THUẬT TOÁN BEES GIẢI BÀI TOÁN CÂY STEINER NHỎ NHẤT TRONG TRƯỜNG HỢP ĐỒ THỊ THƯA

Trần Việt Chương[*], Phan Tấn Quốc[+], Hà Hải Nam[x]

[*]Trung tâm Công nghệ thông tin và Truyền thông, Sở Thông tin và Truyền thông tỉnh Cà Mau

[+]Khoa Công nghệ thông tin, Trường Đại học Sài Gòn

[x]Viện Khoa học Kỹ thuật Bưu điện, Học viện Công nghệ Bưu chính viễn thông

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*. Trong hàng chục năm qua, đã có nhiều bài báo khoa học công bố cách giải bài toán *SMT*. Trong bài báo này, chúng tôi đề xuất một thuật toán mới dựa trên sơ đồ thuật toán bees cơ bản để giải bài toán *SMT*. Chúng tôi đã cài đặt và thực nghiệm thuật toán đề xuất trên 38 bộ dữ liệu trong hệ thống dữ liệu thực nghiệm chuẩn; kết quả thực nghiệm cho thấy thuật toán đề xuất cho lời giải với chất lượng tốt hơn một số thuật toán heuristic và metaheuristic hiện biết trên một số bộ dữ liệu.

Từ khóa: Steiner minimal tree, bees algorihms, metaheuristic algorithms, sparse graphs. heuristic algorihms.

I. MỞ ĐẦU

A. Các định nghĩa

Mục này trình bày một số định nghĩa và tính chất căn bản liên quan đến bài toán cây Steiner nhỏ nhất.

Định nghĩa 1. *Cây Steiner* [1]

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 L là tập con các đỉnh của $V(G)$, cây T đi qua tất cả các đỉnh trong L được gọi là cây Steiner của L.

Tập L được gọi là tập *terminal*, các đỉnh thuộc tập L được gọi là các đỉnh *terminal*, các đỉnh thuộc cây T mà không thuộc tập L 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* L và có thể thêm một số đỉnh khác nữa thuộc tập $V(G)$.

Định nghĩa 2. *Chi phí cây Steiner* [1]

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* [1]

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* [1,9]. Có hai trường hợp đặc biệt đối với bài toán *SMT* có thể giải được bằng thời gian đa thức; đó là khi $L=V(G)$ và khi $|L|=2$ ($|L|$ là ký hiệu số lượng đỉnh của tập Y): Khi $L=V$ 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 $|L|=2$ 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.

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, bài toán định tuyến trong *VLSI*, các bài toán liên quan đến thiết kế hệ thống mạng với chi phí nhỏ nhất [1],…

C. Một số nghiên cứu liên quan bài toán SMT

Bài toán *SMT* đã thu hút được sự quan tâm nghiên cứu của nhiều nhà khoa học trên thế giới trong hàng chục năm qua [5,13,16]. Hiện tại đã có nhiều thuật toán giải bài toán *SMT* được đề xuất:

Hướng thứ nhất là các thuật toán tìm lời giải đúng. Chẳng hạn thuật toán quy hoạch động, thuật toán dựa trên phép nới lỏng Lagrange, thuật toán nhánh cận,…

Hướng thứ hai 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. Ưu điểm của các thuật toán heuristic là cho thời gian chạy nhanh.

Hướng thứ ba 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.

Trong các hướng tiếp cận trên, các thuật toán dạng heuristic và metaheuristic đang được quan tâm nhiều nhất [6,7,8,12].

D. Thuật toán Bees cơ bản

Các thuật toán mô phỏng theo quá trình tìm kiếm thức ăn của loài ong đã được biết đến trong các công trình của các nhóm tác giả Dusan Teodorovic (2010) [4], D.T. Pham (2005) [3], Xin-She Yang (2010) [15],…

Các thuật toán Bees sử dụng đồng thời ba chiến lược tìm kiếm sau: *tìm kiếm lân cận, tìm kiếm ngẫu nhiên* và *tìm kiếm sâu hơn ở những vùng tiềm năng*. Các thuật toán Bees được đánh giá là thích hợp để giải các bài toán tối ưu tổ hợp khó so với nhiều metaheuristic phổ biến trước đó [11,15].

Sơ đồ thuật toán Bees cơ bản

1. Khởi tạo quần thể ban đầu với N cá thể ong; các cá thể này được tạo ngẫu nhiên hoặc bằng một heuristic đặc thù tùy theo bài toán.
2. Đánh giá độ thích nghi cho mỗi cá thể của quần thể.
3. **while** (*điều kiện dừng chưa thoả*) {
4. Trong N cá thể ong, chọn ngẫu nhiên ra p cá thể để thực hiện tìm kiếm lân cận ($p < N$). Trong số p cá thể được chọn này chọn ra h cá thể có độ thích nghi cao nhất (phân bổ các thể vào ba nhóm khác nhau để thực hiện việc tìm kiếm).
5. Cử thêm *nep* ong để thực hiện việc tìm kiếm lân cận cho h cá thể tốt nhất trên và *nsp* ong để thực hiện việc tìm kiếm lân cận cho $p - h$ cá thể được chọn còn lại. Đánh giá độ thích nghi cho các cá thể vừa được bổ sung (*nsp < nep, nsp và nep* là các số nguyên dương, thực hiện việc tìm kiếm lân cận).
6. Sau khi thực hiện tìm kiếm lân cận, mỗi cá thể (gốc) có thể sinh thêm một hoặc một số lân cận. Trong số cá thể gốc và các cá thể lân cận của nó (có thể gọi là một vùng các cá thể) chọn ra một cá thể có độ thích nghi cao nhất đại diện cho vùng cá thể đó ở lần tiến hóa tiếp theo.
7. Mỗi cá thể trong số $N - p$ cá thể không được chọn sẽ được thay thế bằng một cá thể ngẫu nhiên (thực hiện việc tìm kiếm ngẫu nhiên). Đánh giá độ thích nghi cho các cá thể được bổ sung
8. }
9. **return** cá thể tốt nhất trong quần thể ở thế hệ cuối cùng;

Thuật toán bees cơ bản trên sử dụng các tham số sau: N là số lượng cá thể ong được khởi tạo; p là số lượng cá thể được chọn trong số N cá thể; h là số lượng cá thể có chất lượng tốt nhất trong số p cá thể được chọn; $p-h$ là số lượng cá thể được chọn còn lại; $N-p$ là số lượng cá thể không được chọn; *nep* là số lượng ong cử đến h cá thể tốt nhất; *nsp* là số lượng ong cử đến $p-h$ cá thể được chọn còn lại.

II. THUẬT TOÁN BEES GIẢI BÀI TOÁN STEINER MINIMAL TREES

Phần này trình bày chi tiết các bước của thuật toán bees giải bài toán *SMT*; chúng tôi gọi thuật toán này là *Bees-Steiner*.

A. Tạo quần thể ban đầu

Thuật toán *Bees-Steiner* tạo quần thể lời giải ban đầu theo ý tưởng của thuật toán Prim: Bắt đầu từ một đỉnh nào đó thuộc tập *terminal L*, tiếp theo trong 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 dừng khi tất cả các đỉnh thuộc tập *terminal L* đều đã được chọn. Chúng tôi gọi thuật toán này là *LikePrim*.

***LikePrim* (*V,E*)**
Đầu vào: Đồ thị $G=(V(G),E(G))$
Đầu ra: Cây Steiner ngẫu nhiên $T=(V(T),E(T))$
1. Chọn ngẫu nhiên đỉnh $u \in L$;
2. $V(T) = \{u\}$;
3. $E(T) = \emptyset$;
4. **while** ($L \not\subset V(T)$) {
5. Chọn ngẫu nhiên đỉnh $v \in V(G) - V(T)$ sao cho v có kề với một đỉnh $z \in V(T)$;
6. $V(T) = V(T) \cup \{v\}$;
7. $E(T) = E(T) \cup \{(v, z)\}$;
8. }
9. **return** cây Steiner T;

Ưu điểm của thuật toán *LikePrim* so với các heuristic khác trong việc tạo lời giải ban đầu là sự đa dạng các cạnh trong cây Steiner được hình thành. Chất lượng của quần thể ban đầu được tạo bởi thuật toán *LikePrim* dù không tốt bằng cách sử dụng các heuristic đặc thù của bài toán *SMT* chẳng hạn như áp dụng các thuật toán tìm đường đi ngắn nhất như Dijkstra hoặc *Floyd*; tuy nhiên sau quá trình tiến hóa, quần thể ban đầu được khởi tạo bằng thuật toán *LikePrim* sẽ cho chất lượng lời giải tốt hơn.

Tiếp theo, ta xây dựng thuật toán tạo quần thể ban đầu cho bài toán *SMT* như sau:

***InitPopulation* (*V,E*)**
Đầu vào: Đồ thị $G=(V(G), E(G))$
Đầu ra: Quần thể P gồm N cây Steiner $T_1,T_2,\ldots,T_N$ của đồ thị G
1. $\boldsymbol{P} = \emptyset$;
2. **for** i=1..N {
3. T = LikePrim (V,E) ;
4. $P = P \cup \{T\}$;
5. }
6. **return** P;

Xóa các cạnh dư thừa

Với cách tạo cây Steiner như trên; thì trong các cây Steiner có thể có một số cạnh dư thừa. Với mỗi cây Steiner T, 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 thay đổi.

Độ thích nghi của mỗi cá thể cây Steiner được tính theo định nghĩa 2 trên.

B. Điều kiện dừng của thuật toán Bees-Steiner

Thuật toán *Bees-Steiner* sử dụng điều kiện dừng sau: Lời giải tốt nhất tìm được (kỷ lục) của bài toán không được cải thiện sau một số lần lặp định trước (thường là một hàm theo kích cỡ của dữ liệu đầu vào).

C. Phân nhóm các cá thể

Quần thể P có N cá thể, sắp xếp các cá thể trong quần thể P theo chiều tăng dần theo chi phí của các cá thể. Sau khi sắp xếp, ta phân bố các cá thể vào ba nhóm: Nhóm 1 gồm h cá thể tốt nhất, nhóm 2 gồm p-h cá thể tốt tiếp theo và nhóm 3 gồm N-p cá thể còn lại.

Hàm sắp xếp quần thể và phân bố các cá thể vào các nhóm được đặt tên là ***SortPopulation***(P,N,h,p).

Thuật toán sau đây cho phép phân nhóm các cá thể.

***SortPopulation*(*P,N,h,p*)**

Đầu vào: Quần thể P gồm N cây $T_1, T_2,\ldots, T_N$ của đồ thị G. Các tham số h,p.
Đầu ra: Xếp các cá thể trong P vào ba nhóm.

1. Sắp xếp các cá thể của quần thể theo chi phí tăng dần;
2. Xếp h cá thể đầu tiên $T_1, T_2,..., T_h$ vào nhóm 1;
3. Xếp p-h cá thể tiếp theo $T_{h+1}, T_{h+2},..., T_p$ vào nhóm 2;
4. Xếp N-p cá thể còn lại $T_{p+1}, T_{p+2},..., T_N$ vào nhóm 3;

D. Tìm cây Steiner lân cận

Thủ tục tìm kiếm lân cận bắt đầu từ cây Steiner T được tiến hành như sau: Loại ngẫu nhiên khỏi T một cạnh e, chọn ngẫu nhiên cạnh e' từ tập $E - E(T)$, nếu tập cạnh $E(T) - \{e\} \cup \{e'\}$ cho ta cây Steiner T' có chi phí tốt hơn T thì ghi nhận cây Steiner này. Thủ tục trên sẽ được lặp lại k lần đối với cây Steiner T, trong số các cây Steiner được ghi nhận chọn ra T^* là cây Steiner tốt nhất, nếu cây Steiner T có chi phí tốt hơn T^* thì đặt $T = T^*$. Như vậy quần thể P được cập nhật sau khi thực hiện xong thủ tục tìm kiếm lân cận. Hàm tìm kiếm lân cận vừa mô tả được đặt tên là ***NeighSearch***(T,k).

Thuật toán sau đây cho phép tìm kiếm cây Steiner lân cận.

NeighSearch(T, k)

Đầu vào: Cây Steiner $T=(V(T),E(T))$ và số nguyên dương k
Đầu ra: Thay thế cây Steiner T bởi cây Steiner tốt nhất trong số k cây Steiner *lân cận* được tạo ngẫu nhiên.

1. $T^*=T$; // T^* là cây Steiner tốt nhất trong lân cận // của cây T
2. **for** i=1..k{
3. Loại ngẫu nhiên một cạnh $e \in E(T)$;
4. Chọn ngẫu nhiên cạnh $e' \in E - E(T)$;
5. **if** ($T' = (V, E(T) - \{e\} \cup \{e'\})$ là cây Steiner) ***và*** ($C(T') < C(T^*)$)
6. $T^* = T'$;
7. **}**
8. **if** $C(T^*) < C(T)$
9. $\boldsymbol{P} = \boldsymbol{P} - \{T\} \cup \{T^*\}$;// thay cây Steiner T bởi cây // Steiner T^* trong quần thể P;

E. Tìm cây Steiner ngẫu nhiên

Tìm kiếm ngẫu nhiên cho cây Steiner T được tiến hành tương tự như tìm kiếm lân cận nhưng không quan tâm đến chi phí trên cạnh: Loại ngẫu nhiên cạnh e trong T, tìm ngẫu nhiên một cạnh e' từ tập $E - E(T)$ sao cho $E(T) - \{e\} \cup \{e'\}$ cho ta cây Steiner T' (không quan tâm đến việc T' có tốt hơn T hay không). Cập nhật $T=T'$ và lặp lại k lần thao tác trên.

Thuật toán sau đây cho phép tìm kiếm cây Steiner ngẫu nhiên.

***Randsearch*(*T,k*)**

Đầu vào: Cây Steiner $T=(V(T),E(T))$ của đồ thị G, k là số cạnh cần thay thế ngẫu nhiên.
Đầu ra: Cây Steiner T sau khi đã cho thay thế k cạnh ngẫu nhiên

1. **for** i=1.. k{
2. Loại ngẫu nhiên cạnh $e \in T$;
3. Tìm ngẫu nhiên cạnh $e' \in E(G) - E(T)$ thỏa $T-\{e\}\cup\{e'\}$ là một cây Steiner;
4. $T=T-\{e\}\cup\{e'\}$;
5. }
6. **return** T;

F. Sơ đồ thuật toán Bees-Steiner

Thuật toán *Bees-Steiner* trước hết tạo quần thể ban đầu P, sau đó là lặp lại các thao tác: Sắp xếp các cá thể thuộc quần thể P và phân bố các cá thể vào các nhóm; mỗi cá thể thuộc nhóm 1 cho tìm kiếm lân cận k_1 lần, mỗi cá thể thuộc nhóm 2 cho tìm kiếm lân cận k_2 lần. Trong mỗi bước lặp, quần thể P đã được cập nhật thông qua các thao tác tìm kiếm lân cận và tìm kiếm ngẫu nhiên. Khi thuật toán dừng, cá thể tốt nhất tìm được trong quá trình thực hiện thuật toán được công bố làm lời giải cần tìm.

Thuật toán *Bees-Steiner* được mô tả như sau:

Bees-Steiner (*V,E*)

Đầu vào: Đồ thị $G=(V(G),E(G))$
Đầu ra: Cây Steiner có chi phí nhỏ nhất tìm được $\boldsymbol{T_{best}}$

1. ***InitPopulation***(V,E,N); // Tạo quần thể P gồm các cây Steiner $T_1, T_2,...,T_N$
2. ***while*** (*điều kiện dừng chưa thỏa*) {
3. ***SortPopulation***(P,N,h,p);
4. Cập nhật $T_{best} = T_1$;
5. for i=1..h
6. ***NeighSearch***(T_i, k_1);
7. for i=h +1..p
8. ***NeighSearch***(T_i, k_2);
9. for i=p+1..N
10. ***RandSearch***(T_i, k_3);
11. }
12. Cập nhật cá thể T_{best};
13. return T_{best};

III. THỰC NGHIỆM VÀ ĐÁNH GIÁ

Phần này chúng tôi tiến hành thực nghiệm thuật toán *Bees-Steiner* giải bài toán *SMT* trong trường hợp đồ thị thưa và đưa ra một số phân tích đánh giá về kết quả đạt được. Chúng tôi so sánh thuật toán *Bees-Steiner* với hai heuristic giải bài toán *SMT* gồm heuristic MST-Steiner của Bang Ye Wu và Kun-Mao Chao [1] và thuật toán heuristic dựa vào cây đường đi ngắn nhất (Shortest Path Tree-SPT) [1] và hai thuật toán metaheuristic giải bài toán *SMT* gồm thuật toán di truyền song song (PGA-Steiner) [10] và thuật toán tìm kiếm tabu (Tabu-Steiner) [2].

A. Dữ liệu thực nghiệm

Do hầu hết các đồ thị gặp trong thực tế ứng dụng là đồ thị thưa, nên chúng tôi sử dụng 38 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 [14]. Thông tin chi tiết về 38 bộ dữ liệu này được trình bày ở bảng *I*; trong đó các cột lần lượt ghi các thông tin về tên tập tin (*Test*) trong hệ thống dữ liệu thực nghiệm chuẩn, số đỉnh (*n*), số cạnh (*m*) và số đỉnh thuộc tập *terminal* (*y*) của từng đồ thị.

Bảng I. Thông tin về các bộ dữ liệu thực nghiệm

Test	*n*	*m*	y	Test	*n*	*m*	y
steinb1.txt	50	63	9	steinc1.txt	500	625	5
steinb2.txt	50	63	13	steinc2.txt	500	625	10
steinb3.txt	50	63	25	steinc3.txt	500	625	83
steinb4.txt	50	100	9	steinc4.txt	500	625	125
steinb5.txt	50	100	13	steinc5.txt	500	625	250
steinb6.txt	50	100	25	steinc6.txt	500	1000	5
steinb7.txt	75	94	13	steinc7.txt	500	1000	10
steinb8.txt	75	94	19	steinc8.txt	500	1000	83
steinb9.txt	75	94	38	steinc9.txt	500	1000	125
steinb10.txt	75	150	13	steinc10.txt	500	1000	250
steinb11.txt	75	150	19	steinc11.txt	500	2500	5
steinb12.txt	75	150	38	steinc12.txt	500	2500	10
steinb13.txt	100	125	17	steinc13.txt	500	2500	83
steinb14.txt	100	125	25	steinc14.txt	500	2500	125
steinb15.txt	100	125	50	steinc15.txt	500	2500	250
steinb16.txt	100	200	17	steinc16.txt	500	12500	5
steinb17.txt	100	200	25	steinc17.txt	500	12500	10
steinb18.txt	100	200	50	steinc18.txt	500	12500	83
-	-	-	-	steinc19.txt	500	12500	125
-	-	-	-	Steinc20.txt	500	12500	250

B. Môi trường thực nghiệm

Các thuật toán được viết 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 0 @ 2.20 GHz, RAM 4GB.

C. Kết quả thực nghiệm và đánh giá

1) Tham số thực nghiệm

Qua thực nghiệm, thuật toán *Bees-Steiner* sử dụng bộ tham số sau: Số bước lặp *Imax* = 300, $N = 75$, $h = 0.35 \times N$, $p = 0.85 \times N$, $k_1 = 0.50 \times n$, $k_2 = 0.25 \times n$, $k_3 = 0.01 \times n$. Thuật toán *Bees-Steiner* cho thực hiện 30 lần cho mỗi bộ dữ liệu.

2) Chất lượng lời giải

Kết quả thực nghiệm của các thuật toán được ghi nhận ở bảng *II* và bảng *III*. Kết quả thực nghiệm của các thuật toán PGA-Steiner và Tabu-Steiner được chúng tôi ghi nhận lại từ các công trình đã công bố liên quan; còn kết quả của các thuật toán MST-Steiner, SPT-Steiner, Bees-Steiner là do chúng tôi cài đặt; với mỗi đồ thị, kết quả của thuật toán Bees-Steiner là kết quả tốt nhất (có chi phí nhỏ nhất) sau 30 lần chạy.

Bảng II. Kết quả thực nghiệm thuật toán trên các bộ dữ liệu thuộc nhóm steinb

Test	MST-Steiner	SPT-Steiner	PGA-Steiner	Bees-Steiner
steinb1.txt	82	82	82	82
steinb2.txt	90	84	83	83
steinb3.txt	140	147	138	138
steinb4.txt	64	59	59	59
steinb5.txt	64	62	61	61
steinb6.txt	128	134	122	122
steinb7.txt	111	111	111	111
steinb8.txt	104	113	104	104
steinb9.txt	222	222	220	220
steinb10.txt	98	90	86	86
steinb11.txt	91	93	88	88
steinb12.txt	174	192	174	174
steinb13.txt	175	172	165	165
steinb14.txt	237	253	235	235
steinb15.txt	323	335	318	318
steinb16.txt	137	138	127	127
steinb17.txt	134	139	131	132
steinb18.txt	222	250	218	219

Với các đồ thị steinb 50 đỉnh: Chất lượng lời giải (tốt hơn; tương đương; kém hơn) của thuật toán Bees-Steiner so với các thuật toán MST-Steiner, SPT-Steiner, PGA-Steiner lần lượt là (5/6; 1/6; 0/6), (4/6; 2/6; 0/6), (0/6; 6/6; 0/6) bộ dữ liệu.

Với các đồ thị steinb 75 đỉnh: Chất lượng lời giải của thuật toán Bees-Steiner so với các thuật toán MST-Steiner, SPT-Steiner, PGA-Steiner lần lượt là (3/6; 3/6; 0/6), (5/6; 1/6; 0/6), (0/6; 6/6;0/6) bộ dữ liệu.

Với các đồ thị steinb 100 đỉnh: Chất lượng lời giải của thuật toán Bees-Steiner so với các thuật toán MST-Steiner, SPT-Steiner, PGA-Steiner lần lượt là (6/6; 0/6; 0/6), (6/6; 0/6; 0/6), (0/6; 4/6; 2/6) bộ dữ liệu.

Bảng III. Kết quả thực nghiệm thuật toán trên các bộ dữ liệu thuộc nhóm steinc

Test	MST-Steiner	SPT-Steiner	Tabu-Steiner	PGA-Steiner	Bees-Steiner
steinc1.txt	88	86	85	85	85
steinc2.txt	144	158	144	144	144
steinc3.txt	779	843	755	754	754
steinc4.txt	1114	1193	1081	1079	1079
steinc5.txt	1599	1706	1579	1579	1579
steinc6.txt	60	56	55	55	55
steinc7.txt	115	103	102	102	102
steinc8.txt	531	597	509	509	509
steinc9.txt	728	865	708	707	707
steinc10.txt	1117	1327	1093	1093	1094
steinc11.txt	37	32	32	32	32
steinc12.txt	49	46	46	46	46
steinc13.txt	274	322	258	258	260
steinc14.txt	337	417	324	323	324
steinc15.txt	571	703	556	556	556
steinc16.txt	13	12	11	11	11
steinc17.txt	19	19	18	18	18
steinc18.txt	125	146	117	113	116
steinc19.txt	158	195	149	146	149
steinc20.txt	269	339	267	267	267

Với các đồ thị steinc 500 đỉnh, 625 cạnh: Chất lượng lời giải của thuật toán Bees-Steiner so với các thuật toán MST-Steiner, SPT-Steiner, Tabu-Steiner, PGA-Steiner lần lượt là (4/5; 1/5; 0/5), (5/5; 0/5; 0/5), (2/5; 3/5; 0/5), (0/5; 5/5; 0/5) bộ dữ liệu.

Với các đồ thị steinc 500 đỉnh, 1000 cạnh: Chất lượng lời giải của thuật toán Bees-Steiner so với các thuật toán MST-Steiner, SPT-Steiner, Tabu-Steiner, PGA-Steiner lần lượt là (5/5; 0/5; 0/5), (5/5; 0/5; 0/5), (1/5; 3/5;1/5), (0/5; 4/5; 1/5) bộ dữ liệu.

Với các đồ thị steinc 500 đỉnh, 2500 cạnh: Chất lượng lời giải của thuật toán Bees-Steiner so với các thuật toán MST-Steiner, SPT-Steiner, Tabu-Steiner, PGA-Steiner lần lượt là (5/5; 0/5; 0/5), (3/5; 2/5; 0/5), (0/5; 4/5; 1/5), (0/5; 3/5; 2/5) bộ dữ liệu.

Với các đồ thị steinc 500 đỉnh, 12500 cạnh: Chất lượng lời giải của thuật toán Bees-Steiner so với các thuật toán MST-Steiner, SPT-Steiner, Tabu-Steiner, PGA-Steiner lần lượt là (5/5; 0/5; 0/5), (5/5; 0/5; 0/5), (1/5; 4/5; 0/5), (0/5; 3/5; 2/5) bộ dữ liệu.

IV. KẾT LUẬN

Trong bài báo này chúng tôi đã đề xuất thuật toán Bees-Steiner giải bài toán cây Steiner nhỏ nhất dựa trên sơ đồ của thuật toán bees cơ bản. Kết quả thực nghiệm trên 38 bộ dữ liệu là các đồ thị thưa trong hệ thống dữ liệu thực nghiệm chuẩn cho thấy thuật toán Bees-Steiner cho lời giải chất lượng tốt hơn thuật toán MST-Steiner 86.8% và cho chất lượng tương đương 13.2% bộ dữ liệu; cho lời giải có chất lượng tốt hơn thuật toán SPT-Steiner 86.8% và chất lượng tương đương 13.2%; cho lời giải có chất lượng tốt hơn thuật toán Tabu-Steiner 20.0%, tương đương 70.0% và kém hơn 10%; cho lời giải có chất lượng tốt hơn thuật toán PGA-Steiner 0.0%, tương đương 81.6% và kém hơn 18.4%.

TÀI LIỆU THAM KHẢO

[1] Bang Ye Wu, Kun-Mao Chao, Spanning Trees and Optimization Problems, Chapman&Hall/CRC, 2004, pp.13–139.

[2] Celso. C. Ribeiro, Mauricio.C. De Souza, Tabu Search for the Steiner Problem in Graphs, Networks, Vol.36, 2000, pp.138-146.

[3] Duc Truong Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi, The bees algorithm - A novel tool for complex optimisation problems, Proceedings of IPROMS 2006 Conference, Cardiff University, 2006, pp.454–461.

[4] Dusan Teodorovic, Bee Colony Optimization, Belgrade, Serbia, 2010.

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

[6] Hans L. Bodlaender, Johan M. M. van Rooij. Steiner Tree: Algorithms and Networks. 2015, pp.1-21.

[7] M. Hauptmann and M. Karpinski. A Compendium on Steiner Tree Problems, 2015, pp.1-50.

[8] Matteo Fischetti and et. Thinning out Steiner Trees. DIMACS, 2015, pp.1-19.

[9] Marcello Caleffi, Ian F. Akyildiz, Luigi Paura. On the Solution of the Steiner Tree NP-Hard Problem via Physarum BioNetwork. IEEE, 2015, pp.1092-1106.

[10] Nguyen Viet Huy, Nguyen Duc Nghia, Solving Graphical Steiner Tree Problem Using Parallel Genetic Algorithm,RIVF,IEEE, 2008.

[11] Phan Tấn Quốc, Nguyễn Đức Nghĩa. 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.

[12] Tom Decroos, Patrick De Causmaecker, Bart Demoen. Solving Euclidean Steiner Tree Problems with Multi Swarm Optimization. ACM, GECCO Companion '15, 2015, pp.1379-1380.

[13] Vũ Đình Hòa, Bài toán cây Steiner, http://vie.math.ac.vn/ mathclub/vdhoa.pdf.

[14] Website:http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files

[15] Xing-She Yang, Nature-Inspired Meta-heuristic Algorithms, LUNIVER PRESS, 2010, pp.53–62.

[16] Xiuzhen Cheng, Ding-Zhu Du.Steiner Trees in Industry. 2004, pp.193-216.

BEES ALGORITHM FOR SOLVING STEINER MINIMAL TREE PROBLEM IN SPARSE GRAPHS

Abstract: Steiner Minimal Tree (*SMT*) is a combinatorial optimization problem that has many important applications in science and engineering; this is the problem of class *NP*-hard. In this paper, we propose a new algorithm based on the basic bees algorithm to solve the *SMT* problem. We have been experimented the algorithm proposed on 38 sets of data in the standard experimental benchmark instances. Experimental results show that the algorithm proposed for solutions with better well-known heuristic algorithms and metaheuristic algorithms on instances.

Trần Việt Chương, Nhận học vị Thạc sĩ năm 2005. Hiện công tác tại Trung tâm Công nghệ thông tin và Truyền thông, Sở Thông tin và Truyền thông tỉnh Cà Mau. Lĩnh vực nghiên cứu: Hệ thống mạng máy tính, thuật toán tiến hóa.

Phan Tấn Quốc, Nhận học vị Tiến sỹ năm 2015. Hiện công tác tại Khoa Công nghệ thông tin, Trường Đại học Sài Gòn. Lĩnh vực nghiên cứu: Thuật toán metaheuristics.

Hà Hải Nam, Nhận học vị Tiến sỹ năm 2008 tại Vương quốc Anh, nhận học hàm Phó Giáo sư 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.