Thuật toán tìm kiếm Hill climbing giải bài toán Cây Steiner nhỏ nhất

Trần Việt Chương[1], Phan Tấn Quốc[2], Hà Hải Nam[3]
[1] Trung tâm Công nghệ thông tin và Truyền thông, Sở Thông tin và Truyền thông tỉnh Cà Mau
[2] Khoa Công nghệ thông tin, Trường Đại học Sài Gòn
[3] Viện Khoa học Kỹ thuật Bưu điện, Học viện Công nghệ Bưu chính Viễn thông

Tác giả liên hệ: Trần Việt Chương, chuongcm74@gmail.com
Ngày nhận bài: 26/10/2019, ngày sửa chữa: 06/03/2020
Định danh DOI: 10.32913/mic-ict-research-vn.vyyyy.nx.xysz

Tóm tắt: Cây Steiner nhỏ nhất (Steiner Minimum 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. Trước đây đã có nhiều công trình nghiên cứu theo các hướng tiếp cận khác nhau đưa ra các thuật toán để giải bài toán SMT. Bài báo này đề xuất thuật toán Hill climbing search để giải bài toán Cây Steiner nhỏ nhất, trong đó đề xuất cách thức tìm kiếm lân cận tất định và cách thức kết hợp tìm kiếm lân cận tất định với tìm kiếm lân cận ngẫu nhiên để giải quyết bài toán Cây Steiner nhỏ nhất. Kết quả thực nghiệm với hệ thống dữ liệu thực nghiệm chuẩn cho thấy thuật toán đề xuất của chúng tôi cho lời giải với chất lượng tốt hơn so với một số thuật toán heuristic hiện biết trên một số bộ dữ liệu.

Từ khóa: *thuật toán tìm kiếm leo đồi, Cây Steiner nhỏ nhất, thuật toán heuristic, thuật toán metaheuristic, lân cận tất định, lân cận ngẫu nhiên, tối ưu cục bộ.*

Title: **Hill Climbing Search Algorithm For Solving Steiner Minimum Tree Problem**

Abstract: Steiner Minimum Tree (SMT) is a complex optimization problem that has many important applications in science and technology. This is a NP-hard problem. In the past, there were many research projects based on different approaches offering algorithms to solve SMT problems. This paper proposes a Hill climbing search algorithm to solve the SMT problem; which suggests how to search nearby and how to combine with random nearby searches to solve local optimization problems. Experimental results with the standard experimental data system show that our proposed algorithm offers better quality solution than some existing heuristic algorithms on some data sets.

Keywords: *Hill climbing search algorithm, Steiner minimum tree problem, heuristic algorithm, metaheuristic algorithm, neighboring deterministic, random nearby, local optimization.*

I. GIỚI THIỆU

Giả sử chúng ta cần xây dựng một hệ thống giao thông nối một số địa điểm trong khu đô thị. Vấn đề đặt ra là phải thiết kế sao cho hệ thống giao thông đó có độ dài ngắn nhất nhằm giảm kinh phí xây dựng. Yêu cầu của bài toán cho phép thêm vào một số địa điểm trung gian để giảm thiểu tổng chiều dài hệ thống cần xây dựng. Đây là bài toán có thể vận dụng kiến thức bài toán Cây Steiner nhỏ nhất để giải.

Bài toán Cây Steiner nhỏ nhất hiện được nghiên cứu ở một số dạng sau: Bài toán thứ nhất là bài toán Cây Steiner nhỏ nhất với khoảng cách Euclide [11, 27, 28]. Bài toán thứ hai là bài toán Cây Steiner nhỏ nhất với khoảng cách chữ nhật [11]. Bài toán thứ ba là Cây Steiner nhỏ nhất với khoảng cách được cho ngẫu nhiên [4]. Đây là giới hạn nghiên cứu về bài toán Cây Steiner nhỏ nhất của chúng tôi trong bài báo này [28].

Mục này trình bày một số định nghĩa và khả năng ứng dụng của bài toán Cây Steiner nhỏ nhất.

A. Một số định nghĩa

Định nghĩa 1: Cây Steiner [4]

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

Tập L được gọi là tập *terminal*, các đỉnh thuộc tập L được gọi là đỉnh *terminal*.

Đỉnh thuộc cây T mà không thuộc tập L được gọi là đỉnh *Steiner* của cây T đối với tập L.

Định nghĩa 2: Chi phí Cây Steiner [4]

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 ta có $C(T) = \sum_{e \in E(T)} w(e)$.

Định nghĩa 3: Cây Steiner nhỏ nhất [4]

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 Minimum Tree problem - SMT) hoặc được gọi ngắn gọn là bài toán Cây Steiner (Steiner Tree problem).

SMT là bài toán tối ưu tổ hợp của lý thuyết đồ thị. Trong trường hợp tổng quát, SMT đã được chứng minh thuộc lớp *NP-hard* [4, 13].

Khác với bài toán cây khung nhỏ nhất (Minimum Spanning Trees Problem) - đó là bài toán đơn giản. Cây Steiner là cây đi qua tất cả các đỉnh thuộc tập *terminal* L và có thể thêm một số đỉnh khác nữa thuộc tập $V(G)$ chứ không nhất thiết phải đi qua tất cả các đỉnh của đồ thị.

Để ngắn gọn, trong bài báo này từ *đồ thị* sẽ được hiểu là đơn đồ thị, vô hướng, liên thông và có trọng số không âm trên cạnh.

Định nghĩa 4: *1-lân cận* của Cây Steiner T

Cho đồ thị G và T là một Cây Steiner của G. Ta gọi *1-lân cận* của Cây Steiner T là tập tất cả các Cây Steiner của đồ thị G sai khác với T đúng một cạnh. Nếu T' là một Cây Steiner thuộc *1-lân cận* của T thì ta nói T và T' là *1-lân cận* với nhau.

Trong một số trường hợp chúng ta còn sử dụng những lân cận rộng hơn so với *1-lân cận*. Khái niệm *k-lân cận* dưới đây là mở rộng trực tiếp của khái niệm *1-lân cận*.

Định nghĩa 5: *k-lân cận* của Cây Steiner T

Cho đồ thị G và T là một Cây Steiner của nó. Ta gọi *k-lân cận* của Cây Steiner T là tập tất cả các Cây Steiner của đồ thị G sai khác với T không quá k cạnh. Nếu T' là một Cây Steiner thuộc *k-lân cận* của T thì ta nói T và T' là *k-lân cận* với nhau.

Định nghĩa 6: Lân cận tất định và lân cận ngẫu nhiên

Nếu các Cây Steiner trong lân cận được xác định không phụ thuộc vào yếu tố ngẫu nhiên thì ta nói về lân cận tất định, còn nếu ngược lại, ta nói về lân cận ngẫu nhiên.

B. Ứng dụng của bài toán Cây Steiner nhỏ nhất

Bài toán SMT có thể được tìm thấy trong các ứng dụng quan trọng như: Bài toán thiết kế mạng truyền thông, bài toán thiết kế VLSI (Very Large Scale Integrated), các bài toán liên quan đến hệ thống mạng với chi phí nhỏ nhất [3, 4, 12, 17, 31].

Đóng góp chính của chúng tôi trong bài báo này là đã đề xuất cách thức tìm kiếm lân cận mới để giải bài toán Cây Steiner nhỏ nhất đồng thời cài đặt thực nghiệm thuật toán trên hệ thống dữ liệu thực nghiệm chuẩn.

II. KHẢO SÁT MỘT SỐ THUẬT TOÁN GIẢI BÀI TOÁN CÂY STEINER NHỎ NHẤT

Hiện tại, có nhiều hướng tiếp cận giải bài toán Cây Steiner nhỏ nhất như các thuật toán rút gọn đồ thị, các thuật toán tìm lời giải đúng, các thuật toán tìm lời giải gần đúng cận tỉ lệ, các thuật toán heuristic và các thuật toán metaheuristic.

A. Các thuật toán rút gọn đồ thị

Một số nghiên cứu 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 [10], công trình của Thorsten Koch Alexander Martin [26], C. C. Ribeiro, M.C. Souza [5],...

Ý tưởng chung của các thuật toán rút gọn đồ thị là nhằm gia tăng các đỉnh cho tập terminal và loại bỏ các đỉnh của đồ thị mà nó chắc chắn không thuộc về Cây Steiner nhỏ nhất cần tìm. Chất lượng các thuật toán giải bài toán SMT phụ thuộc vào độ lớn của hệ số $n - |L|$. Do vậy, mục đích của các thuật toán rút gọn đồ thị là làm giảm thiểu tối đa hệ số $n - |L|$.

Các thuật toán rút gọn đồ thị cũng được xem là bước tiền xử lý dữ liệu quan trọng trong việc giải bài

toán SMT. Hướng tiếp cận này là rất cần thiết khi tiếp cận bài toán SMT bằng các thuật toán tìm lời giải đúng [16].

B. Các thuật toán tìm lời giải đúng

Các nghiên cứu tìm lời giải đúng cho bài toán SMT như thuật toán quy hoạch động của Dreyfus và Wagner [33], thuật toán dựa trên phép nới lỏng Lagrange của Beasley [9], các thuật toán nhánh cận của Koch và Martin [26], Xinhui Wang [15, 30],...

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; nên khả năng ứng dụng của chúng không cao. Việc giải đúng bài toán SMT thực sự là một thách thức trong lý thuyết tối ưu tổ hợp.

Hướng tiếp cận này là cơ sở quan trọng có thể được sử dụng để đánh giá chất lượng lời giải của các thuật toán giải gần đúng khác khi giải bài toán SMT.

C. Các thuật toán gần đúng cận tỉ lệ

Thuật toán gần đúng cận tỉ lệ α nghĩa là lời giải tìm được gần đúng một cận tỉ lệ α so với lời giải tối ưu.

Ưu điểm của thuật toán gần đúng cận tỉ lệ là có sự đảm bảo về mặt toán học theo nghĩa cận tỉ lệ trên, nhược điểm của thuật toán dạng này là cận tỉ lệ tìm được trong thực tế thường là kém hơn nhiều so với chất lượng lời giải tìm được bởi nhiều thuật toán gần đúng khác dựa trên thực nghiệm.

Thuật toán MST-Steiner của Bang Ye Wu và Kun-Mao Chao có cận tỉ lệ 2 [4], thuật toán Zelikovsky-Steiner có cận tỉ lệ 11/6 [4]. Cận tỉ lệ tốt nhất tìm được hiện tại cho bài toán SMT là 1.39 [6, 18, 24].

D. Các thuật toán heuristic

Thuật toán heuristic chỉ ra 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 chính xác. Các thuật toán heuristic cũng không chắc hiệu quả trên mọi loại dữ liệu đối với một bài toán cụ thể.

Các thuật toán heuristic điển hình cho bài toán SMT như: SPT [21], PD Steiner [21] (bài báo này sẽ gọi tắt là PD), SPH [5], Heu [26], Distance network heuristic của Kou, Markowsky và Berman [34].

Ưu điểm của các thuật toán heuristic là thời gian chạy nhanh hơn nhiều so với các thuật toán meta-heuristic. Giải pháp này thường được lựa chọn với các đồ thị có kích thước lớn [11, 20, 21, 25].

E. 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 local search sử dụng các chiến lược tìm kiếm lân cận node-based neighborhood [19], path-based neighborhood [19], thuật toán local search [7], thuật toán tìm kiếm lân cận biến đổi [23], thuật toán di truyền [2], thuật toán tabu search [5], thuật toán di truyền song song [14], thuật toán bees [22],...

Từ những hướng tiếp cận trên, bài báo này đề xuất một thuật toán metaheuristic dạng cá thể. Cụ thể là thuật toán Hill climbing search để giải bài toán SMT. Cách tiếp cận này có thể giải được các bài toán SMT có kích thước lớn, có chất lượng tốt hơn so với các hướng tiếp cận heuristic, các thuật toán gần đúng cận tỉ lệ và có thời gian chạy nhanh hơn so với các thuật toán metaheuristic dạng quần thể [32].

III. THUẬT TOÁN HILL CLIMBING SEARCH

A. Ý tưởng thuật toán hill climbing search

Thuật toán Hill climbing search là một trong những kỹ thuật dùng để tìm kiếm tối ưu cục bộ cho một bài toán tối ưu [1].

Thuật toán Hill climbing search là một trong những giải pháp để giải bài toán tối ưu, đặc biệt là dạng bài toán ưu tiên về thời gian tính như bài toán dạng thiết kế.

B. Sơ đồ tổng quát thuật toán Hill climbing search

Thuật toán Hill climbing search liên tục thực hiện việc di chuyển từ một lời giải S đến một lời giải mới S' trong một cấu trúc lân cận xác định trước theo sơ đồ sau:

Bước 1: Khởi tạo. Chọn lời giải xuất phát S, tính giá trị hàm mục tiêu $F(S)$.

Bước 2: Sinh lân cận. Chọn tập lân cận $N(S)$ và tìm lời giải S' trong tập lân cận này với giá trị hàm mục tiêu $F(S')$.

Bước 3: Test chấp nhận. Kiểm tra xem có chấp nhận di chuyển từ S sang S'. Nếu chấp nhận thì thay S bằng S'; trái lại giữ nguyên S là lời giải hiện tại.

Bước 4: Test điều kiện dừng. Nếu điều kiện dừng thỏa mãn thì kết thúc thuật toán và đưa ra lời giải tốt nhất tìm được; trái lại quay lại bước 2.

C. Giải quyết vấn đề tối ưu hóa cục bộ

Vấn đề lớn nhất mà thuật toán Hill climbing search gặp phải là nó dễ rơi vào bẫy tối ưu cục bộ, đó là lúc chúng ta leo lên một đỉnh mà chúng ta không thể tìm được một lân cận nào tốt hơn được nữa nhưng đỉnh này lại không phải là đỉnh cao nhất.

Để giải quyết vấn đề này, khi leo đến một đỉnh tối ưu cục bộ, để tìm được lời giải tốt hơn nữa ta có thể lặp lại thuật toán Hill climbing search với nhiều điểm xuất phát khác nhau được chọn ngẫu nhiên và lưu lại kết quả tốt nhất ở mỗi lần lặp. Nếu số lần lặp đủ lớn thì ta có thể tìm đến được đỉnh tối ưu toàn cục, tuy nhiên với những bài toán mà không gian tìm kiếm lớn; thì việc tìm được lời giải tối ưu toàn cục là một thách thức lớn [1, 29].

Để giải quyết vấn đề tối ưu cục bộ, trong bài báo này chúng tôi đề xuất việc kết hợp thuật toán Hill climbing search với chiến lược tìm kiếm lân cận ngẫu nhiên để hy vọng nâng cao chất lượng lời giải của thuật toán.

IV. THUẬT TOÁN HILL CLIMBING SEARCH GIẢI BÀI TOÁN CÂY STEINER NHỎ NHẤT

A. Ý tưởng thuật toán

Cho đồ thị vô hướng liên thông có trọng số G. Bắt đầu từ Cây Steiner T của G được khởi tạo ngẫu nhiên, chèn lần lượt từng cạnh $e \in E(G) - E(T)$ vào Cây Steiner T. Nếu Cây Steiner T không chứa chu trình thì cạnh e không cần được xem xét; nếu $E(T) \cup e$ chứa một chu trình thì tìm một cạnh e' trên chu trình này sao cho việc loại nó dẫn đến Cây Steiner T' có chi phí nhỏ nhất. Tiếp theo, nếu $C(T') < C(T)$ thì thay T bằng T'. Thuật toán dừng nếu trong một lần duyệt qua tất cả các cạnh $e \in E(G) - E(T)$ mà không cải thiện được chi phí của Cây Steiner T.

Chúng tôi đặt tên thuật toán Hill climbing search giải bài toán SMT là HCSMT.

B. Sơ đồ thuật toán HCSMT

C. Một số bàn luận thêm

a) Vấn đề tạo lời giải ngẫu nhiên ban đầu

Cũng lưu ý rằng Cây Steiner ban đầu được khởi tạo ngẫu nhiên thông qua hai giai đoạn như sau (dòng 3 của thuật toán HCSMT):

Bắt đầu từ cây chỉ gồm một đỉnh nào đó của đồ thị, tiếp theo, thuật toán sẽ thực hiện $n-1$ bước lặp với n là số đỉnh của đồ thị đang xét. 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; thuật toán này sử dụng ý tưởng của thuật toán Prim.

Duyệt các đỉnh treo $u \in T$, nếu $u \notin L$ thì xóa cạnh chứa đỉnh u khỏi $E(T)$, xóa đỉnh u trong $V(T)$ và cập nhật bậc của đỉnh kề với đỉnh u trong T. Lặp lại bước này đến khi T không còn sự thay đổi nào nữa; bước này gọi là bước xóa các cạnh dư thừa.

b) Vấn đề tìm kiếm lân cận

Thao tác tìm chu trình trong Cây Steiner T sau khi chèn thêm một cạnh e được tiến hành như sau: Khi chèn cạnh $e = (u, v)$ vào T, duyệt Cây Steiner T theo chiều sâu bắt đầu từ u, lưu vết trên đường đi bằng mảng p (đỉnh trước của một đỉnh trong phép duyệt). Tiếp theo, bắt đầu từ đỉnh v, truy vết theo mảng p đến khi gặp u thì kết thúc, các cạnh trên đường truy vết chính là các cạnh trong chu trình cần tìm.

Thuật toán HCSMT ngoài việc lời giải ban đầu được khởi tạo ngẫu nhiên thì các Cây Steiner lân cận tìm được trong quá trình tìm kiếm là kiểu *1-lân cận* tất định. Hiệu quả của thuật toán HCSMT có thể được cải thiện khi ta thay đổi thứ tự các cạnh được duyệt trong tập $E(G) - E(T)$; 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.

c) Vấn đề kết hợp với tìm kiếm ngẫu nhiên

Thuật toán HCSMT chủ yếu sử dụng tính tăng cường, thể hiện qua các chiến lược tìm kiếm Cây

Thuật toán 1: Thuật toán HCSMT

```
Đầu vào: Đồ thị G = (V(G), E(G))
Đầu ra : Cây Steiner có chi phí nhỏ nhất tìm được
stop = false;
d = 0; //d là số lần tăng cường
T là Cây Steiner được khởi tạo ngẫu nhiên;
T_best = T; //lưu lại Cây Steiner tốt nhất trước khi thực hiện chiến lược đa dạng hóa;
while (d < N){ // N là số lần lặp định trước
    stop = true;
    S = E − E(T);
    for (với mỗi cạnh e ∈ S){
        min = +∞;
        if (cạnh e có 2 đỉnh (u, v) ∈ T){
            F = T ∪ e;
            Xác định chu trình Cycle trong F chứa e;
            for (với mỗi cạnh e′ ∈ Cycle)
                if (C(F − e′) < min){
                    min = C(F − e′);
                    Ghi nhận T′ = F − e′;
                }
            if (min < C(T)){
                T = T′;
                stop = false;
            }
        }
    }
    if (stop){
        d++; //tăng số biến đếm tăng cường
        if (C(T) < C(T_best)) T_best = T;
        while (Thỏa điều kiện đa dạng hóa){
            Loại bỏ một số cạnh ngẫu nhiên của Cây Steiner đang xét;
            Chọn ngẫu nhiên một số cạnh trong các cạnh còn lại của đồ thị G cho đến khi cây T liên thông trở lại (điều kiện cạnh thêm vào là: Phải có đỉnh thuộc T mới được thêm vào để tránh trường hợp thêm vào quá nhiều lần nhưng T không liên thông trở lại). Sử dụng định nghĩa k-lân cận ở trên;
            Nếu thêm quá nhiều lần mà không làm cho T liên thông trở lại, thì tiếp tục đa dạng hóa lại với các cạnh khác;
        }
    }
}
return T_best;
```

Steiner lân cận. Tính đa dạng được sử dụng vào các thời điểm sau: thứ nhất là khi khởi tạo lời giải ban đầu, thứ hai là thay đổi thứ tự duyệt của các cạnh trong tập cạnh ứng viên (như đã đề cập ở đoạn trên), thứ ba khi việc tìm kiếm lân cận không cải thiện qua một số lần lặp thì tiến hành chọn ngẫu nhiên một số cạnh của cây để bắt đầu trở lại việc tìm kiếm lân cận.

V. THỰC NGHIỆM VÀ ĐÁNH GIÁ

Phần này chúng tôi mô tả chi tiết việc thực nghiệm thuật toán HCSMT do chúng tôi đề xuất; đồng thời đưa ra một số so sánh, đánh giá về các kết quả đạt được.

A. Dữ liệu thực nghiệm

Để thực nghiệm các thuật toán liên quan, chúng tôi sử dụng 60 bộ dữ liệu là các đồ thị thưa trong hệ thống dữ liệu thực nghiệm chuẩn dùng cho bài toán Cây Steiner tại địa chỉ URL http://people.brunel.ac.uk/~mastjjb/jeb/orlib/steininfo.html[8]. Trong đó nhóm steinc có 20 đồ thị, nhóm steind có 20 đồ thị, nhóm steine có 20 đồ thị (các đồ thị này có tập $|L|$ tối đa 1250 đỉnh).

B. Môi trường thực nghiệm

Thuật toán HCSMT được chúng tôi cài đặt bằng ngôn ngữ C++ sử dụng môi trường DEV C++ 5.9.2; được thực nghiệm trên một máy chủ ảo, Hệ điều hành Windows server 2008 R2 Enterprise 64bit, Intel(R) Xeon (R) CPU E5-2660 @ 2.20 GHz, RAM 4GB.

C. Kết quả thực nghiệm và đánh giá

Kết quả thực nghiệm của các thuật toán được ghi nhận ở các Bảng I,II,III. Các bảng này có cấu trúc như sau: Cột đầu tiên (Test) là tên các bộ dữ liệu trong hệ thống dữ liệu thực nghiệm, số đỉnh (n), số cạnh (m) của từng đồ thị, các cột tiếp theo ghi nhận giá trị chi phí Cây Steiner lần lượt ứng với hai thuật toán hueristic: Heu [26], PD [21]. Hai thuật toán metahueristic: VNS [23], TS [5] và thuật toán HCSMT.

Bộ tham số được xác định như sau qua thực nghiệm: Số lần chạy mỗi bộ dữ liệu là 30, số lần tăng cường là 50; số cạnh loại bỏ ngẫu nhiên của mỗi lần tăng cường là $0.05\times|E(T)|$.

Bảng I
KẾT QUẢ THỰC NGHIỆM THUẬT TOÁN TRÊN NHÓM ĐỒ THỊ STEINC

Test	*n*	*m*	Heu	PD	VNS	TS	HC SMT
steinc1.txt	500	625	85	85	85	85	85
steinc2.txt	500	625	144	144	144	144	144
steinc3.txt	500	625	755	762	754	754	754
steinc4.txt	500	625	1080	1085	1079	1079	1080
steinc5.txt	500	625	1579	1583	1579	1579	1579
steinc6.txt	500	1000	55	55	55	55	55
steinc7.txt	500	1000	102	102	102	102	102
steinc8.txt	500	1000	510	516	509	509	509
steinc9.txt	500	1000	715	718	707	707	707
steinc10.txt	500	1000	1093	1107	1093	1093	1093
steinc11.txt	500	2500	32	34	33	32	32
steinc12.txt	500	2500	46	48	46	46	46
steinc13.txt	500	2500	262	268	258	258	258
steinc14.txt	500	2500	324	332	323	324	324
steinc15.txt	500	2500	557	562	556	556	557
steinc16.txt	500	12500	11	12	11	11	11
steinc17.txt	500	12500	19	20	18	18	18
steinc18.txt	500	12500	120	123	115	117	115
steinc19.txt	500	12500	150	159	148	148	149
steinc20.txt	500	12500	268	268	268	267	268

Bảng II
KẾT QUẢ THỰC NGHIỆM THUẬT TOÁN TRÊN NHÓM ĐỒ THỊ STEIND

Test	*n*	*m*	Heu	PD	VNS	TS	HC SMT
steind1.txt	1000	1250	106	107	106	106	106
steind2.txt	1000	1250	220	228	220	220	220
steind3.txt	1000	1250	1570	1771	1565	1567	1565
steind4.txt	1000	1250	1936	2174	1935	1935	1936
steind5.txt	1000	1250	3252	3511	3250	3250	3250
steind6.txt	1000	2000	70	70	67	70	67
steind7.txt	1000	2000	103	111	103	103	103
steind8.txt	1000	2000	1092	1287	1073	1078	1073
steind9.txt	1000	2000	1462	1773	1448	1450	1448
steind10.txt	1000	2000	2113	2550	2111	2112	2113
steind11.txt	1000	5000	29	29	29	30	29
steind12.txt	1000	5000	42	44	42	42	42
steind13.txt	1000	5000	510	643	502	502	507
steind14.txt	1000	5000	675	851	671	667	674
steind15.txt	1000	5000	1120	1437	1116	1117	1118
steind16.txt	1000	25000	13	13	13	13	13
steind17.txt	1000	25000	23	25	23	23	23
steind18.txt	1000	25000	238	301	228	230	231
steind19.txt	1000	25000	325	424	318	315	321
steind20.txt	1000	25000	539	691	538	538	539

D. Đánh giá kết quả thực nghiệm

Mục này nhằm so sánh chất lượng lời giải của thuật toán HCSMT với hai thuật toán heuristic: Heu [26], PD [21] và hai thuật toán metaheuristic: VNS [23], TS [5]. Nội dung của Bảng IV,V,VI,VII cho biết số lượng (SL) và tỉ lệ phần trăm (%) tương ứng với số lượng bộ dữ liệu cho chất lượng lời giải tốt hơn (ghi nhận bởi ký hiệu "<") hoặc cho chất lượng lời giải bằng nhau (ghi nhận bởi ký hiệu "=") hoặc cho chất lượng lời giải kém hơn (ghi nhận bởi ký hiệu ">") khi so sánh thuật toán HCSMT với các thuật toán: Heu [26], PD [21], VNS [23] và TS [5].

Với 20 bộ dữ liệu nhóm steinc, thuật toán HCSMT cho chất lượng lời giải (tốt hơn, bằng, kém hơn) thuật toán Heu [26] (35,0%, 65,0%, 0,0%). Kết quả so sánh thuật toán HCSMT với thuật toán Heu [26] trên các nhóm dữ liệu steind, steine cũng đã được thể hiện chi tiết ở Bảng IV.

Tương tự, kết quả so sánh thuật toán HCSMT với thuật toán PD [21] trên các nhóm dữ liệu steinc, steind, steine cũng đã được thể hiện chi tiết ở Bảng V.

Kết quả so sánh thuật toán HCSMT với thuật toán VNS [23] trên các nhóm dữ liệu steinc, steind, steine cũng đã được thể hiện chi tiết ở Bảng VI.

Đánh giá chung trên toàn bộ 60 bộ dữ liệu, thuật toán HCSMT cho chất lượng lời giải (tốt hơn, bằng, kém hơn) thuật toán Heu [26] (46,7%, 51,7%, 1,6%). Thuật toán HCSMT cho chất lượng lời giải (tốt hơn, bằng, kém hơn) thuật toán PD [21] (78,4%, 21,6%, 0,0%). Thuật toán HCSMT cho chất lượng lời giải (tốt hơn, bằng, kém hơn) thuật toán VNS [23] (1,7%, 70%, 28,3%). Thuật toán HCSMT cho chất lượng lời giải (tốt hơn, bằng, kém hơn) thuật toán TS [5] (26,7%,

Bảng III
KẾT QUẢ THỰC NGHIỆM THUẬT TOÁN TRÊN NHÓM ĐỒ THỊ STEINE

Test	*n*	*m*	Heu	PD	VNS	TS	HC SMT
steine1.txt	2500	3125	111	111	111	111	111
steine2.txt	2500	3125	214	214	214	216	214
steine3.txt	2500	3125	4052	4570	4015	4018	4015
steine4.txt	2500	3125	5114	5675	5101	5105	5101
steine5.txt	2500	3125	8130	8976	8128	8128	8130
steine6.txt	2500	5000	73	73	73	73	73
steine7.txt	2500	5000	149	150	145	149	145
steine8.txt	2500	5000	2686	3254	2648	2649	2648
steine9.txt	2500	5000	3656	4474	3608	3605	3608
steine10.txt	2500	5000	5614	6847	5600	5602	5600
steine11.txt	2500	12500	34	34	34	34	34
steine12.txt	2500	12500	68	68	67	68	68
steine13.txt	2500	12500	1312	1704	1292	1299	1312
steine14.txt	2500	12500	1752	2304	1735	1740	1735
steine15.txt	2500	12500	2792	3626	2784	2784	2799
steine16.txt	2500	62500	15	15	15	15	15
steine17.txt	2500	62500	26	27	25	25	25
steine18.txt	2500	62500	608	804	583	595	594
steine19.txt	2500	62500	788	1059	768	778	768
steine20.txt	2500	62500	1349	1753	1342	1352	1342

46,6%, 26,7%).

Thời gian tính trung bình của thuật toán HCSMT trên các đồ thị ứng với nhóm dữ liệu steinc là 2,54 giây. Tương tự với nhóm đồ thị steind là 16,78 giây, với nhóm đồ thị steine là 214,81 giây. Thời gian chạy chương trình ngoài việc phụ thuộc vào độ phức tạp thời gian tính của thuật toán, còn phụ thuộc vào môi trường thực nghiệm, kỹ thuật lập trình, cấu trúc dữ liệu chọn cài đặt, các bộ tham số,... và thông thường, thời gian chạy của thuật toán heuristic nhanh hơn thuật toán metaheuristic. Tuy vậy, thời gian chạy của thuật toán HCSMT như trên cũng cho chúng ta thông tin tham khảo cần thiết về thuật toán này.

Kết quả so sánh thuật toán HCSMT với thuật toán TS [5] trên các nhóm dữ liệu steinc, steind, steine cũng đã được thể hiện chi tiết ở Bảng VII.

Bảng IV
SO SÁNH CHẤT LƯỢNG LỜI GIẢI CỦA THUẬT TOÁN HCSMT VỚI THUẬT TOÁN HEU

Nhóm đồ thị	HCSMT<Heu		*HCSMT=Heu*		*HCSMT>Heu*	
	SL	%	SL	%	SL	%
steinc	7	35%	13	65%	0	0%
steind	10	50%	10	50%	0	0%
steine	11	55%	8	40%	1	5%
Tổng cộng:	**28**	**46,7%**	**31**	**51,7%**	**1**	**1,6%**

Bảng V
SO SÁNH CHẤT LƯỢNG LỜI GIẢI CỦA THUẬT TOÁN HCSMT VỚI THUẬT TOÁN PD

Nhóm đồ thị	HCSMT<PD		*HCSMT=PD*		*HCSMT>PD*	
	SL	%	SL	%	SL	%
steinc	15	75%	5	25%	0	0%
steind	18	90%	2	10%	0	0%
steine	14	70%	6	30%	0	0%
Tổng cộng:	**47**	**78.4%**	**13**	**21.6%**	**0**	**0%**

VI. KẾT LUẬN

Bài báo này đề xuất thuật toán HCSMT dạng thuật toán Hill climbing search để giải bài toán Cây Steiner nhỏ nhất. Đóng góp của chúng tôi trong bài báo này là đã đề xuất cách thức tìm kiếm lân cận kết hợp với tìm kiếm lân cận ngẫu nhiên nâng cao chất lượng của thuật toán. Chúng tôi đã thực nghiệm thuật toán đề xuất trên hệ thống gồm 60 bộ dữ liệu thực nghiệm chuẩn. Kết quả thực nghiệm cho thấy rằng thuật toán này cho chất lượng lời giải tốt hơn hoặc bằng hai thuật toán dạng heuristic và cho chất lượng lời giải tốt hơn, bằng, kém hơn hai thuật toán dạng metaheuristic tốt hiện biết trên một số bộ dữ liệu thực nghiệm chuẩn.

TÀI LIỆU THAM KHẢO

[1] Alan W. Johnson, *"Generalized Hill Climbing Algorithms For Discrete Optimization Problems"*, doctor of philosophy, Industrial And Systems Engineering, Blacksburg, Virginia, pp.1-119, 1996.

[2] 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.

Bảng VI
SO SÁNH CHẤT LƯỢNG LỜI GIẢI CỦA THUẬT TOÁN HCSMT VỚI THUẬT TOÁN VNS

Nhóm đồ thị	HCSMT<VNS		*HCSMT=VNS*		*HCSMT>VNS*	
	SL	%	SL	%	SL	%
steinc	1	5%	15	75%	4	20%
steind	0	0%	12	60%	8	40%
steine	0	0%	15	75%	5	25%
Tổng cộng:	**1**	**1.7%**	**42**	**70%**	**17**	**28.3%**

Bảng VII
SO SÁNH CHẤT LƯỢNG LỜI GIẢI CỦA THUẬT TOÁN HCSMT VỚI THUẬT TOÁN TS

Nhóm đồ thị	HCSMT<TS		*HCSMT=TS*		*HCSMT>TS*	
	SL	%	SL	%	SL	%
steinc	1	5%	15	75%	4	20%
steind	5	25%	7	35%	8	40%
steine	10	50%	6	30%	4	20%
Tổng cộng:	**16**	**26,7%**	**28**	**46,6%**	**16**	**26,7%**

[3] Arie M.C.A. Koster, Xavier Munoz (Eds), *"Graphs and algorithms in communication networks"*, Springer, pp.1-177, 2010.

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

[5] C. C. RIBEIRO, M.C. SOUZA, *"Tabu Search for the Steiner problem in graphs"*, Networks, 36, pp.138-146, 2000.

[6] Chi-Yeh Chen, *"An efficient approximation algorithm for the Steiner tree problem"*, National Cheng Kung University, Taiwan, 2018.

[7] Eduardo Uchoa, Renato F. Werneck, *"Fast local search for Steiner trees in graphs"*, SIAM, 2010.

[8] J. E. Beasley, OR-Library: URL http://people.brunel.ac.uk/ mastjjb/jeb/orlib/stein-info.html

[9] J. E. Beasley, *"An SST-Based algorithm for the Steiner problem in graphs"*, Networks, 19, pp.1-16, 1989.

[10] 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.

[11] Jon William Van Laarhoven, *"Exact and heuristic algorithms for the euclidean Steiner tree problem"*, University of Iowa, 2010 (Doctoral thesis).

[12] M. Hauptmann, M. Karpinski (Eds), *"A compendium on Steiner tree problems"*, pp.1-36, 2015.

[13] Marcello Caleffi, Ian F. Akyildiz, Luigi Paura, *"On the solution of the Steiner tree np-hard problem via physarum bionetwork"*, IEEE, pp.1092-1106, 2015.

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

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

[16] Phan Tấn Quốc, *"Rút gọn đồ thị cho bài toán Cây Steiner nhỏ nhất"*, Kỷ yếu Hội nghị Quốc gia lần thứ IX về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR), Viện Công nghệ thông tin, Viện Hàn lâm Khoa học và Công nghệ Việt Nam, Trường Đại học Cần Thơ, ngày 04-05/08/2016, ISBN: 978-604-913-472-2, trang 638-644, 2016.

[17] Poompat Saengudomlert, *"Optimization for Communications and Networks"*, Taylor and Francis Group, LLC, pp.1-201, 2011.

[18] Reyan Ahmed and et al, *"Multi-level Steiner tree"*, ACM J. Exp. Algor., 1(1), 2018.

[19] S.L. Martins, M.G.C. Resende, C.C. Ribeiro, and P.M. Pardalos, *"A parallel grasp for the Steiner tree problem in graphs using a hybrid local search strategy"*, 1999.

[20] 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.

[21] Trần Việt Chương, Phan Tấn Quốc, Hà Hải Nam, *"Đề xuất một số thuật toán heuristic giải bài toán Cây Steiner nhỏ nhất"*, Kỷ yếu Hội nghị Quốc gia lần thứ X về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR), Trường Đại học Đà Nẵng, ngày 17-18/08/2017, ISBN: 978-604-913-614-6, trang 138-147, 2017.

[22] Trần Việt Chương, Phan Tấn Quốc, Hà Hải Nam, *"Thuật toán Bees giải bài toán Cây Steiner nhỏ nhất trong trường hợp đồ thị thưa"*. Tạp chí khoa học công nghệ thông tin và truyền thông, Học viên Công nghệ Bưu chính Viễn thông, Bộ Thông tin và Truyền thông, ISSN: 2525-2224, tháng 12, trang 24-29, 2017.

[23] Tran Viet Chuong, Ha Hai Nam *"A variable neighborhood search algorithm forsolving the Steiner minimal tree problem in sparse graphs"*, Context-Aware Systems and Applications, and Nature of Computation and Communication, EAI, Springer, pp.218-225, 2018.

[24] Thomas Pajor, Eduardo Uchoa, Renato F. Werneck, *"A robust and scalable algorithm for the Steiner problem in graphs"*, Springer, 2018.

[25] Thomas Bosman, *"A solution merging heuristic for the Steiner problem in graphs using tree decompositions"*, VU University Amsterdam, The Netherlands,

pp.1-12, 2015.

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

[27] Trần Lê Thủy, *"Về bài toán Steiner"*, Viện Toán học, Viện Hàn lâm Khoa học và Công nghệ Việt Nam, 2014 (luận văn thạc sĩ).

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

[29] Wassim Jaziri (Edited). *"Local Search Techniques: Focus on Tabu Search"*. In-teh, Vienna, Austria, pp.1-201, 2008.

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

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

[32] Xin-she Yang. *"Engineering optimization"*. Wiley, pp.21-137, 2010.

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

[34] L. Kou, G. Markowsky, L. Berman, *"A Fast Algorithm for Steiner Trees"*, acta informatica, Vol.15, pp.141-145, 1981.

SƠ LƯỢC VỀ CÁC TÁC GIẢ

Trần Việt Chương

Sinh ngày: 01/12/1974

Nhận bằng Thạc sỹ Khoa học máy tính năm 2005 tại Trường Đại học Sư phạm I Hà Nội.

Hiện công tác tại Trung tâm CNTT và Truyền thông tỉnh Cà Mau, Sở Thông tin và Truyền thông tỉnh Cà Mau. Lĩnh vực nghiên cứu: Thuật toán tiến hóa và tối ưu.

Điện thoại: 0913 688 477

E-mail: chuongtv@camau.gov.vn

Phan Quốc Tấn

Sinh ngày: 12/10/1971

Nhận bằng Thạc sỹ Tin học tại Trường Đại học KHTN - ĐHQG TP.HCM năm 2002, Tiến sỹ chuyên ngành Khoa học máy tính tại Trường Đại học Bách Khoa Hà Nội năm 2015.

Hiện công tác tại Bộ môn Khoa học Máy tính, khoa Công nghệ Thông tin, Trường Đại học Sài Gòn. Lĩnh vực nghiên cứu: Thuật toán gần đúng.

Điện thoại: 0918 178 052

E-mail: quocpt@sgu.edu.vn

Hà Hải Nam

Sinh ngày: 10/02/1975

Nhận bằng Thạc sỹ Tin học năm 2004 tại Trường Đại học Bách Khoa Hà Nội, Tiến sỹ năm 2008 tại Vương quốc Anh. Được phong học hàm PGS năm 2014

Hiện công tác tại Viện Khoa học Kỹ thuật Bưu điện, Học viện Công nghệ Bưu chính Viễn thông. Lĩnh vực nghiên cứu: Hệ thống phân tán và tối ưu hóa.

Điện thoại: 091 66 34567

E-mail: namhh@ptit.edu.vn