $V(T_1)$ và $V(T_2)$. Ta gọi tải định tuyến của cạnh e, ký hiệu là $l(T,e)$, là giá trị $2|V(T_1)|\times|V(T_2)|$.

Từ định nghĩa, dễ thấy tải định tuyến của cạnh e chính là bằng số lượng đường đi trên cây T có chứa cạnh e.

Định lý sau cho ta cách tính chi phí định tuyến của cây khung thông qua tải định tuyến của các cạnh.

Định lý. Cho T là một cây khung của G, ta có:

$$C(T) = \sum_{e \in E(T)} l(T,e) \times w(e) \quad (2)$$

và chi phí định tuyến của cây khung T có thể tính được trong thời gian $O(n)$ [4].

Có thể tìm thấy các ứng dụng của bài toán *MRCST* trong lĩnh vực thiết kế mạng truyền thông và trong tin sinh học [4,7,11,12].

1.2 Các nghiên cứu liên quan

Hiện đã có hơn hai mươi công trình giải bài toán *MRCST* được đề xuất theo các hướng tiếp cận giải đúng, giải gần đúng cận tỉ lệ, heuristic và metaheuristic.

Các thuật toán giải đúng phát triển dựa trên phương pháp nhánh cận kết hợp phương pháp sinh cột được đề xuất bởi Matteo Fischetti, Giuseppe Lancia, Paolo Serafini vào năm 2002 [6]; tiếp cận hướng này chỉ có thể giải được các bài toán *MRCST* có không quá 30 đỉnh và do đó tính ứng dụng thực tiễn của nó không cao.

Một thuật toán gần đúng cận tỉ lệ α nghĩa là trong trường hợp xấu nhất thì lời giải tìm được của thuật toán đó cũng không tệ hơn α lần so với lời giải tối ưu; tuy nhiên các 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 [3,4] – điển hình theo hướng này là thuật toán *WONG* với cận tỉ lệ 2 và có độ phức tạp thời gian tính $O(nm + n^2 \log n)$ [4]; đây là thuật toán định tuyến mạng chạy ổn định trên mọi loại dữ liệu và đã được sử dụng trong thực tế ứng dụng [4,11].

Các thuật toán heuristic có chất lượng lời giải chấp nhận được với một loại dữ liệu cụ thể nào đó của bài toán; chẳng hạn như thuật toán *ADD* được đề xuất bởi Vic Grout vào năm 2005 [12] với độ phức tạp $O(n \log n)$ và chỉ tương đối hiệu quả với các đồ thị đồng nhất có kích thước nhỏ [9], thuật toán *CAMPOS* được đề xuất bởi Rui Campos và Manuel Ricardo vào năm 2008 [11] với độ phức tạp $O(m + n \log n)$ và chỉ hiệu quả đối với các đồ thị đầy đủ euclid [9]; cả hai heuristic này đều không hiệu quả đối với các đồ thị thưa về chất lượng lời giải [9]; tuy nhiên đây là hai thuật toán nhanh nhất hiện nay trong việc giải bài toán *MRCST;* điều này có ý nghĩa quan trọng; đặc biệt là trong các mạng viễn thông di động, nơi mà cấu trúc của các mạng thường xuyên thay đổi và cần phải có một cơ chế hiệu quả để tìm *MRCST* mới trong thời gian nhanh nhất mặc dù có thể chấp nhận một mức chi phí nhiều hơn.

Các thuật toán metaheuristic giải bài toán *MRCST* đã được đề xuất như các thuật toán di truyền *ESCGA* [5], *BCGA* [5], các thuật toán tìm kiếm địa phương *SHC* [5], *PBLS* [1], *REPRI* [8], *REPIR* [8], *TABU* [10], các thuật toán ong *PABC* [2], *ABC+LS* [2], *BEE* [9],... Hiện tại, các tiếp cận giải bài toán *MRCST* theo hướng metaheuristic là hiệu quả nhất [9]; tuy nhiên các thuật toán trên chỉ hiệu quả đối với loại đồ thị đầy đủ euclid và đồ thị ngẫu nhiên có kích thước nhỏ (tất cả các thực nghiệm đã công bố hiện nay với đồ thị đầy đủ đều không quá 300 đỉnh); các metaheuristic gần đây nhất như *PABC*, *ABC+LS*, *BEE* không hiệu quả về thời gian tính và cả chất lượng lời giải đối với các đồ thị thưa – nhất là với các đồ thị thưa có kích thước lớn.

Trong số các thuật toán đã khảo sát trên, chỉ có các thuật toán *WONG, SHC, PBLS, REPRI, REPIR* là khả thi đối với các đồ thị thưa có kích thước lớn

(khả thi về thời gian chạy). Tiếp theo, chúng tôi sẽ trình bày ngắn gọn ý tưởng của các thuật toán này.

Thuật toán WONG: Tìm tất cả các cây đường đi ngắn nhất (*Shortest Path Tree - SPT*) [4] có gốc lần lượt tại các đỉnh của đồ thị, sau đó chọn ra trong số chúng một cây đường đi ngắn nhất có chi phí định tuyến nhỏ nhất [4].

Thuật toán SHC: Trước hết khởi tạo một cây khung lời giải ngẫu nhiên; sau đó từng bước cải thiện dần chất lượng lời giải. Ở mỗi bước lặp, *SHC* sinh ra một hoặc một số lời giải lân cận và sau đó chọn một lời giải tốt nhất trong số đó để làm lời giải hiện tại cho bước lặp kế tiếp. Việc tìm kiếm lân cận *T*' của cây khung *T* được thực hiện như sau: Loại ngẫu nhiên một cạnh *e* thuộc *T*, sau đó tìm cạnh *e*' tốt nhất từ tập cạnh $E - T$ để thay thế cho *e* (trong ký hiệu $E - T$; *E* là tập cạnh của đồ thị *G*, còn *T* là tập cạnh của cây khung *T*. Ký hiệu $G - T$ cũng được hiểu tương tự). Một lần lặp của thuật toán *SHC* đòi hỏi thời gian tính $O(mn)$. Thuật toán *SHC* ấn định số bước lặp là $10000n$. Vậy *SHC* đòi hỏi thời gian tính cỡ $10000n \times mn = 10000mn^2$ đối với mỗi bộ dữ liệu [5].

Thuật toán PBLS: Điểm khác biệt duy nhất của thuật toán *PBLS* so với thuật toán *SHC* là *PBLS* có sử dụng chiến lược đa dạng hóa lời giải - theo nghĩa là thay một số cạnh của cây khung bằng một số cạnh ngẫu nhiên khác. Thuật toán *PBLS* ấn định số bước lặp là 50000 và khi chất lượng lời giải không được cải thiện sau $2n$ bước lặp thì tiến hành đa dạng hóa lời giải. Một lần lặp của thuật toán *PBLS* đòi hỏi thời gian tính $O(mn)$. Vậy *PBLS* đòi hỏi thời gian tính cỡ $50000 \times mn = 50000mn$ đối với mỗi bộ dữ liệu [1].

Thuật toán REPRI: Bắt đầu từ một cây khung *T* được khởi tạo bằng kết quả của thuật toán *WONG*. Chiến lược tìm kiếm lân cận: Loại lần lượt từng cạnh $e \in T$, với mỗi cạnh *e* như vậy, tìm một cạnh $e' \in E - T$ sao cho $T' = (T - \{e\}) \cup \{e'\}$ có chi phí định tuyến nhỏ nhất. Nếu $C(T') < C(T)$ thì thay *T* bằng *T'* (thay cạnh *e* trong *T* bằng cạnh *e'* trong $E - T$). Thuật toán dừng nếu trong một lần duyệt qua tất cả các cạnh $e \in T$ mà không tìm được cạnh *e*' để cải thiện chi phí định tuyến của cây khung *T* [8].

Thuật toán REPIR: Bắt đầu từ một cây khung *T* được khởi tạo bằng cây khung nhỏ nhất theo thuật toán *KRUSKAL* hoặc thuật toán *PRIM*. Chiến lược tìm kiếm lân cận: Chèn lần lượt từng cạnh $e \in E-T$ vào cây khung *T,* khi đó $T \cup \{e\}$ sẽ chứa một chu trình, tìm một cạnh *e'* trên chu trình này sao cho việc loại nó dẫn đến cây khung *T'* có chi phí định tuyến là nhỏ nhất. Nếu $C(T') < C(T)$ thì thay *T* bằng *T'* (hoán đổi cạnh *e* trong $G - T$ với cạnh *e'* trong *T*). Thuật toán dừng nếu trong một lần duyệt qua tất cả các cạnh $e \in G - T$ mà không cải thiện được chi phí định tuyến của cây khung *T* [8].

1.3 Vấn đề tồn tại cần giải quyết

Tất cả các thuật toán đã khảo sát trên đều có chung cách làm là khi cần thay thế một cạnh nào đó thì phải duyệt qua tập tất cả các cạnh ứng viên để tìm ra cạnh tốt nhất; chính điều này làm cho các thuật toán trên không khả thi về thời gian khi số cạnh của đồ thị là đủ lớn.

Bài báo này đề xuất một thuật toán *HCST* dạng tìm kiếm leo đồi giải bài toán *MRCST*. Đề xuất này luôn cho chất lượng lời giải tốt hơn hoặc tương đương với các thuật toán tốt nhất hiện biết giải bài toán *MRCST* trên các đồ thị thưa. Đề xuất này thực sự có ý nghĩa đối với các đồ thị thưa có kích thước lớn.

2 ĐỀ XUẤT THUẬT TOÁN *HCST* GIẢI BÀI TOÁN *MRCST* TRONG TRƯỜNG HỢP ĐỒ THỊ THƯA

Thuật toán đề xuất *HCST* dựa vào chiến lược tìm kiếm lân cận như đã được đề cập trong *REPIR* [8]*; HCST* được bổ sung một số chiến lược mới nhằm tăng chất lượng lời giải và làm cho thuật toán khả thi khi làm việc với các đồ thị thưa có kích thước lớn.

2.1 Tạo lời giải ban đầu bằng thuật toán *tựa PRIM*

Khác với thuật toán *REPIR* cho khởi tạo cây khung ban đầu bằng thuật toán tìm cây khung nhỏ nhất của đồ thị theo thuật toán *KRUSKAL* hoặc thuật toán *PRIM*; *HCST* cho khởi tạo cây khung ngẫu nhiên bằng thuật toán *tựa* PRIM: Bắt đầu từ cây chỉ gồm một đỉnh nào đó của đồ thị, tiếp theo, thuật toán sẽ thực hiện n-1 bước lặp. Ở mỗi bước lặp, trong số các đỉnh chưa được chọn để tham gia vào cây, ta chọn một đỉnh kề với ít nhất một đỉnh nằm trong cây đang được xây dựng mà không quan tâm đến trọng số của cạnh. Đỉnh được chọn và cạnh nối nó với đỉnh của cây đang được xây dựng sẽ được bổ sung vào cây [9].

2.2 Chiến lược tìm kiếm lân cận

HCST thực hiện chiến lược tìm kiếm lân cận như đã trình bày trong thuật toán *REPIR* [8].

2.3 Giới hạn kích thước tập cạnh ứng viên

Trong các giáo trình về lý thuyết đồ thị và cấu trúc dữ liệu, nếu một đồ thị có số lượng cạnh lớn

hơn 5 lần số lượng đỉnh thì có thể xem đó là đồ thị dày. Với các đồ thị dày thì *HCST* làm việc như sau: Thay vì duyệt tất cả các cạnh ứng viên để tìm ra một cạnh tốt nhất thì *HCST* chỉ cần duyệt trong một tập con (tối đa 5*n* cạnh) ngẫu nhiên gồm các cạnh được chọn từ tập cạnh ứng viên. Trong khi đó, trước đây, thuật toán *REPIR* làm việc kém hiệu quả đối với các đồ thị dày có nhiều đỉnh.

2.4 Tăng tính ngẫu nhiên cho thuật toán

Hiệu quả của thuật toán *HCST* có thể được cải thiện khi ta thay đổi thứ tự các cạnh được duyệt trong tập $E - T$; nghĩa là ta sẽ duyệt tập cạnh này theo một hoán vị được sinh ngẫu nhiên chứ không theo một thứ tự cố định ở tất cả các lần duyệt (như thuật toán *REPIR*).

2.5 Cho thực hiện thuật toán nhiều lần trên mỗi bộ dữ liệu

Thuật toán *REPIR* thực hiện một lần duy nhất đối với mỗi bộ dữ liệu (do lời giải khởi tạo của *REPIR* không có tính ngẫu nhiên nên việc chạy nhiều lần sẽ không có tác dụng). Thuật toán *HCST* cho thực hiện nhiều lần đối với mỗi bộ dữ liệu.

2.6 Sơ đồ thuật toán *HCST*

Thuật toán *REPIR* [8] chỉ nêu ngắn gọn ý tưởng chiến lược tìm kiếm lân cận và không phân tích độ phức tạp tính toán của thuật toán. Trong bài báo này chúng tôi đã trình bày chi tiết nội dung thuật toán *HCST* và phân tích độ phức tạp tính toán của thuật toán *HCST* một cách chi tiết.

HCST (V,E)

Đầu vào: Đồ thị $G=(V, E)$

Đầu ra: Cây khung có chi phí định tuyến nhỏ nhất tìm được

1. stop=false;
2. T là cây khung được khởi tạo ngẫu nhiên bằng thuật toán *tựa PRIM*;
3. **while** (!stop){
4. stop=true;
5. $S= E - T$;
6. **for** (với mỗi cạnh $e \in S$){
7. $min= +\infty$;
8. $F=T \cup \{e\}$; // F có một chu trình
9. Xác định chu trình *Cycle* trong F chứa cạnh e;
10. **for** (với mỗi cạnh $e' \in Cycle$)
11. **if** ($C(F - \{e'\}) < min$){
12. $min= C(F - \{e'\})$;
13. Ghi nhận $T'= F - \{e'\}$;
14. }
15. **if** ($min < C(T)$){
16. $T=T'$;
17. stop=false;
18. } // end if dòng 15
19. } // end for dòng 6
20. } // end while dòng 3
21. **return** T;

2.7 Đánh giá độ phức tạp của *HCST*

Độ phức tạp một lần lặp của thuật toán *HCST* được đánh giá bởi thời gian tính hai vòng lặp **for** lồng nhau ở dòng 6 và 10. Câu lệnh **if** dòng 11 có độ phức tạp $O(n)$, vòng lặp **for** ở dòng 10 chứa câu lệnh **if** trên lặp $O(n)$ lần; do đó có độ phức tạp là $O(n^2)$. Thao tác tìm các cạnh trong chu trình ở dòng 9 có độ phức tạp $O(n)$. Vòng lặp **for** ở dòng 6 lặp m-(n-1) lần. Do đó vòng **for** này có độ phức tạp là $O(n^2)$. Gọi k là giá trị lớn nhất mà vòng lặp **while** ở dòng 3 có thể thực hiện, thì độ phức tạp một lần lặp của thuật toán *HSCT* là $O(kn^2m)$.

Trong khi các thuật toán *SHC*, *PBLS* đưa ra số lần lặp lần lượt là 10000 n^2m và 50000 nm để đạt được kết quả như công bố (đã được khảo sát ở trên); thì thuật toán *HCST* với cách thức tìm lân cận đã nêu có giá trị k khá nhỏ. Với đồ thị thưa; chẳng hạn là 1000 đỉnh, qua thống kê trên các thực nghiệm của chúng tôi thì giá trị k tối đa chỉ bằng 10. Thời gian tính của các thuật toán *HCST* nhanh hơn so với nhiều thuật toán metaheuristic khác, đặc biệt là khi làm việc với các đồ thị thưa có kích thước lớn.

3 THỰC NGHIỆM VÀ ĐÁNH GIÁ

Chúng tôi tiến hành thực nghiệm thuật toán *HCST* và so sánh chất lượng lời giải và thời gian tính của *HCST* với các thuật toán tốt nhất hiện biết giải bài toán *MRCST* trong trường hợp đồ thị thưa như *WONG, CAMPOS, SHC, PBLS*.

Do hiện tại chưa có một công trình nào công bố chất lượng lời giải đối với các đồ thị thưa có kích thước lớn, nên chúng tôi đã cài đặt lại các thuật toán *WONG, SHC, PBLS* trên cùng môi trường thực nghiệm với thuật toán *HCST*. Để kiểm tra chất lượng các chương trình do chúng tôi cài đặt, chúng tôi đã đối sánh output từ các chương trình do chúng tôi cài đặt với output của các công trình khác đối với các đồ thị đầy đủ euclid và đồ thị đầy đủ ngẫu nhiên. Riêng thuật toán *CAMPOS* chúng tôi được

cung cấp chương trình nguồn bởi tác giả công trình [11].

3.1 Hệ thống dữ liệu thực nghiệm

Trong bài báo này, chúng tôi sử dụng tổng cộng 20 bộ dữ liệu là các đồ thị thưa được lấy từ website lưu trữ hệ thống dữ liệu thực nghiệm chuẩn cho các bài toán cây khung tối ưu: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files.

Bảng 1 mô tả thông tin về các bộ dữ liệu thực nghiệm: Tên tập tin lưu trữ các bộ dữ liệu, số thứ tự bộ test trong mỗi tập tin; số đỉnh và số cạnh của đồ thị tương ứng.

Bảng 1: Thông tin các bộ dữ liệu thực nghiệm

Tập tin	Số thứ tự	*n*	*m*
steinc.txt	11-15	500	2500
steind.txt	11-15	1000	5000
steine.txt	11-15	2500	12500
steine.txt	16-20	2500	62500

3.2 Môi trường thực nghiệm

Thuật toán *HCST* được cài đặt bằng ngôn ngữ C^{++} sử dụng môi trường DEV C 5.0, CPU Pentium ® Dual–Core, E6500, 2.93 GHz, RAM 2048 MB, hệ điều hành Windows 7 Professional 32 bit.

3.3 Tham số thực nghiệm

Thuật toán *HCST* thực hiện 120 lần đối với mỗi bộ dữ liệu loại steinc, steind; 60 lần đối với mỗi bộ dữ liệu loại steine11-steine15 và 30 lần đối với mỗi bộ dữ liệu loại steine16-steine20. Với mỗi bộ dữ liệu steine16..steine20, tham số kích thước của tập cạnh ứng viên được chọn là $N = 5n$; đây chính là ngưỡng phân biệt đồ thị thưa và đồ thị dày. Các tham số này được lựa chọn dựa trên kết quả chạy thực nghiệm với một số bộ dữ liệu được trích chọn từ các bộ dữ liệu thực nghiệm chuẩn.

Các thuật toán *SHC*, *PBLS* sử dụng bộ tham số được đề nghị từ các công trình [1,5]; chỉ riêng tham số *số lần lặp* tìm kiếm lân cận trong thuật toán *SHC* được chúng tôi đề nghị là 100000 (do số lần thực hiện việc tìm kiếm lân cận mà tác giả công trình [5] đề nghị là quá lớn: $10000n^2m$). Hai thuật toán *WONG*, *CAMPOS* được cho thực hiện một lần duy nhất.

3.4 Kết quả thực nghiệm và đánh giá

3.4.1 Chất lượng lời giải

Kết quả thực nghiệm của các thuật toán *WONG*, *CAMPOS, SHC, PBLS, HCST* trên 20 bộ dữ liệu đồ thị thưa được ghi nhận chi tiết ở Bảng 2 và Bảng 3; trong đó các thuật toán *WONG* và *CAMPOS* ghi nhận các giá trị chi phí định tuyến và thời gian thực hiện tương ứng (Best, Time); các thuật toán *SHC, PBLS, HCST* ghi nhận chi phí định tuyến tốt nhất, thời gian trung bình một lần chạy, chi phí định tuyến trung bình và độ lệch chuẩn sau các lần chạy mỗi bộ dữ liệu (Best, Time, Mean, SD).

Bảng 2: Kết quả thực nghiệm các thuật toán WONG, CAMPOS, SHC trên đồ thị thưa

Test	*WONG*		*CAMPOS*		*SHC*			
	Best	**Time**	**Best**	**Time**	**Best**	**Time**	**Mean**	**SD**
steinc-11	1704347	0.328	1717399	0.002	1686517	64.3	1724650.5	50840.7
steinc-12	1780241	0.359	1867361	0.002	1757608	63.6	1818089.6	68275.7
steinc-13	1755973	0.297	1828029	0.002	1737046	69.6	1741048.3	4280.4
steinc-14	1698543	0.329	1822326	0.002	1686765	65.1	1746846.4	77025.7
steinc-15	1720180	0.304	1844404	0.002	1705026	65.6	1786204.3	75980.3
steind-11	7952947	1.359	8305895	0.005	7893906	149.6	8321488.5	404702.2
steind-12	7838330	1.422	8591643	0.005	7754894	151.6	7915906.4	184314.9
steind-13	7594958	1.391	8854375	0.005	7544222	152.8	7726898.5	320037.3
steind-14	7669320	1.375	7653716	0.005	7623099	148.7	7972443.7	247386.4
steind-15	7408711	1.399	8948301	0.005	7375307	149.8	7666580.4	316608.6
steine-11	56146228	8.845	56896282	0.021	55836577	456.2	58189380.3	1768421.5
steine-12	55866042	8.766	62387839	0.021	55529655	458.2	58067574.3	2249273.6
steine-13	55924075	8.828	65221501	0.021	55444595	467.5	58246409.5	1940857.9
steine-14	55806878	8.876	57372344	0.020	55472368	526.7	57865660.6	2240762.2
steine-15	53970547	8.748	61006740	0.020	53645750	460.7	57237681.6	2635969.4
steine-16	24799865	10.703	25907752	0.031	25194136	2024.5	25976539.6	558738.1
steine-17	24520680	10.469	27478296	0.030	24926161	1965.7	25511900.7	381470.7
steine-18	24624612	10.860	26910649	0.030	24792320	2051.3	25589496.0	600738.5
steine-19	24905297	10.750	27127729	0.030	24507129	2010.8	25963992.3	855882.6
steine-20	24928266	10.453	26920558	0.030	24712530	1960.8	25667553.9	779055.9

Bảng 3: Kết quả thực nghiệm các thuật toán PBLS, HCST trên đồ thị thưa

Test	*PBLS*				*HCST*			
	Best	**Time**	**Mean**	**SD**	**Best**	**Time**	**Mean**	**SD**
steinc-11	1686517	32.7	1717990.9	36696.3	1686517	4.5	1720235.2	39181.4
steinc-12	1757608	32.2	1825533.1	65953.5	1757608	4.1	1829371.2	71099.0
steinc-13	1737046	34.8	1741192.5	6497.3	1737046	5.3	1741543.8	9020.2
steinc-14	1686765	33.5	1738269.9	67697.2	1686765	4.0	1734371.0	69878.6
steinc-15	1705026	33.3	1786707.6	65190.2	1705026	4.8	1804119.7	75948.4
steind-11	7893905	75.3	8366693.7	369261.8	7893905	25.9	8216646.1	324215.0
steind-12	7754894	77.0	8007221.1	268238.6	7754894	26.6	7938324.0	238412.4
steind-13	7544222	77.3	7856117.0	378010.3	7544222	28.4	7794287.7	359951.4
steind-14	7623099	76.3	8003156.4	253849.1	7623099	22.9	7953101.4	220923.8
steind-15	7375307	75.7	7763717.9	360355.4	7375307	25.4	7731268.7	374296.1
steine-11	55836577	232.5	59055464.3	2097113.0	55836577	211.2	57893046.9	1769743.3
steine-12	55538602	233.6	59071697.0	2459137.7	55529655	236.5	57495152.2	2369165.1
steine-13	55464606	237.4	58776296.1	2129813.8	55444595	288.3	57211399.0	1669583.5
steine-14	55472368	229.9	59409258.3	2923580.4	55472368	211.0	57668606.6	2226082.7
steine-15	53670509	231.9	57905340.9	2720464.5	53645750	221.9	56897370.5	2498028.3
steine-16	24700731	1003.0	25995852.2	804225.6	24685730	715.3	26008465.7	676451.7
steine-17	24793249	999.9	25791919.1	734408.8	24429309	811.1	25485799.1	586067.4
steine-18	24878203	1005.5	26042029.3	601822.0	24548694	821.1	25761859.1	836149.5
steine-19	25351609	1059.6	26046522.1	396944.0	24532631	757.7	25589437.9	635616.4
steine-20	25064031	996.1	26120580.8	601714.9	24625014	843.6	25623622.4	704946.4

Từ kết quả thực nghiệm trên, có thể đánh giá về chất lượng lời giải của thuật toán *HCST* so với các thuật toán *WONG*, *CAMPOS*, *SHC*, *PBLS*.

Thuật toán *HCST* cho chất lượng lời giải tốt hơn các thuật toán *WONG*, *CAMPOS* trên 100% bộ dữ liệu.

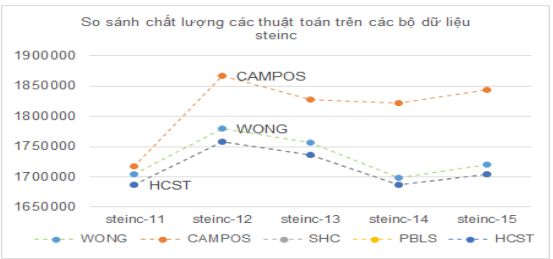
Thuật toán *HCST* cho chất lượng lời giải tốt hơn thuật toán *SHC* trên 25% bộ dữ liệu, bằng thuật toán *SHC* trên 70% bộ dữ liệu và kém hơn thuật toán *SHC* trên 5% bộ dữ liệu.

Thuật toán *HCST* cho chất lượng lời giải tốt hơn thuật toán *PBLS* trên 40% bộ dữ liệu và bằng thuật toán *PBLS* trên 60% bộ dữ liệu.

Các thuật toán *SHC, PBLS, HCS* cho chất lượng lời giải trung bình nhìn chung là kém hơn thuật toán *WONG*. Thuật toán *WONG* có chất lượng lời giải tốt hơn thuật toán *CAMPOS*.

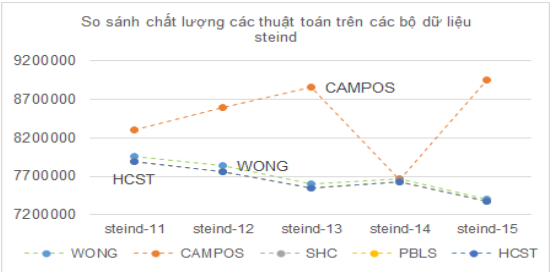
Tiếp theo chúng tôi sẽ so sánh chất lượng lời giải của thuật toán *HCST* và các thuật toán đã khảo sát trên qua một số hình vẽ; trong đó chi phí định tuyến đem so sánh là chi phí định tuyến tốt nhất.

Các thuật toán *SHC, PBLS, HCST* cho chất lượng lời giải như nhau trên các đồ thị steinc.



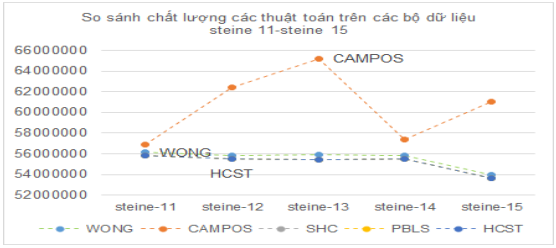
Hình 1: Chất lượng lời giải của các thuật toán qua các bộ dữ liệu steinc

Các thuật toán *PBLS, HCST* cho chất lượng lời giải bằng nhau trên các bộ dữ liệu steind và cùng cho chất lượng tốt hơn thuật toán *SHC* trên một bộ dữ liệu.



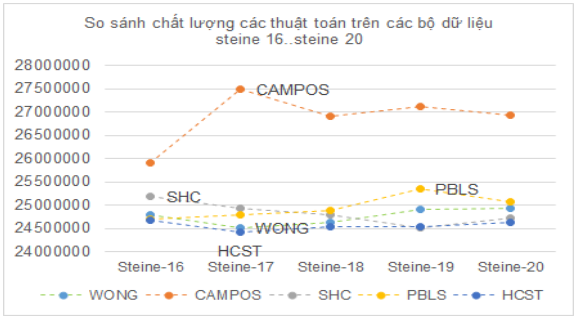
Hình 2: Chất lượng lời giải của các thuật toán qua các bộ dữ liệu steind

Các thuật toán *SHC, HCST* cho chất lượng lời giải bằng nhau trên các bộ dữ liệu steine11..steine15 và cùng cho chất lượng tốt hơn thuật toán *PBLS* trên ba bộ dữ liệu.



Hình 3: Chất lượng lời giải của các thuật toán qua các bộ dữ liệu steine11..steine15

Thuật toán *HCST* cho chất lượng lời giải tốt hơn thuật toán *SHC* trên bốn bộ dữ liệu thuộc steine16..steine20 và kém hơn trên một bộ dữ liệu còn lại; thuật toán *HCST* luôn luôn cho lời giải có chất lượng tốt hơn thuật toán *PBLS* trên các bộ dữ liệu steine16..steine20.



Hình 4: Chất lượng lời giải của các thuật toán qua các bộ dữ liệu steine16..steine20

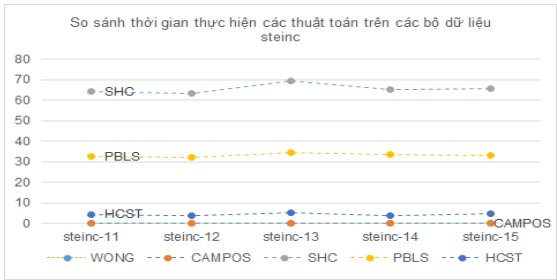
3.4.2 Thời gian tính

Thời gian tính được ghi nhận là thời gian trung bình của các lần chạy. Từ kết quả thực nghiệm của Bảng 2 và Bảng 3 chỉ ra rằng:

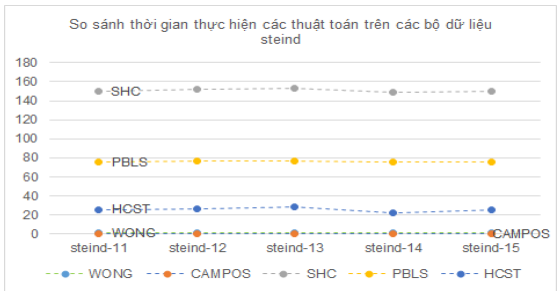
Thuật toán *HCST* có thời gian tính không quá 7.6% thời gian tính của thuật toán *SHC* đối với các đồ thị 500 đỉnh, không quá 18.6% đối với các đồ thị 1000 đỉnh, không quá 43.0% đối với các đồ thị 2500 đỉnh steine16..steine20 và không quá 61.7% đối với các đồ thị 2500 đỉnh steine11..steine15.

Thuật toán *HCST* có thời gian tính không quá 15.2% thời gian tính của thuật toán *PBLS* đối với các đồ thị 500 đỉnh, không quá 36.7% đối với các đồ thị 1000 đỉnh, không quá 84.7% đối với các đồ thị 2500 đỉnh steine16..steine20, và có thời gian tính tương đương đối với các đồ thị 2500 đỉnh steine11..steine15.

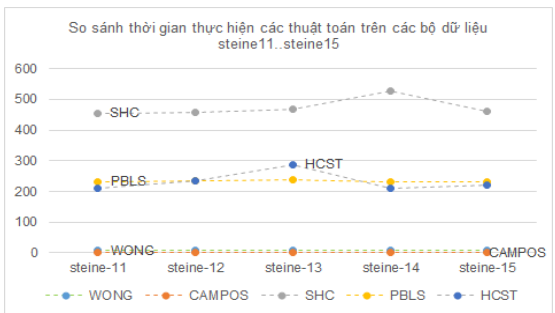
Tiếp theo chúng tôi sẽ so sánh thời gian tính của thuật toán *HCST* với các thuật toán đã khảo sát trên qua một số hình vẽ.



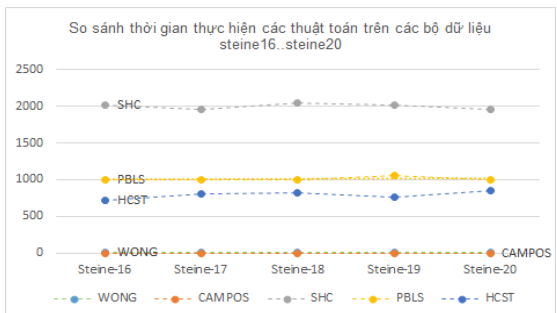
Hình 5: Thời gian tính của các thuật toán qua các bộ dữ liệu steinc



Hình 6: Thời gian tính của các thuật toán qua các bộ dữ liệu steind



Hình 7: Thời gian tính của các thuật toán qua các bộ dữ liệu steine11.. steine15



Hình 8: Thời gian tính của các thuật toán qua các bộ dữ liệu steine16..steine20

Chúng tôi có tạo website để trao đổi với cộng đồng nghiên cứu bài toán *MRCST* tại địa chỉ https://sites.google.com/site/mrcsttest; website này giới thiệu các công trình nghiên cứu liên quan đến bài toán *MRCST*, hệ thống dữ liệu thực nghiệm chuẩn và kết quả thực nghiệm liên quan đến bài báo này:https://sites.google.com/site/mrcsttest/home/experimental-results.

4 KẾT LUẬN

Bài báo đã khảo sát các thuật toán tốt nhất hiện biết giải bài toán MRCST trong trường hợp đồ thị thưa và qua đó đề xuất một thuật toán mới hiệu quả để giải bài toán MRCST trong trường hợp đồ thị thưa. Bài báo này cũng là công trình đầu tiên công bố kết quả thực nghiệm giải bài toán MRCST trên các đồ thị thưa có kích thước lớn.

LỜI CẢM TẠ

Tác giả xin trân trọng cảm ơn thầy giáo Nguyễn Đức Nghĩa, Bộ môn Khoa học máy tính, Viện Công nghệ Thông tin và Truyền thông, Trường Đại học Bách khoa Hà Nội đã giúp tôi hoàn chỉnh một số ý tưởng để viết bài báo này.

TÀI LIỆU THAM KHẢO

1. Alok Singh (2008). *A new heuristic for the minimum routing cost spanning tree problem*. International Conference on Information Technology-ICIT,IEEE, Bhubaneswar,India,pp.9-13.
2. Alok Singh, Shyam Sundar (2011). *An artificial bee colony algorithm for the minimum routing cost spanning tree problem*. Soft Computing, volume 15 (12), Springer-Verlag, pp.2489-2499.
3. Alexandra Hochuli, Stephan Holzer, Roger Wattenhofer (2014). *Distributed approximation Of minimum routing cost trees*. volume 8576 of Lecture Notes in Computer Science, page 121-136. Springer, Berlin & Heidelberg, Germany.
4. Bang Ye Wu, Kun-Mao Chao (2004). *Spanning trees and optimization problems*. Chapman&Hall/CRC, pp.13–139.
5. Bryant A.Julstrom (2005). *The Blob code is competitive with edgesets in genetic algorithms for the minimum routing cost spanning tree problem*. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), ACM, pp. 585–590.
6. Matteo Fischetti, Giuseppe Lancia, Paolo Serafini (2002). *Exact algorithms for minimum routing cost trees*. Networks, Wiley, volume 39 (3), pp.161–173.
7. Neil C. Jones and Pavel A. Pevzner (2004). An Introduction to Bioinformatics Algorithms. MIT, pp.1-417.
8. Phan Tan Quoc (2012). *A Heuristic approach for solving the minimum routing cost spanning tree problem*. International Journal of Machine Learning and Computing (IJMLC), IACSIT, volume 2, pp.406-409.
9. Phan Tấn Quốc, Nguyễn Đức Nghĩa (2013). *Thuật toán bầy ong giải bài toán cây khung với chi phí định tuyến nhỏ nhất*. Tạp chí Tin học và Điều khiển học, T.29, S3, 2013, pp.265-276.
10. Phan Tấn Quốc, Nguyễn Đức Nghĩa (2013). *Thuật toán tìm kiếm Tabu giải bài toán cây khung với chi phí định tuyến nhỏ nhất*. Tạp chí Công nghệ Thông tin và Truyền thông, pp.5-13.
11. Rui Campos, Manuel Ricardo (2008). *A fast algorithm for computing minimum routing cost spanning trees*. ELSEVIER, Computer Networks, volume 52, pp.3229-3247.
12. Vic Grout (2005). *Principles of cost minimization in wireless networks*. Journal of Heuristics, volume 11 (2), pp.115-133.