Tạp chí Tin học và Điều khiển học, T.29, S.3 (2013), 265–276

THUẬT TOÁN BẦY ONG GIẢI BÀI TOÁN CÂY KHUNG VỚI CHI PHÍ ĐỊNH TUYẾN NHỎ NHẤT*

PHAN TẤN QUỐC[1], NGUYỄN ĐỨC NGHĨA[2]

[1]*Khoa công nghệ thông tin, Trường Đại học Sài Gòn; Email: quocpt@sgu.edu.vn*

[2]*Viện Công nghệ Thông tin và Truyền thông, Trường Đại học Bách khoa Hà Nội*

Tóm tắt. Bài toán tìm cây khung chi phí định tuyến nhỏ nhất (Minimum Routing Cost Spanning Tree - MRCST) có thể được tìm thấy trong nhiều bài toán thiết kế mạng. Trong trường hợp tổng quát, bài toán MRCST đã được chứng minh là NP- khó. Bài báo này đề xuất thuật toán giải bài toán MRCST được phát triển dựa trên sơ đồ thuật toán bầy ong. Các kết quả tính toán thực nghiệm cho thấy thuật toán đề xuất cho lời giải với chất lượng tốt hơn thuật toán xấp xỉ Wong, các thuật toán meta-heuristics dạng quần thể như Max-Min Ant System (MMAS), thuật toán di truyền (GA), thuật toán Artificial Bee Colony (ABC) và một số thuật toán heuristic hiện biết. Cái giá phải trả để đạt được kết quả này là số lượng cây khung mà thuật toán của chúng tôi phải khảo sát là lớn hơn nhiều so với các thuật toán khác, chẳng hạn, trung bình là gấp 16000 lần so với thuật toán Wong và gấp 1.7 lần so với thuật toán ABC.

Từ khóa. Cây khung có chi phí định tuyến nhỏ nhất, thuật toán bầy ong, thuật toán meta-heuristic, trí tuệ bầy đàn.

Abstract. The task to find Minimum Routing Cost Spanning Tree (MRCST) can be found in many network design problems. In general cases, the MRCST problem is proved to be NP-hard. This paper proposes an algorithm to solve MRCST problem based on the schema of bee algorithm. The computational experiment results show that our proposed algorithm outperforms the Wong's algorithm, population-based meta-heuristics like Max-Min Ant System (MMAS), Genetic Algorithm (GA), Artificial Bee Colony algorithm (ABC), and other well-known heuristic algorithms. The cost to get this result is the large number of spanning trees examined by our algorithm compared to other methods, e.g., 16000 times and 1.7 times more than that of Wong's algorithm and ABC algorithm on average, respectively.

Key words. Minimum routing cost spanning tree, bee algorithms; meta-heuristic algorithms, swarm intelligence.

1. GIỚI THIỆU

Phần này sẽ nhắc lại bài toán MRCST và khảo sát một số thuật toán giải bài toán MRCST đã được đề xuất trong những năm gần đây.

*Công trình này nhận được sự tài trợ từ Đề tài khoa học công nghệ của Bộ Giáo dục và Đào tạo: Xây dựng giải pháp thiết kế mạng chịu lỗi tối ưu sử dụng các kỹ thật meta-heuristics, mã số B2012-01-28.

1.1. Phát biểu bài toán

Định nghĩa 1. (Chi phí định tuyến [1]) Cho $G = (V, E, w)$ là một đồ thị vô hướng liên thông có trọng số (chi phí) 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, $e \in E$. Giả sử 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 của các cạnh trên đường đi nối đỉnh u với đỉnh v trên cây T. Chi phí định tuyến của T, ký hiệu là $C(T)$, được xác định là tổng các chi phí định tuyến giữa mọi cặp đỉnh thuộc cây T, tức là

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

Bài toán đặt ra là tìm một cây khung với chi phí định tuyến nhỏ nhất trong số tất cả các cây khung của đồ thị G. Việc tính chi phí định tuyến của cây khung trực tiếp theo Định nghĩa 1 sẽ đòi hỏi thời gian $O(n^2)$. Tuy nhiên, trên cơ sở đưa ra khái niệm "tải định tuyến" ("routing load"), công trình [1] đã chỉ ra cách tính chi phí định tuyến của cây khung với độ phức tạp tuyến tính.

Định nghĩa 2. (Tải định tuyến [1]) Cho cây khung T với tập cạnh là $E(T)$. Nếu loại khỏi 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 là $V(T_1)$ và $V(T_2)$. Tải định tuyến của cạnh e được định nghĩa là: $l(T, e) = 2 \times |V(T_1)| \times |V(T_2)|$. Khi đó chi phí định tuyến của cây khung T có thể tính theo công thức sau đây

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

Xây dựng cây khung chi phí định tuyến nhỏ nhất là tương đương với việc xây dựng cây khung sao cho độ dài trung bình giữa mọi cặp đỉnh là nhỏ nhất. Bài toán này có ý nghĩa ứng dụng quan trọng trong thiết kế mạng, đặc biệt là ở các mạng ngang hàng khi mà các nút có xác suất truyền tin và độ ưu tiên là như nhau. Bài toán có ứng dụng thiết thực trong việc tối ưu hóa chi phí định tuyến của các thiết bị switch và bridge (về xuất xứ và ứng dụng của bài toán có thể xem thêm ở các công trình [1, 2, 8]).

Bài toán MRCST được xác định là bài toán thuộc lớp NP-khó [1, 15]. Trọng số trên mỗi cạnh và cấu trúc của cây khung là hai yếu tố cơ bản ảnh hưởng đến chi phí định tuyến của cây khung, trong đó yếu tố cấu trúc của cây khung có ảnh hưởng lớn hơn đối với những đồ thị mà trọng số của các cạnh là không quá cách biệt. Hiện đã có những phương pháp giải gần đúng bài toán MRCST dựa trên các cách tiếp cận khác nhau như sơ đồ xấp xỉ, heuristic, meta-heuristic.

1.2. Khảo sát các thuật toán giải MRCST

Thuật toán xấp xỉ

Thứ nhất là thuật toán Wong được đề xuất bởi Richard Wong vào năm 1980. Thuật toán Wong là thuật toán xấp xỉ với cận tỉ lệ 2 (nghĩa là chi phí của cây khung tìm được theo thuật toán không vượt quá 2 lần chi phí của cây khung tối ưu) và có độ phức tạp là $O(nm+n^2 \log n)$. Thuật toán Wong sử dụng khái niệm cây đường đi ngắn nhất (Shortest Path Tree – SPT); cây đường đi ngắn nhất có gốc tại đỉnh u là cây khung có các cạnh là hợp của các cạnh trên các đường đi ngắn nhất xuất phát từ đỉnh u đến các đỉnh còn lại của đồ thị [1]. Thuật toán

Wong trước hết tìm tất cả các SPT xuất phát từ các đỉnh của đồ thị, sau đó, trong số chúng chọn ra SPT có chi phí định tuyến nhỏ nhất làm lời giải cần tìm.

Thứ hai là thuật toán Add được đề xuất bởi Vic Grout vào năm 2005 [2], có độ phức tạp là $O(n \log n)$. Thuật toán Add bỏ qua trọng số của các cạnh và lấy bậc của các đỉnh làm điều kiện tiên quyết để xây dựng cây khung. Thuật toán Add chỉ sử dụng hiệu quả đối với đồ thị đồng nhất và đồ thị gần đồng nhất (là loại đồ thị mà trọng số của các cạnh sai khác nhau không đáng kể - trong bài báo này sẽ gọi là đồ thị đồng nhất).

Thứ ba là thuật toán Campos được đề xuất bởi nhóm tác giả Rui Campos và Manuel Ricardo vào năm 2008 [8]. Thuật toán này cũng có cận tỉ lệ 2 và với độ phức tạp là $O(m + n \log n)$.

Thuật toán heuristic

Thứ nhất là thuật toán xóa cạnh trước rồi chèn cạnh sau [13]: Bắt đầu từ cây khung T tìm được nhờ thuật toán Wong. Ở mỗi lần lặp, với mỗi cạnh e của cây T ta tìm cạnh $e' \in E - E(T)$ sao cho cây khung T' thu được từ T bởi việc thay cạnh e bởi e' có chi phí định tuyến là nhỏ nhất. Nếu $C(T') < C(T)$ thì thay T bằng T'. Thuật toán dừng nếu sau một lần duyệt qua tất cả các cạnh $e \in T$ mà vẫn không tìm được $e' \in E - E(T)$ tốt hơn để thay thế.

Thứ hai là thuật toán chèn cạnh trước rồi xóa cạnh sau [13]. Bắt đầu từ cây khung T tìm được nhờ thuật toán Wong, ở mỗi lần lặp, với mỗi cạnh $e \in E - E(T)$, bổ sung cạnh e vào cây T. Khi đó tập cạnh $E(T) \cup \{e\}$ sẽ chứa chu trình và cần tìm cạnh trên chu trình này mà việc loại bỏ nó dẫn đến cây khung với chi phí nhỏ nhất. Cập nhật cây T nếu thu được kết quả tốt hơn. Thuật toán dừng nếu sau một lần duyệt qua tất cả các cạnh $e \in E - E(T)$ mà vẫn không cải thiện được T.

Thuật toán meta-heuristic

Thứ nhất là các thuật toán meta-heuristic dạng cá thể. Chẳng hạn thuật toán Stochastic Hill Climber Search (SHCS) [3] và thuật toán Local Search (LS) [7] với ý tưởng cơ bản là từ một lời giải hiện tại được chọn là T, ở mỗi bước lặp, thuật toán sẽ tìm k lân cận trong một tập con các lân cận của T. Nếu tìm được một lời giải tốt hơn T thì nó sẽ trở thành lời giải hiện tại ở bước lặp kế tiếp. Nếu lời giải tốt nhất hiện tại không được cải thiện qua một số bước lặp định trước, thuật toán sẽ tiến hành xáo trộn lời giải hiện tại bằng cách cho thay thế ngẫu nhiên một số cạnh của lời giải hiện tại bằng một số cạnh hợp lệ khác. Quá trình giải sẽ kết thúc khi thuật toán thực hiện đủ một số lần lặp xác định trước.

Thứ hai là các thuật toán meta-heuristic dạng quần thể như thuật toán bầy kiến [11], thuật toán di truyền [3, 14], thuật toán bầy ong [12],...

Có thể nhận xét rằng: Các thuật toán xấp xỉ và heuristic cho lời giải chất lượng chưa cao, các thuật toán meta-heuristic cho lời giải tốt hơn. Bài báo này trình bày thuật toán bầy ong giải bài toán MRCST, và với một số cải tiến cụ thể đã cải thiện được chất lượng của lời giải.

Thuật toán bầy ong

Bầy ong mật trong tự nhiên tìm kiếm thức ăn theo quy trình sau: Đầu tiên các ong do thám sẽ được cử đi thăm dò các vùng thức ăn, các ong do thám sẽ di chuyển ngẫu nhiên từ bụi hoa này sang bụi hoa khác, sau đó chúng quay về tổ và thông tin cho cả bầy về hướng di chuyển, khoảng cách từ tổ đến vùng thức ăn và chất lượng của các vùng thức ăn, những thông tin này sẽ làm cho bầy ong bay đến các vùng thức ăn nhanh chóng và chính xác hơn. Một vấn đề tự nhiên nhưng là điểm trọng tâm cho thuật toán bầy ong có thể rút ra là: Nơi nào có thức ăn dồi dào hơn thì nơi đó sẽ có nhiều ong được cử đến hơn [4].

Các thuật toán mô phỏng theo quá trình tìm kiếm thức ăn của loài ong mật (gọi chung là

các thuật toán bầy ong) đã được biết đến trong các công trình của các nhóm tác giả Karaboga và Basturk (2007) [5, 6], Dusan Teodorovic (2010) [9], Alok Singh và Shyam Sundar (2011) [12] dưới tên gọi là ABC (Artificial Bee Colony), và trong các công trình của Duc Truong Pham (2005) [4], Xing-She Yang (2010) [10] dưới tên gọi thuật toán bầy ong (Bee Algorithm).

Thuật toán bầy ong sử dụng ba chiến lược trọng tâm là tìm kiếm lân cận, tìm kiếm ngẫu nhiên và tìm kiếm nhiều hơn ở những vùng không gian tiềm năng. Thuật toán bầy ong được cho là có khả năng thoát khỏi tối ưu cục bộ và do đó nó có thể tìm được lời giải tối ưu toàn cục, thuật toán bầy ong được đánh giá là một trong những thuật toán meta-heuristic thích hợp cho việc giải các bài toán tối ưu tổ hợp khó [4].

Thuật toán bầy ong có những ý tưởng khác biệt so với thuật toán ABC [4, 5, 6]. Thuật toán đề xuất trong bài báo này dựa trên sơ đồ thuật toán bầy ong cơ bản của nhóm tác giả Duc Truong Pham (2005), thuật toán này là khác biệt so với thuật toán giải MRCST dựa trên sơ đồ ABC [12].

2. THUẬT TOÁN BẦY ONG GIẢI BÀI TOÁN CÂY KHUNG VỚI CHI PHÍ ĐỊNH TUYẾN NHỎ NHẤT

Mục này trình bày đề xuất áp dụng thuật toán bầy ong giải bài toán MRCST (ta gọi thuật toán này là BEE-MRCST).

2.1. Mã hóa cây khung

Trong thuật toán BEE-MRCST, ta sử dụng phương pháp mã hóa dạng cạnh để mã hóa cây khung, mỗi cây khung được mã hóa bởi một chuỗi gồm $n-1$ số nguyên, trong đó mỗi số nguyên là chỉ số của cạnh của đồ thị tham gia vào cây khung (các cạnh của đồ thị được đánh số từ 1 đến m). Trong thuật toán BEE-MRCST, ta đồng nhất hai khái niệm cá thể và cây khung khi diễn đạt.

2.2. Tính chi phí định tuyến của cây khung

Việc tính chi phí định tuyến của cây khung T theo công trình [1] có thể tiến hành như sau: Thực hiện duyệt cây T theo chiều sâu bắt đầu từ đỉnh v_1 ta thu được biểu diễn của T dưới dạng cây có gốc tại đỉnh v_1. Gọi V_u là số lượng đỉnh của cây con có gốc là u. Với mỗi đỉnh u của cây T, $u \neq v_1$, ký hiệu $e_u = (parent(u), u)$; trong đó $parent(u)$ là cha của u trong cây T. Sau đây là đoạn mã giả mô tả việc tính routing cost theo công thức (2).

INPUT: Cây khung $T = (V(T), E(T))$ được biểu diễn như cây có gốc tại v_1

OUTPUT: Chi phí định tuyến của T

$routingcost(T)$

{ if $T = \emptyset$ return $+\infty$; // Ta qui ước cây rỗng có chi phí $+\infty$

Duyệt cây T theo chiều sâu bắt đầu từ đỉnh v_1;

$C = 0$;

for (mỗi đỉnh $u \in V(T)$)

if $(u! = v_1)\{l(e_u) = 2 \times V_u \times (n - V_u)$;

$C = C + l(e_u) + w(e_u)$;

}
return C;
}

2.3. Tạo quần thể ban đầu

Thuật toán BEE-MRCST sử dụng hai cách sau đây để tạo quần thể ban đầu: Thứ nhất, mỗi cá thể là một cây đường đi ngắn nhất có gốc xuất phát từ một đỉnh nào đó của đồ thị. Cách này cho phép tạo ra các cá thể với chi phí tương đối tốt, nhưng chỉ tạo được quần thể với kích thước không vượt quá số đỉnh của đồ thị. Thứ hai, mỗi cá thể là một cây khung được tạo theo thuật toán tựa Prim: Bắt đầu từ cây chỉ gồm một đỉnh nào đó của đồ thị, để xây dựng cây khung, thuật toán thực hiện $n-1$ bước lặp. Ở mỗi bước lặp, trong số các đỉnh chưa tham gia vào cây khung đang xây dựng ta chọn một đỉnh kề với ít nhất một đỉnh trong cây khung đang xây dựng. Đỉnh được chọn và cạnh nối nó với đỉnh của cây khung đang xây dựng sẽ được bổ sung vào cây. Cách này cho phép tạo được quần thể với kích thước lớn hơn, tính đa dạng của quần thể được bảo đảm, nhưng chất lượng của quần thể sẽ không cao. Các cách tạo quần thể ban đầu này đã được nhóm tác giả đề xuất trong [11, 14]. Trong thuật toán BEE-MRCST, ta sẽ cố định kích thước quần thể là N, khi $N < n$ thì chọn cách thứ nhất để tạo quần thể, ngược lại, chọn cách thứ hai. Sau đây là đoạn mã giả cho việc tạo quần thể ban đầu.

INPUT: $G = (V, E, w)$

OUTPUT: Quần thể Q gồm N cây khung $T_1, T_2, ..., T_N$ của đồ thị G

$initpopulation(V, E, w)$

{ $Q = \emptyset$;
if ($N \geq n$) for $i = 1..N\{T = likeprim(V, E);\ Q = Q \cup \{T\};\}$
else
for $i = 1..N\{$
Chọn ngẫu nhiên một đỉnh u trong số các đỉnh chưa được chọn trước đó;
$T = sptwong(V, E, w, u);\ Q = Q \cup \{T\};$
}
}

// Hàm $sptwong(V, E, w, s)$ trả lại $T = (V, E(T))$ là SPT xuất phát từ đỉnh s trên G

$sptwong(V, E, w, s)$

{ $Dijkstra(V, E, w, s)$; // Thực hiện thuật toán $Dijkstra$ tìm cây đường đi ngắn nhất từ đỉnh s đến các

// đỉnh còn lại, ta thu được $prev(v)$ - đỉnh đi trước đỉnh v trong đường đi ngắn nhất từ s đến v.

Đặt $E(T) = \{(v, prev(v)) : v \in V - \{s\}\}$;

return cây khung $T = (V, E(T))$;

}

//hàm $likeprim(V, E)$ trả lại cây khung ngẫu nhiên $T = (V(T), E(T))$ theo thuật toán tựa Prim

$likeprim(V, E)$

{ $V(T) = u$; // u là một đỉnh được chọn ngẫu nhiên trong số các đỉnh của $G = (V, E)$

while ($|V(T)| < n$){
Chọn ngẫu nhiên đỉnh $v \in V - V(T)$ sao cho v có kề với một đỉnh z nào đó trong $V(T)$;
$V(T) = V(T) \cup \{v\}$; $E(T) = E(T) \cup \{(v, z)\}$;
}
return cây khung $T = (V, E(T))$;
}

2.4. Sắp xếp quần thể

Quần thể có N cá thể, sử dụng thuật toán Quick Sort sắp xếp các cá thể trong quần thể Q theo chiều tăng dần của chi phí định tuyến các cá thể. Sau khi sắp xếp, ta phân bố các cá thể vào ba loại vùng: h cá thể tốt nhất phân vào h-vùng, $p - h$ cá thể tiếp theo phân vào ph-vùng và $N - p$ cá thể cuối cùng phân vào np-vùng. Hàm sắp xếp quần thể và và phân bố các cá thể vào các vùng được đặt tên là $sortpopulation(Q, N)$.

2.5. Tìm kiếm lân cận

Thủ tục tìm kiếm lân cận bắt đầu từ cây khung T được tiến hành như sau: Loại ngẫu nhiên khỏi T một cạnh e, sau đó thực hiện q lần thao tác sau đây: chọn ngẫu nhiên cạnh e' từ tập $E - E(T)$, nếu tập cạnh $E(T) - \{e\} \cup \{e'\}$ cho ta cây khung T' có chi phí tốt hơn T thì ghi nhận cây khung này. Thủ tục trên sẽ được lặp lại k lần đối với cây khung T, và trong số các cây khung được ghi nhận chọn ra T^* là cây khung tốt nhất, rồi đặt $T = T^*$. Như vậy quần thể Q được cập nhật sau khi thực hiện xong tìm kiếm lân cận. Tìm kiếm lân cận vừa đề xuất là khác với các tiếp cận đã được công bố ở các công trình [11, 12, 14]. Hàm tìm kiếm lân cận vừa mô tả được đặt tên là $neighsearch(T, k, q)$. Sau đây là mã giả mô tả việc tìm kiếm cây khung lân cận tốt hơn.

INPUT: Cây khung $T = (V, E(T))$ và số nguyên dương k

OUTPUT: Thay thế cây khung T bởi cây khung tốt nhất trong lân cận gồm k cây khung được tạo ngẫu nhiên chỉ khác T một cạnh.

$neighsearch(T, k)$
{ $T^* = \emptyset$; //T^* - là cây khung tốt nhất trong lân cận của cây T
for $i = 1..k${
Xóa một cạnh e được chọn ngẫu nhiên trong $E(T)$;
for $i = 1..q${ Chọn ngẫu nhiên cạnh $e' \in E - E(T)$;
if ($T' = (V, E(T) - \{e\} \cup \{e'\})$ là cây khung)
and ($routingcost(T') < routingcost(T^*)) T^* = T'$;
}
if $routingcost(T^*) < routingcost(T)$ $Q = Q - \{T\} \cup \{T^*\}$; // thay cây khung T bởi cây khung T^*;
}

2.6. Tìm kiếm lân cận ngẫu nhiên

Trong tìm kiếm lân cận ở mục trên, cây khung tìm được ở bước sau luôn không tồi hơn cây khung ở bước trước đó. Điều này dễ làm cho quá trình tìm kiếm bị rơi vào ngõ cụt. Khi đó cần thực hiện việc đa dạng hóa lời giải nhờ thực hiện tìm kiếm lân cận ngẫu nhiên. Trong BEE-MRCST ta gọi việc làm này là xáo trộn lời giải. Công việc này được thực hiện ở tất cả các vùng thuộc *np*-vùng và các vùng thuộc *h*-vùng và *ph*-vùng đã khai thác cạn, nghĩa là ở các vùng mà chất lượng lời giải không được cải thiện qua một số lần lặp xác định. Tìm kiếm lân cận ngẫu nhiên cho cây khung T được tiến hành tương tự như tìm kiếm lân cận: Xóa 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 khung T' (không quan tâm đến việc T' có tốt hơn T hay không). Lặp lại k lần thao tác trên và chọn ra trong số các cây khung thu được, cây khung tốt nhất thay thế cho T. Như vậy quần thể Q được cập nhật sau khi thực hiện xong tìm kiếm lân cận ngẫu nhiên. Mã giả cho việc tìm kiếm cây khung lân cận ngẫu nhiên được mô tả như sau.

INPUT: Cây khung $T = (V, E(T))$ và số nguyên dương k

OUTPUT: Cây khung T sau khi đã thay thế ngẫu nhiên k cạnh

$randsearch(T, k)$

{ $T^* = \emptyset$; //T^* là cây khung tốt nhất trong lân cận ngẫu nhiên của T

for $i = 1..k${

Xóa một cạnh e được chọn ngẫu nhiên trong cây T;

Tìm ngẫu nhiên một cạnh e' từ tập $E - E(T)$ sao cho $T' = (V, E(T) - \{e\} \cup \{e'\})$

là cây khung;

if ($routingcost(T') < routingcost(T^*))T^* = T'$;

}

$Q = Q - \{T\} \cup \{T^*\}$; // thay cây khung T bởi cây khung T^*

}

2.7. Tìm lời giải tốt nhất

Để đưa ra cây khung được chọn làm lời giải của bài toán, ta tiến hành so sánh cây khung tốt nhất tìm được với các cá thể thuộc quần thể Q khi kết thúc quá trình tìm kiếm. Sau đây là mã giả mô tả việc tìm lời giải tốt nhất của bài toán.

INPUT: Quần thể Q gồm N cây khung $T_1, T_2, ..., T_N$, và $bestT$ là cây khung tốt nhất hiện tại

OUTPUT: $bestT$ – cây khung được chọn làm lời giải của bài toán

$bestsolution(Q, N, bestT)$

{ for $i = 1..n$

if $routingcost(T_i) < routingcost(bestT)$ $bestT = T_i$;

return $bestT$;

}

2.8. Thuật toán BEE-MRCST

BEE-MRCST trước hết là tạo quần thể ban đầu Q, sau đó là lặp lại các thao tác: Sắp xếp quần thể Q và phân bố các cá thể vào mỗi loại vùng; mỗi cá thể thuộc *h*-vùng cho tìm kiếm

lân cận k_1 lần, mỗi cá thể thuộc *ph*-vùng cho tìm kiếm lân cận k_2 lần, mỗi cá thể đã được khai thác cạn cho tìm kiếm lân cận ngẫu nhiên k_3 lần, mỗi cá thể thuộc *np*-vùng cho tìm kiếm lân cận ngẫu nhiên k_4 lần. Trong mỗi bước lặp, quần thể Q đã được cập nhật thông qua các thao tác tìm kiếm lân cận và tìm kiếm lân cận 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. Sau đây là mã giả minh họa cho việc tìm kiếm cây khung có chi phí định tuyến nhỏ nhất.

Thuật toán BEE-MRCST

INPUT: $G = (V, E, w)$

OUTPUT: Cây khung có chi phí định tuyến nhỏ nhất tìm được $bestT$

$BEE - MRCST(V, E, w)\{$

$initpopulation(V, E, w, N)$; // Tạo quần thể Q gồm các cây khung $T_1, T_2, ..., T_N$

while (điều kiện dừng chưa thỏa)

{ $sortpopulation(Q, N)$; // sắp xếp các cá thể trong Q theo thứ tự $C(T_1) \leq C(T_2) \leq ... \leq C(T_N)$

Cập nhật $bestT = T_1$;

for $i = 1..h\{neighsearch(T_i, k_1)$;

if (sau $itermax$ lần lặp mà T_i không được cải thiện) $randsearch(T_i, k_3)$;

}

for $i = h + 1..p\{neighsearch(T_i, k_2)$;

if (sau $itermax$ lần lặp mà T_i không được cải thiện) $randsearch(T_i, k_3)$;

}

for $i = p + 1..N$ $randsearch(T_i, k_4)$;

}

$bestsolution(Q, N, bestT)$;

return $bestT$;

}

Điều kiện dừng thuật toán thường được chọn là: hoặc là thực hiện đủ số tối đa lần lặp định trước hoặc là sau một số định trước lần lặp lời giải tốt nhất hiện có không được cải thiện. Trong bài báo này, thuật toán kết thúc khi thực hiện đủ $Imax = 350$ lần lặp.

Khác với sơ đồ thuật toán bầy ong tổng quát, thuật toán BEE-MRCST không sử dụng tham số *ngh*. Việc phân vùng tìm kiếm trong BEE-MRCST được thực hiện dựa vào việc sắp xếp cả quần thể. Những vùng đã khai thác cạn được mở rộng nhờ tìm kiếm lân cận ngẫu nhiên. Thuật toán BEE-MRCST là khác biệt so với thuật toán ABC đề xuất trong [5, 6, 12] ở phương pháp tạo quần thể ban đầu - có sử dụng thuật toán Wong, thuật toán tìm kiếm lân cận và phương pháp phân vùng tìm kiếm.

Đánh giá thời gian tính của thuật toán BEE-MRCST

Ta có thể đánh giá thời gian tính của một lần lặp của thuật toán BEE-MRCST như sau. Hàm Sắp xếp quần thể có thời gian tính $O(N \log N)$, hàm xử lý cho *h*-vùng có thời gian tính $O(n^2)$, hàm xử lý cho *ph*-vùng có thời gian tính $O(n^2)$, hàm xử lý cho *np*-vùng có thời gian tính $O(n)$. Tổng cộng, một lần lặp của thuật toán BEE-MRCST đòi hỏi thời gian tính $O(n^2 + N \log N)$.

3. KẾT QUẢ THỰC NGHIỆM

3.1. Xây dựng bộ dữ liệu

Như đã đề cập ở phần đầu bài báo, hiệu quả của thuật toán giải bài toán MRCST phụ thuộc vào hai đặc trưng quan trọng của dữ liệu, đó là cấu trúc của đồ thị và trọng số trên các cạnh của đồ thị. Liên quan đến trọng số, chúng tôi sẽ tiến hành thực nghiệm với các đồ thị có trọng số trên các cạnh là sai khác nhau lớn hoặc sai khác nhau nhỏ hoặc trọng số các cạnh là như nhau. Liên quan đến cấu trúc đồ thị, có thể tiến hành thực nghiệm với các loại đồ thị như đồ thị đầy đủ (đồ thị dày) và đồ thị có các cạnh được phân bố đều hoặc không đều giữa các đỉnh (đồ thị thưa), đồ thị tổng quát (ngẫu nhiên về trọng số và cấu trúc cạnh).

Đồ thị tổng quát. Trước hết xây dựng ngẫu nhiên một cây khung T gồm n đỉnh và $n-1$ cạnh, sau đó tiếp tục thêm ngẫu nhiên $m-(n-1)$ cạnh khác nữa vào T; trọng số các cạnh của đồ thị là số nguyên ngẫu nhiên trong đoạn $[1..max\_weight]$.

Đồ thị đồng nhất. Việc sinh ngẫu nhiên các đồ thị đồng nhất được thực hiện tương tự như đối với trường hợp đồ thị tổng quát; điểm khác biệt duy nhất là trọng số các cạnh được sinh phải thỏa mãn điều kiện đồng nhất hoặc gần đồng nhất: đặt $\delta = random(r), maxcost = random(max\_weight) + \delta + 1$; khi đó $maxcost + random(2 \times \delta + 1) - \delta$ là trọng số cạnh; các trọng số này sẽ sai khác từ $-(r-1)$ đến $(r-1)$. Khi r giảm, tính đồng nhất của đồ thị sẽ tăng (hàm $random(r)$ trả về một số nguyên ngẫu nhiên trong phạm vi từ 0 đến $(r-1)$).

Đồ thị có các cạnh được phân bố đều. Đồ thị có các cạnh được phân bố đều giữa các đỉnh là đồ thị mà bậc của các đỉnh là chênh lệch nhau không đáng kể.

Trước hết sinh ngẫu nhiên cây khung T có $n-1$ cạnh mà mỗi cạnh có các đỉnh tối đa là bậc 2 (đơn giản là đường nối n đỉnh); sau đó chèn thêm $m-(n-1)$ cạnh ngẫu nhiên khác vào T để được đồ thị G. Cạnh (u, v) được chèn vào G nếu bậc của các đỉnh u, v (tính trên T) $\leq$[2m/n], trọng số các cạnh của đồ thị là số nguyên ngẫu nhiên trong đoạn $[1..max\_weight]$.

Đồ thị có các cạnh được phân bố không đều. Trước hết tạo một cây khung T ngẫu nhiên đi qua n đỉnh: T được tạo có α cụm hình sao (các cạnh có chung đỉnh một đỉnh), mỗi cụm có $n/\alpha - 2$ cạnh và $n - \alpha \times (n/\alpha - 1)$ đỉnh còn lại sẽ tạo thành cụm hình sao cuối cùng, nối α cụm này lại với nhau để tạo thành một cây khung. Việc sinh thêm $m-(n-1)$ cạnh còn lại được thực hiện như đối với đồ thị tổng quát.

Đồ thị đầy đủ. Đồ thị có n đỉnh có số cạnh là $m = (n-1) \times n/2$, trọng số trên các cạnh của đồ thị được tạo là các số nguyên ngẫu nhiên trong đoạn $[1..max\_weight]$.

3.2. Môi trường và dữ liệu thực nghiệm

Thuật toán BEE-MRCST được cài đặt trên ngôn ngữ C++ sử dụng trình biên dịch CFREE 5 và chạy trên máy tính cấu hình 4GB RAM, CPU INTEL 2.20GHz.

Ở đây, ta tiến hành thực nghiệm thuật toán BEE-MRCST trên 5 loại đồ thị đã đề cập ở trên. Dữ liệu thực nghiệm được phát sinh ngẫu nhiên như đề xuất từ các bài báo [11, 12] với hai hệ thống test: Hệ thống test 1 có 96 test (trong đó có 18 test STEI-01..STEI-18 được lấy từ trang WEB[2]) và hệ thống test 2 có 75 test. Trong hệ thống test 1 các đồ thị được sinh có số đỉnh trong phạm vi [20..100], số cạnh trong phạm vi [50..1200] và trọng số các cạnh trong phạm vi 1..2500. Trong hệ thống test 2 các đồ thị được giữ nguyên số đỉnh và số cạnh như đối với hệ thống test 1, cấu trúc của đồ thị được sinh ngẫu nhiên, trọng số các cạnh được cho ngẫu nhiên phạm vi 1..100.

3.3. Các tham số thực nghiệm

Kết quả thực nghiệm của thuật toán BEE-MRCST được so sánh chi tiết với các thuật toán Wong [1], SHCS [3], LS [7], MMAS [11], GA [14] và ABC [12]. Chi phí định tuyến trong các bảng thực nghiệm được ghi nhận bằng 1/2 giá trị tính theo công thức (2). Trong thực nghiệm, thuật toán Wong thực hiện 1 lần chạy, thuật toán SHCS thực hiện 20 lần chạy; mỗi lần chạy có 2000 vòng lặp và trong mỗi vòng lặp mỗi cá thể sinh 5 lân cận; thuật toán LS cho thực hiện 10 lần chạy; mỗi lần chạy có 20000 vòng lặp (các thuật toán SHCS, LS được khởi tạo bằng thuật toán Wong); thuật toán MMAS được thực hiện 1 lần chạy khi cho 20 kiến dò đường 300 lần; các thuật toán GA, BEE-MRCST được thực hiện 10 lần chạy, và ở mỗi lần chạy thuật toán dừng khi thực hiện đủ $Imax = 350$ lần lặp; thuật toán ABC được thực hiện 30 lần chạy và tất cả các tham số đều được giữ nguyên như tác giả đã đề xuất [12]. Đối với mỗi thuật toán, kết quả được ghi nhận là kết quả tốt nhất của các lần chạy.

3.4. Đánh giá kết quả thực nghiệm

Về chi phí định tuyến

Kết quả thực nghiệm BEE-MRCST cho từng loại đồ thị ứng với từng thuật toán được tổng hợp lại trong Bảng 1 (chi tiết hơn có thể xem tại trang WEB[1]). Nội dung của bảng 1 cho biết số lượng (SL) bộ test cho kết quả tốt hơn (ghi nhận bởi ký hiệu "<") hoặc bằng nhau (ghi nhận bởi ký hiệu "=") hoặc kém hơn (ghi nhận bởi ký hiệu ">") khi so sánh thuật toán BEE-MRCST với các thuật toán Wong, SHCS, LS, MMAS, GA và ABC; đồng thời cũng cho biết tỷ lệ phần trăm (%) tương ứng. Bảng 1 ghi nhân kết quả so sánh riêng cho từng dạng đồ thị và tổng hợp cho tất cả các dạng đồ thị đã đề cập. Kết quả ghi nhận trong Bảng 1 cho thấy với hầu hết các bộ test, BEE-MRCST cho kết quả tốt hơn hẳn các thuật toán Wong, SHCS, LS, MMAS, GA và ABC (chương trình cài đặt thuật toán ABC do nhóm tác giả đề xuất).

Bảng 1. So sánh thuật toán BEE-MRCST với các thuật toán khác trên hệ thống test 1

BEE	Wong		SHCS		LS		MMAS		GA		ABC	
(96 test)	*SL*	*%*	*SL*	*%*	*SL*	*%*	*SL*	*%*	*SL*	*%*	*SL*	*%*
Dạng các đồ thị tổng quát (36 test)												
<	34	94.4	1	2.8	0	0.0	21	58.3	0	0.0	1	2.8
=	2	5.6	35	97.2	36	100.0	15	41.7	36	100.0	35	97.2
>	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0
Dạng các đồ thị đồng nhất (15 test)												
<	15	100.0	14	93.3	4	26.7	15	100.0	3	20.0	2	13.3
=	0	0.0	1	6.7	11	73.3	0	0.0	12	80.0	13	86.7
>	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0
Dạng các đồ thị có các cạnh được phân bố đều (15 test)												
<	13	86.7	2	13.3	0	0.0	9	60.0	0	0.0	0	0.0
=	2	13.3	13	86.7	15	100.0	6	40.0	15	100.0	15	100.0
>	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0
Dạng các đồ thị có các cạnh được phân bố không đều (15 test)												
<	14	93.3	2	13.3	0	0.0	8	53.3	0	0.0	0	0.0
=	1	6.7	13	86.7	15	100.0	7	46.7	15	100.0	15	100.0
>	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0
Dạng các đồ thị đầy đủ (15 test)												
<	15	100.0	2	13.3	0	0.0	10	66.7	0	0.0	0	0.0
=	0	0.0	13	86.7	15	100.0	5	33.3	15	100.0	15	100.0
>	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0
Tổng hợp cho các dạng đồ thị (96 test)												
<	91	94.8	21	21.9	4	4.2	63	65.6	3	3.1	3	3.1
=	5	5.2	75	78.1	92	95.8	33	34.4	93	96.9	93	96.9
>	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0

Kết quả thực nghiệm các thuật toán trên với hệ thống test 2 được trình bày trong bảng 2.

Bảng 2. So sánh thuật toán BEE-MRCST với các thuật toán khác trên hệ thống test 2

BEE	Wong		SHCS		LS		MMAS		GA		ABC	
(75 test)	*SL*	*%*	*SL*	*%*	*SL*	*%*	*SL*	*%*	*SL*	*%*	*SL*	*%*
Dạng các đồ thị tổng quát (15 test)												
<	15	100.0	1	6.7	0	0.0	9	60.0	0	0.0	0	0.0
=	0	0.0	14	93.3	15	100.0	6	40.0	15	100.0	15	100.0
>	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0
Dạng các đồ thị đồng nhất (15 test)												
<	15	100.0	6	40.0	1	6.7	15	100.0	1	6.7	5	33.3
=	0	0.0	9	60.0	14	93.3	0	0.0	14	93.3	10	66.7
>	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0
Dạng các đồ thị có các cạnh được phân bố đều (15 test)												
<	14	93.3	2	13.3	0	0.0	11	73.3	0	0.0	0	0.0
=	1	6.7	13	86.7	15	100.0	4	26.7	15	100.0	15	100.0
>	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0
Dạng các đồ thị có các cạnh được phân bố không đều (15 test)												
<	13	86.7	0	0.0	0	0.0	8	53.3	0	0.0	0	0.0
=	2	13.3	15	100.0	15	100.0	7	46.7	15	100.0	15	100.0
>	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0
Dạng các đồ thị đầy đủ (15 test)												
<	10	66.7	1	6.7	0	0.0	3	20.0	0	0.0	0	0.0
=	5	33.3	14	93.3	15	100.0	12	80.0	15	100.0	15	100.0
>	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0
Tổng hợp cho các dạng đồ thị (75 test)												
<	67	89.3	10	13.3	1	1.3	46	61.3	1	1.3	5	6.7
=	8	10.7	65	86.7	74	98.7	29	38.7	74	98.7	70	93.3
>	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0

Về số lượng cây khung phải khảo sát

Đối với mỗi đồ thị trong 171 test đã thực nghiệm, ta ghi nhận số lượng cây khung mà các thuật toán Wong, SHCS, LS, MMAS, GA và ABC đã khảo sát khi thực hiện theo các tham số đã mô tả ở trên. Vì khuôn khổ có hạn của bài báo, bảng 3 dưới đây chỉ trích dẫn kết quả cho 5 bộ dữ liệu với các đồ thị đầy đủ. Bảng kết quả thực nghiệm đầy đủ và các file dữ liệu INPUT/OUTPUT được tải lên trên trang WEB[1].

Bảng 3. Số lượng cây khung được khảo sát theo các thuật toán

Test	Wong	SHCS	LS	MMAS	GA	ABC	BEE
COMP-71	30	11548560	11915630	6030	22495640	6688965	4761346
COMP-72	35	16378180	16536120	6035	36848100	9113498	5706112
COMP-73	40	18911220	19542990	6040	49266790	11182685	6336058
COMP-74	45	22173580	22546120	6045	65222190	13963337	7281010
COMP-75	50	26325180	25818520	6050	83452860	22056276	8226002

Kết quả thống kê trong bảng 3 cho thấy: Nếu tính trung bình cho một lần chạy thuật toán, số lượng cây khung mà thuật toán BEE-MRCST phải khảo sát trung bình là nhiều gấp hơn 16000 lần so với thuật toán Wong, và hơn 1.7 lần so với thuật toán ABC (chú ý là kết quả trong bảng 3 là thống kê của 30 lần chạy ABC và 10 lần chạy của BEE-MRCST).

4. KẾT LUẬN

Bài báo đề xuất thuật toán BEE-MRCST dựa trên thuật toán bầy ong để giải bài toán MRCST, trong đó đã đưa ra những cải tiến cụ thể về cách thức tạo quần thể ban đầu, cách thức tìm kiếm lân cận, cách thức phân bố các vùng để thực hiện việc tìm kiếm. Thuật toán

BEE-MRCST đã được cài đặt và thử nghiệm trên hai hệ thống test được sinh ngẫu nhiên với 171 bộ test. Kết quả thực nghiệm cho thấy thuật toán BEE-MRCST với những cải tiến đã nêu cho chất lượng lời giải cao hơn các thuật toán hiện biết. Việc tiếp tục phát triển thuật toán cho lời giải của bài toán MRCST với chất lượng cao hơn nữa là vấn đề cần giải quyết trong những nghiên cứu tiếp theo.

TÀI LIỆU THAM KHẢO

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

[2] V. Grout, Principles of cost minimization in wireless networks, *Journal of Heuristics* **11** (2) (2005) 115–133.

[3] B. A. Julstrom, 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 2005 (GECCO-2005)*, Hans-Georg Beyer et al.. Eds, vol. 1, ACM Press, New York, 2005 (585–590).

[4] 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 (454–461).

[5] Dervis Karaboga, Bahriye Basturk, *A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm*, Springer, 2007.

[6] D. Karaboga, B. Basturk, *On the Performance of Artificial Bee Colony (ABC) Algorithm,* Elsevier, 2007.

[7] Alok Singh, A new heuristic for the minimum routing cost spanning tree problem, *International Conference on Information Technology, IEEE,* Bhubaneswar, India, 2008 (9–13).

[8] Rui Campos, Manuel Ricardo, A fast algorithm for computing minimum routing cost spanning trees, *Computer Networks* **52** (17) (2008) 3229–3247.

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

[10] Xing-She Yang, *Nature-Inspired Meta-heuristic Algorithms*, Luniver Press, 2010 (53–62).

[11] Nguyen Minh Hieu, Phan Tan Quoc, Nguyen Duc Nghia, An approach of ant algorithm for solving minimum routing cost spanning tree problem, *SoICT 2011, ACM*, Ha Noi, Viet Nam, 2011 (5–10).

[12] Alok Singh, Shyam Sundar, An artificial bee colony algorithm for the minimum routing cost spanning tree problem, *Soft Computing - A Fusion of Foundations, Methodologies and Applications* **15** (12) (2011) Springer-Verlag, 2489–2499. (DOI: 10.1007/s00500-011-0711-6).

[13] Phan Tan Quoc, A Heuristic approach for solving the minimum routing cost spanning tree problem, *International Journal of Machine Learning and Computing (IJMLC), IACSIT* **2** (2012) 406–409.

[14] Phan Tan Quoc, A genetic approach for solving the minimum routing cost spanning tree problem, *International Journal of Machine Learning and Computing (IJMLC) IACSIT* **2** (2012) 410–414.

[15] D.S. Johnson, J.K. Lenstra, A.H.G. Rinnooy Kan, The complexity of the network design problem, *Networks* **8** (1978) 279–285 (John Wiley & Sons).

Ngày nhận bài 27 - 2 - 2013
Nhận lại sau sửa ngày 09 - 7 - 2013