Thuật toán tìm kiếm TABU giải bài toán cây khung với chi phí định tuyến nhỏ nhất

TABU Search Algorithm for Solving Minimum Routing Cost Spanning Tree Problem

Phan Tấn Quốc và Nguyễn Đức Nghĩa

***Abstract**: Minimum Routing Cost Spanning Tree (MRCST) is one of spanning tree optimization problems having several applications in network design. In general cases, the problem is proved as NP (Non-deterministic Polynomial) - hard. This paper proposes an algorithm for solving MRCST problem based on the schema of TABU search algorithm. Experimental results show that our proposal algorithm is better than Wong algorithm, many population-based meta-heuristics like Max-Min Ant System (MMAS), Genetic Algorithm (GA) and outperforms recently proposed Artificial Bee Colony algorithm (ABC) both in terms of solution quality as well as running time.*

I. GIỚI THIỆU BÀI TOÁN

Mục này phát biểu bài toán MRCST và khảo sát một số thuật toán giải bài toán MRCST được đề xuất trong những năm gần đây.

I.1. Phát biểu bài toán

Định nghĩa 1. *Chi phí định tuyến* [4]

Cho $G = (V,E,w)$ là 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; *với* $e \in E$. 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:

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

Bài toán MRCST yêu cầu tìm một cây khung với chi phí định tuyến nhỏ nhất trong số tất cả các cây khung của G. Bài toán này thuộc lớp NP–khó [4].

Xây dựng cây khung với chi phí định tuyến nhỏ nhất cũng 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 việc thiết kế mạng; chẳng hạn trong việc xây dựng các hệ thống mạng; đặc biệt là ở các mạng ngang hàng khi các nút có xác suất truyền tin và độ ưu tiên là như nhau (về xuất xứ và ứng dụng của bài toán có thể xem trong các tài liệu [4,7]).

Việc tính chi phí định tuyến của một cây khung có n đỉnh trực tiếp theo định nghĩa 1 đòi hỏi thời gian $O(n^2)$. Tuy nhiên, trên cơ sở khái niệm "tải định tuyến" ("routing load") ở định nghĩa 2; công trình [4] đã chỉ ra cách tính chi phí định tuyến của một cây khung với độ phức tạp tuyến tính.

Định nghĩa 2. *Tải định tuyến* [4]

Cho cây khung T với tập cạnh $E(T)$. Nếu loại khỏi T một cạnh e thì T sẽ được tách thành hai cây con gọi là T_1 và T_2 với 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 công thức (1) là tương đương với công thức (2) sau đây:

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

Bài toán MRCST khi n nhỏ thì có thể giải đúng bằng thuật toán nhánh cận [3]; kết quả giải đúng này được sử dụng để đánh giá các cách tiếp cận xấp xỉ, heuristic, meta-heuristics cho bài toán MRCST.

I.2. Khảo sát các thuật toán giải MRCST

a. Các thuật toán xấp xỉ

Thứ nhất là thuật toán Wong được đề xuất bởi Richard Wong vào năm 1980, thuật toán Wong có cận tỉ lệ 2 (nghĩa là chi phí của cây khung tìm được không vượt quá 2 lần chi phí của cây khung tối ưu) và 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) [1,4]; cây đường đi ngắn nhất có gốc tại đỉnh r là cây được hợp thành từ các đường đi ngắn nhất bắt đầu từ đỉnh r đến tất cả các đỉnh còn lại của đồ thị. Ý tưởng của thuật toán Wong là tìm các SPT xuất phát từ các đỉnh của đồ thị, sau đó chọn ra một SPT có chi phí thấp nhất – đây chính là kết quả của thuật toán Wong.

Thứ hai là thuật toán Add được đề xuất bởi Vic Grout vào năm 2005 [5,7], 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 được sử dụng đối với đồ thị đồng nhất và đồ thị gần đồng nhất (là loại đồ thị mà trọng số các cạnh khác nhau không đáng kể - trong bài báo này sẽ gọi chung 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; đây là thuật toán xấp xỉ 2 và có độ phức tạp là $O(m + n \log n)$ [7].

b. Các thuật toán heuristic

Ý tưởng chung của các thuật toán heuristic cho bài toán MRCST là bắt đầu từ một cây khung rồi từng bước cải thiện dần để thu được một cây khung tốt hơn; cây khung ban đầu có thể là cây đường đi ngắn nhất tìm được bằng thuật toán Wong hoặc là cây khung được sinh ngẫu nhiên. Ở đây chúng tôi trình bày sơ lược ba thuật toán heuristic khá hiệu quả trong những năm gần đây.

Thứ nhất là thuật toán thay thế cạnh *xóa cạnh trước rồi chèn cạnh sau* (viết tắt là REPRI) [15]: Từ cây khung T tìm được bằng thuật toán Wong; lần lượt duyệt từng cạnh e thuộc T, với mỗi e thì duyệt hết mọi cạnh thuộc $E–T$ để tìm cạnh e' sao cho $T–e+e$' là tốt nhất, nếu $T–e+e$' tốt hơn T thì thay T bằng $T–e+e$'; thuật toán dừng nếu trong một lần duyệt mọi cạnh thuộc T mà vẫn không tìm ra được cạnh nào thuộc $E–T$ để cải thiện T.

Thứ hai là thuật toán thay thế cạnh *chèn cạnh trước rồi xóa cạnh sau* (viết tắt là REPIR) [15]: Từ cây khung T tìm được bằng thuật toán Wong, lần lượt duyệt từng cạnh e thuộc tập $E–T$ và thay vào T, khi đó trong T sẽ xuất hiện chu trình; tìm cạnh xấu nhất trong chu trình này để loại đi và cập nhật T nếu thu được kết quả tốt hơn; thuật toán dừng nếu trong một lần duyệt hết mọi cạnh trong $E–T$ mà vẫn không tìm được cạnh nào trong T thay ra để cải thiện T.

Thứ ba là dựa vào sơ đồ thuật toán Local Srearch-LS [8]: Từ một giải pháp hiện tại được chọn là T, thuật toán LS sẽ tìm một giải pháp lân cận T' của T; nếu T' tốt hơn T thì T' sẽ trở thành giải pháp hiện tại. Quá trình này sẽ dừng khi thuật toán lặp đến một số lần xác định trước; nếu lời giải tốt nhất hiện tại không được cải thiện qua một số bước lặp xác định thì sẽ tiến hành xáo trộn lời giải hiện tại bằng cách thay thế một số cạnh của cây khung bằng một số cạnh ngẫu nhiên hợp lệ khác.

c. Các thuật toán meta-heuristic

Một số thuật toán meta-heuristic dạng quần thể đã được áp dụng cho bài toán MRCST như thuật toán di truyền [6,16], thuật toán bầy kiến [12], các thuật toán bầy ong [13,17]. Các thuật toán meta-heuristic dạng quần thể mất nhiều thời gian tìm kiếm hơn so với các meta-heuristic dạng đơn cá thể nhưng khả năng khai phá vùng không gian mới là có thể tốt hơn.

Bài báo này đề xuất một hướng tiếp cận mới; được phát triển dựa trên sơ đồ của thuật toán tìm kiếm TABU để giải bài toán MRCST. Thuật toán tìm kiếm TABU thuộc dạng meta-heuristic dạng đơn cá thể [2,9]; ngoài ý tưởng tìm kiếm TABU cơ bản, thuật toán còn đề xuất việc khai thác các chiến lược bổ sung

đặc thù của thuật toán tìm kiếm TABU vào bài toán MRCST.

II. THUẬT TOÁN TÌM KIẾM TABU

II.1. Ý tưởng chung của thuật toán tìm kiếm TABU

Từ một lời giải ban đầu, tìm kiếm TABU sẽ lặp đi lặp lại quá trình tìm kiếm nhằm cải thiện dần lời giải tốt nhất hiện có (ta sẽ gọi tắt là kỷ lục) của bài toán. Tại mỗi bước lặp, thuật toán sẽ duyệt trong một miền lân cận (cũng có thể là toàn bộ lân cận) của lời giải hiện tại để chọn ra lời giải tốt nhất, lời giải này sẽ thay thế cho lời giải hiện tại ở bước lặp kế tiếp. Mỗi lời giải trong lân cận của lời giải hiện tại được gọi là một lân cận (neighborhood) của lời giải hiện tại. Quá trình tác động lên lời giải hiện tại để biến nó thành một lân cận của lời giải hiện tại được gọi là một bước chuyển (move).

Điểm khác biệt căn bản của tìm kiếm TABU so với các thuật toán tìm kiếm địa phương khác chẳng hạn LS là tại mỗi bước lặp; để tránh việc duyệt trở lại những lời giải đã từng được khảo sát, tìm kiếm TABU sử dụng một danh sách để lưu trữ một số bước chuyển đã từng được sử dụng, gọi là danh sách TABU; danh sách này sẽ chứa một số bước chuyển vừa được thực hiện trong một số bước lặp ngay trước đó, các bước chuyển nằm trong danh sách TABU được gọi là các bước chuyển TABU. Các bước chuyển này sẽ bị cấm sử dụng lại chừng nào nó còn nằm trong danh sách TABU. Mỗi bước chuyển TABU sẽ nằm trong danh sách TABU trong khoảng thời gian *t* bước lặp, sau đó, bước chuyển này sẽ được loại khỏi danh sách TABU và nó có thể lại được sử dụng. Số *t* vừa nêu được gọi là giá trị TABU tenure của bước chuyển. Giá trị *t* có thể cố định cho tất cả các bước chuyển hoặc cũng có thể là một số ngẫu nhiên được chọn cho từng bước chuyển.

Hiệu quả của thuật toán tìm kiếm TABU phụ thuộc vào các yếu tố như: Cách thức tạo lời giải ban đầu, cách thức lựa chọn miền lân cận của lời giải hiện tại, tiêu chuẩn mong đợi cụ thể, các chiến lược bổ sung cụ thể, chiều dài danh sách TABU, giá trị TABU tenure [3,14].

II.2. Một số khái niệm liên quan tìm kiếm TABU

Tạo lời giải ban đầu

Lời giải ban đầu có thể được khởi tạo bằng một heuristic đơn giản hoặc phương pháp ngẫu nhiên.

Chọn miền lân cận

Tùy thuộc vào không gian tìm kiếm của bài toán mà có các cách thức chọn miền lân cận phù hợp. Thường thì hai cách thức sau được sử dụng: *Thứ nhất* là xét toàn bộ lân cận của lời giải hiện tại và từ đó chọn ra lời giải tốt nhất; cách này sẽ không hiệu quả khi số lượng lân cận của lời giải là đủ lớn. *Thứ hai* là xét một tập con lân cận ngẫu nhiên của lời giải hiện tại (trong bài báo này chúng tôi sử dụng cách thứ hai này).

Chọn lân cận

Nếu bước chuyển tốt nhất trong miền lân cận có thể cải thiện được kỷ lục thì tất nhiên bước chuyển đó sẽ được chọn, ngược lại thì bước chuyển đó sẽ được chọn với xác suất *p* nào đó, nếu sau phép thử xác suất mà bước chuyển này vẫn không được chọn thì sẽ chuyển sang thực hiện bước lặp tiếp theo với lời giải hiện tại được giữ nguyên.

Tiêu chuẩn mong đợi

Một vấn đề có thể xảy ra là một bước chuyển dù đang bị cấm nhưng nó lại có khả năng cải thiện kỷ lục; do đó để tránh bỏ sót các bước chuyển tốt này, thuật toán tìm kiếm TABU đưa ra khái niệm *tiêu chuẩn mong đợi* **(**aspiration criteria). Tiêu chuẩn mong đợi thường được áp dụng là: Nếu một bước chuyển TABU có thể cải thiện được kỷ lục thì bước chuyển này vẫn được chọn và nó sẽ được loại khỏi danh sách TABU.

Chiến lược bổ sung

Nhằm nâng cao chất lượng tìm kiếm, thuật toán tìm kiếm TABU đưa ra hai chiến lược tìm kiếm bổ sung là: chiến lược đa dạng hóa và chiến lược tăng cường hóa.

- Đa dạng hóa lời giải (diversifying)

Mục đích của việc đa dạng hóa là hướng đến những miền không gian tìm kiếm mới. Có thể thực

hiện việc đa dạng hóa lời giải bằng cách cho xáo trộn ngẫu nhiên một số phần tử của lời giải.

- Tăng cường hóa lời giải (intensifying)

Mục đích của việc tăng cường hóa là tập trung tìm kiếm sâu hơn ở những vùng không gian tìm kiếm có triển vọng chứa lời giải tốt. Có thể thực hiện việc tăng cường hóa lời giải bằng cách là: Nếu sau một số bước lập nhất định mà kỷ lục vẫn không được cải thiện; khi đó quá trình tìm kiếm TABU sẽ được khởi động lại với lời giải ban đầu chính là lời giải ứng với kỷ lục.

II.3. Sơ đồ thuật toán tìm kiếm Tabu cơ bản

- Khởi tạo lời giải ban đầu s_0; $s = s_0$; *sbest* = s_0;
- *tabulist* = ∅; *// danh sách TABU cho bằng rỗng*

while (*điều kiện dừng chưa thỏa*)

- *M* = ∅; *// M là tập các bước chuyển*
- *Với* $\forall s' \in N_S$ *(là miền lân cận của s):*
  - *p* = *move* (*s*, *s'*);
  - **if** (*p* ∉ *tabulist*) *M* = *M* ∪ *p*;
  - **if** (*p* ∈ *tabulist* và *p* thỏa tiêu chuẩn mong đợi)
    M = *M* ∪ *p*;
- *Chọn bước chuyển p tốt nhất trong tập M.*
  - *Tạo* s'_t *từ bước chuyển p của giải pháp s;*
  - $s = s'_t$;
  - if (s'_t *tốt hơn sbest*) *sbest* = s'_t;
- *Cập nhật tabulist;*
- *Thực hiện đa dạng hóa lời giải;*
- *Thực hiện tăng cường hóa lời giải;*

end while

return *sbest;*

III. THUẬT TOÁN TÌM KIẾM TABU GIẢI BÀI TOÁN MRCST

Trước hết ta sẽ trình bày một số vấn đề liên quan khi áp dụng thuật toán tìm kiếm TABU giải bài toán MRCST (ta gọi thuật toán này là TABU-MRCST).

III.1. Mã hóa cây khung

Trong thuật toán này chúng tôi sử dụng phương pháp mã hóa dạng cạnh để mã hóa cây khung: Mỗi cây khung được mã hóa thành một chuỗi gồm $n-1$ số nguyên, trong đó mỗi số nguyên là chỉ số cạnh của đồ thị có tham gia vào cây khung (các cạnh của đồ thị được đánh số từ 1 đến *m*).

III.2. Chi phí định tuyến của cây khung

Chi phí định tuyến của cây khung *T* được tính theo thuật toán trong công trình [4] dựa trên tải định tuyến; hàm tính chi phí của cây khung *T* đặt là ***routingcost*** (*T*).

III.3. Tạo lời giải ban đầu

Thuật toán của chúng tôi sử dụng thuật toán Wong để tạo lời giải ban đầu. Thuật toán Wong sử dụng khái niệm cây đường đi ngắn nhất xuất phát từ đỉnh *i* trên đồ thị *G*, ký hiệu là *SPT*(*G*,*i*) [4].

Đoạn mã giả để tạo lời giải ban đầu bởi thuật toán Wong được mô tả như sau:

```
initsolution(G,n,m)
// Khởi tạo lời giải ban đầu
{ Tbest=SPT(G,1)
for (i=2..n)
{
   T=SPT(G,i);
   if (routingcost(T)< routingcost(Tbest))
        Tbest=T ;
  }
return Tbest ;
}
```

III.4. Tìm tập cây khung lân cận của cây khung *T*

Gọi T^* là tập con ngẫu nhiên các cạnh của *T*, với mỗi cạnh $e \in T^*$, ta gọi *U* là một tập con của tập các cạnh có thể thay thế *e* khi loại *e* khỏi *T*. Xét mỗi cạnh $f \in U$, gọi *p* là bước chuyển khi thay *e* bởi *f*; nếu $p \notin$ *tabulist* thì đưa *p* vào *M*, ngược lại nếu $p \in$ *tabulist* và *p* thỏa tiêu chuẩn mong đợi - là *p* cải thiện được kỷ lục - thì đưa *p* vào *M*.

Đoạn mã giả để tìm tập các bước chuyển *M* sinh ra từ lời giải *T* được mô tả như sau:

```
neighborset(T,M)
//M là tập các bước chuyển sinh ra từ lời giải T
{
Đặt T* là tập con ngẫu nhiên của T;
```

for (mỗi cạnh $e \in T^*$)
{
Đặt U là tập con ngẫu nhiên của tập cạnh có thể thay thế *e;*
for (mỗi cạnh $f \in U$)
{
p=move(*e,f*);
if ($p \notin$ tabulist) $M = M \cup p$;
else
if (*p cải thiện được kỷ lục*)
$M=M \cup p$;
}
}
}

III.5. Tìm bước chuyển tốt nhất trong tập M

***bestmove*(*T,M*)**
// Từ *M*, tìm bước chuyển *p* tốt nhất
{
mincost = ∞;
for (mỗi bước chuyển $p_1 \in M$)
{
Đặt *T*' sinh bởi *T* qua p_1*;*
if (*routingcost* (*T*') < *mincost)*
{
mincost =routingcost(*T'*);
$p=p_1$;
}
}
return *p*;
}

III.6. Cập nhật danh sách TABU

updatetabulist(*p*)
// Cập nhật *tabutenure*, xóa cạnh không bị cấm và thêm *p* vào *tabulist*
{
for (mỗi bước chuyển $q \in$ *tabulist)*
{
-Giảm *tabutenure* của *q* (cạnh loại ra khỏi cây và cạnh thêm vào cây) nếu *tabutenure* > 0;
-Loại *q* nếu *tabutenure* của *q* đều bằng 0;
}
tabulist = *tabulist* ∪ *p*;
Gán giá trị *tabutenure* cho hai thành phần của *p* và giới hạn lại *tabulist* nếu danh sách vượt quá độ dài *tabulist;*
}

III.7. Chiến lược tìm kiếm bổ sung

Tăng cường hóa lời giải bằng cách khởi tạo lại danh sách các biến và gọi lại hàm TABU SEARCH với cây khung bắt đầu lại là *Tbest*.

intensify(*Tbest*);

Đa dạng hóa lời giải *T* bằng cách cho xáo trộn *k* cạnh của *T* để hy vọng quá trình tìm kiếm tiếp theo có kỷ lục tốt hơn.

diversify(T,k)
{
for i=1..*k*
Xóa cạnh *e* trong *T*, tìm ngẫu nhiên một cạnh *e*' hợp lệ để thay vào *T*;
}

Cuối cùng, thuật toán TABU-MRCST được mô tả như sau:

***TABU-MRCST* ()**
{
T_0=*initsolution*(*G,n,m*);
T=T_0;
Tbest = T_0;
while (*điều kiện dừng chưa thỏa*)
{
M=∅;
neighborset (*T,M*);
p=bestmove(*T,M*);
Đặt *T*' sinh từ *T* qua bước chuyển *p;*
if ((*routingcost*(*T'*) < *routingcost*(*T*)) hoặc (*T'* thỏa xác suất được chọn p_t))
{
T = *T'*;
updatetabulist (*p*);
if (*routingcost*(*T*) < *routingcost*(*Tbest*))
Tbest = *T*;
}

Sau k_1 lần lặp mà kỷ lục không được cải thiện thì thực hiện đa dạng hóa *diversify* (*Tbest*,*k*), sau k_2 lần đa dạng hóa thì thực hiện tăng cường *intensify*(*Tbest*);

}

}

Ta có thể đánh giá thời gian tính của một lần lặp của thuật toán TABU-MRCST như sau: Hàm *neighborset* có thời gian tính $O(n^2)$, hàm *bestmove* có thời gian tính $O(n)$, hàm *updatetabulist* có thời gian tính $O(1)$. Vậy, một lần lặp của thuật toán TABU-MRCST đòi hỏi thời gian $O(n^2)$.

IV. KẾT QUẢ THỰC NGHIỆM

IV.1. Hệ thống thực nghiệm

Thuật toán TABU-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.

Kết quả thực nghiệm TABU-MRCST được so sánh với Wong và với các thuật toán meta-heuristic dạng quần thể như MMAS [12], GA [16], ABC [13].

Trong thực nghiệm TABU-MRCST, chúng tôi đề nghị giá trị của các tham số là: *tabulist*=100, *tabutenure*=*n*/10, số lượng cạnh trong tập T^* là $n/4$, số lượng cạnh trong tập U là 5, xác suất p_t để chọn bước chuyển p khi p không cải thiện được kỷ lục là 0.75, số cạnh cần xáo trộn đa dạng hóa là k=4, và k_1=5**n*, k_2=4. Thuật toán TABU-MRCST cho thực hiện 10 lần chạy và mỗi lần cho thực hiện 2500 vòng lặp.

IV.2. Xây dựng các bộ dữ liệu thực nghiệm

Theo chúng tôi hiệu quả của thuật toán giải bài toán MRCST có thể phụ thuộc vào hai đặc trưng quan trọng của dữ liệu là: cấu trúc của đồ thị và trọng số trên các cạnh của đồ thị. Liên quan đến trọng số, chúng tôi đã sinh các đồ thị với 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, chúng tôi sinh các loại đồ thị: đồ thị đầy đủ (đồ thị dày) và đồ thị có các cạnh được phân bố đều/không đều giữa các đỉnh (đồ thị thưa), đồ thị tổng quát (ngẫu nhiên về trọng số và cấu trúc cạnh).

a. Đồ thị tổng quát

Trước hết xây dựng ngẫu nhiên một cây khung *T* gồm *n* đỉnh và $n-1$ cạnh, sau đó lần lượt thêm ngẫu nhiên $m-(n-1)$ cạnh hợp lệ khác; trọng số các cạnh của đồ thị là số nguyên ngẫu nhiên trong đoạn [1.. *max_weight*].

b. 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 Δ = random(*k*), *maxcost* = random(*max_weight)* + Δ + 1 ; khi đó *maxcost* + random(2*Δ+1) – Δ là trọng số cạnh; các trọng số này sẽ sai khác từ – (*k* – 1) đến (*k* – 1). Khi giảm *k* thì tính đồng nhất của đồ thị sẽ tăng.

c. 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 tương đương nhau hoặc chênh lệch là nhỏ, 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*) ≤ [2*m*/*n*], trọng số các cạnh của đồ thị là số nguyên ngẫu nhiên trong đoạn [1, *max_weight*].

d. 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 *k* cụm hình sao (các cạnh có chung đỉnh một đỉnh), mỗi cụm có *n*/*k*–2 cạnh và *n*–*k**(*n*/*k*–1) đỉnh còn lại sẽ tạo thành cụm hình sao cuối cùng, nối *k* 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.

e. Đồ thị đầy đủ

Đồ thị có *n* đỉnh có số cạnh là $m = (n-1)*n/2$, trọng số các cạnh của đồ thị là số nguyên ngẫu nhiên trong đoạn [1, *max_weight*].

Các bộ dữ liệu thực nghiệm được phát sinh ngẫu nhiên theo đề xuất từ các bài báo [8,12,17] với hai hệ thống test: Hệ thống test 1 có 96 test (trong đó có 18 test của các tác giả khác thường sử dụng ở trang WEB[2]) và hệ thống test 2 có 75 test. Hệ thống test 1 các đồ thị được sinh có số đỉnh trong khoảng [20, 100], số cạnh trong khoảng [50, 1200] và trọng số các cạnh trong khoảng [1, 250]. Hệ thống test 2 các đồ thị được giữ nguyên số đỉnh và số cạnh như đối với hệ thống test 1; riêng cấu trúc của đồ thị được sinh mới ngẫu nhiên, và trọng số các cạnh được cho ngẫu nhiên khoảng [1,100].

Các bộ dữ liệu (INPUT/OUTPUT) được lưu trữ chi tiết tại các trang WEB[1], WEB[2].

IV.3. Đánh giá kết quả thực nghiệm

a. Chi phí định tuyến

Kết quả thực nghiệm TABU-MRCST cho từng loại đồ thị ứng với từng thuật toán đã được trình bày chi tiết tại trang WEB[1] và được tổng hợp lại trong Bảng 1.

Nội dung của Bảng 1 cho biết số lượng (SL) bộ test cho kết quả tốt hơn (<) hoặc bằng nhau (=) hoặc kém hơn (>) khi so sánh thuật toán TABU-MRCST với các thuật toán Wong, MMAS, GA, ABC; đồng thời cũng cho biết phần trăm (%) tương ứng. Bảng 1 có so sánh riêng cho từng dạng đồ thị và tổng hợp cho tất cả các dạng đồ thị đã đề cập. Bảng 1 cho thấy với hầu hết các bộ test thì thuật toán TABU-MRCST cho kết quả tốt hơn hẳn các thuật toán Wong, MMAS, GA và ABC (với bản cài đặt của chúng tôi). Chi phí định tuyến trong các bảng được ghi bằng ½ giá trị tính theo công thức (2).

Và khi thực nghiệm các thuật toán trên với hệ thống test 2 ta thu được kết quả như ở Bảng 2.

Thuật toán TABU-MRCST có kết quả tốt hơn hẳn các thuật toán Add và Campos [15]. Khi thực nghiệm các thuật toán REPIR [15], REPRI [15], HCS (tìm kiếm leo đồi) [5] và LS [8] với các hệ thống test 1, 2 ta thu được kết quả như ở Bảng 3.

Bảng 1. So sánh TABU-MRCST với các thuật toán Wong, MMAS, GA, ABC trên hệ thống test 1

TABU	**Wong**		**MMAS**		**GA**		**ABC**	
(96 test)	*SL*	*%*	*SL*	*%*	*SL*	*%*	*SL*	*%*
Dạng các đồ thị tổng quát (36 test)								
<	34	94.4	21	58.3	1	2.8	1	2.8
=	2	5.6	15	41.7	35	97.2	35	97.2
>	0	0.0	0	0.0	0	0.0	0	0.0
Dạng các đồ thị đồng nhất (15 test)								
<	15	100.0	15	100.0	5	33.3	3	20.0
=	0	0.0	0	0.0	10	66.7	12	80.0
>	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	10	66.7	1	6.7	1	6.7
=	1	6.7	5	33.3	14	93.3	14	93.3
>	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	8	53.3	0	0.0	0	0.0
=	1	6.7	7	46.7	15	100.0	15	100.0
>	0	0.0	0	0.0	0	0.0	0	0.0
Dạng các đồ thị đầy đủ (15 test)								
<	15	100.0	10	66.7	0	0.0	0	0.0
=	0	0.0	5	33.3	15	100.0	15	100.0
>	0	0.0	0	0.0	0	0.0	0	0.0
Tổng hợp cho các dạng đồ thị (96 test)								
<	92	95.8	64	66.7	7	7.3	5	5.2
=	4	4.2	32	33.3	89	92.7	91	94.8
>	0	0.0	0	0.0	0	0.0	0	0.0

Bảng 2. So sánh TABU-MRCST với các thuật toán Wong, MMAS, GA, ABC trên hệ thống test 2.

TABU	**Wong**		**MMAS**		**GA**		**ABC**	
(75 test)	*SL*	*%*	*SL*	*%*	*SL*	*%*	*SL*	*%*
<	68	90.7	46	61.3	4	5.3	9	12.0
=	7	9.3	28	37.3	71	94.7	66	88.0
>	0	0.0	1	1.3	0	0.0	0	0.0

Bảng 3. So sánh TABU-MRCST với các thuật toán heuristic trên các hệ thống test 1 và 2.

TABU	REPIR		REPRI		HCS		LS	
(171 test)	*SL*	*%*	*SL*	*%*	*SL*	*%*	*SL*	*%*
<	34	19.9	41	24.0	35	20.5	10	5.8
=	137	80.1	130	76.0	136	79.5	161	94.2
>	0	0.0	0	0.0	0	0.0	0	0.0

(kết quả các thuật toán REPIR, REPRI, MMAS, GA được chúng tôi cập nhật tốt hơn kết quả đã đăng trong các bài báo [12,15,16]).

b. So sánh thời gian thực hiện các thuật toán

Thời gian trung bình thực hiện mỗi test theo các thuật toán Wong, MMAS, GA, ABC, TABU-MRCST lần lượt là 0.00 giây, 92.7 giây, 15.8 giây, 175.4 giây, 3.7 giây.

V. KẾT LUẬN

Bài báo đã đề xuất thuật toán TABU-MRCST được phát triển dựa trên sơ đồ thuật toán tìm kiếm TABU để giải bài toán MRCST. Thuật toán TABU-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 TABU-MRCST cho chất lượng lời giải cao hơn các thuật toán như Wong, MMAS, GA, ABC. Việc tiếp tục phát triển thuật toán cho lời giải của bài toán MRCST với chất lượng cao hơn là vấn đề mà chúng tôi sẽ giải quyết trong những nghiên cứu tiếp theo.

TÀI LIỆU THAM KHẢO

[1] R. WONG, "*Worst-case analysis of network design problem heuristics*", SIAM J. Algebra. Discr., 1:51–63, 1980.

[2] FRED GLOVER, MANUEL LAGUNA, "*Tabu Search"*, Kluwer Academic Publishers, 1998.

[3] M. FISCHETTI, G. LANCIA, AND P. SERAFINI, "*Exact algorithms for minimum routing cost trees*", Networks, vol.39, 2002 pp.161-173

[4] BANG YE WU, KUN-MAO CHAO,"*Spanning Trees and Optimization Problems*", Chapman & Hall/CRC, 2004.

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

[6] 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, pp. 585–590.

[7] RUI CAMPOS, MANUEL RICARDO, "*A fast Algorithm for Computing Minimum Routing Cost* Spanning Trees", Computer Networks, Volume 52, Issue 17, 2008, pp.3229-3247.

[8] ALOK SINGH, *"A New Heuristic for the Minimum Routing Cost Spanning Tree Problem "*, International Conference on Information Technology, IEEE, 2008.

[9] XING-SHE YANG, *"Nature-Inspired Meta-heuristic Algorithms*", LUNIVER, 2010.

[10] STEFFEN WOLF, PETER MERZ, "*Efficient Cycle Search for the Minimum Routing Cost Spanning Tree Problem*", Springer-Verlag Berlin Heidelberg,2010.

[11] JASON BROWNLEE, "*Clever Algorithms - Nature-Inspired Programming Recipes*", Swinburne University, Australia, 2011.

[12] NGUYEN DUC NGHIA, PHAN TAN QUOC, NGUYEN MINH HIEU, "*An Approach of Ant Algorithm for Solving Minimum Routing Cost Spanning Tree Problem*", SoICT 2011, ACM, pp.5-10.

[13] 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, Volume 15, Number 12, 2489-2499, DOI: 10.1007/s00500-011-0711-,6, Springer-Verlag 2011.

[14] NGUYỄN TẤN TRẦN MINH KHANG, TRIỆU TRÁNG KHÔN, ĐẶNG THỊ THANH NGUYÊN, TRẦN THỊ HUỆ NƯƠNG, "*Khảo sát một số giải thuật Tabu giải bài toán Xếp thời khóa biểu*", Tạp chí trường ĐH Sài Gòn, 2011.

[15] PHAN TAN QUOC, "*A Heuristic Approach for Solving the Minimum Routing Cost Spanning Tree Problem*", IJMLC 2012, pp.V2.406-409.

[16] PHAN TAN QUOC, "*A Genetic Approach for Solving the Minimum Routing Cost Spanning Tree Problem*", IJMLC 2012, pp.V2.410-414.

[17] PHAN TẤN QUỐC, NGUYỄN ĐỨC NGHĨA, *"Thuật toán bầy ong giải bài toán cây khung với chi phí định tuyến nhỏ nhất"*, ICTFIT 2012, "Tuyển tập công trình nghiên cứu Công nghệ thông tin & Truyền thông" của Nhà xuất bản Khoa học Kỹ thuật năm 2012, trang 73-81.

Nhận bài ngày: 22/03/2013

SƠ LƯỢC VỀ TÁC GIẢ

PHAN TẤN QUỐC

Sinh ngày 12 tháng 10 năm 1971.

Nhận bằng thạc sỹ Tin học, chuyên ngành Khoa học Máy tính tại Trường Đại học Khoa học Tự nhiên – Đại học Quốc gia TP.HCM năm 2002, NCS tiến sỹ tại trường Đại học Bách Khoa Hà Nội từ tháng 3/2010 - chuyên ngành Khoa học Máy tính.

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

NGUYỄN ĐỨC NGHĨA

Sinh ngày: 02 tháng 06 năm 1954.

Nhận bằng Tiến sỹ tại Trường Đại học Tổng hợp Quốc gia Bạch Nga. Hiện là Phó Giáo Sư, giảng viên cao cấp tại Bộ môn Khoa học Máy tính, Viện Công nghệ Thông tin và Truyền thông, Trường Đại học Bách khoa Hà Nội.

Lĩnh vực nghiên cứu: Tối ưu hóa tổ hợp và toàn cục, đồ thị, thiết kế và phân tích thuật toán, các thuật toán gần đúng, tính toán song song.

E-mail: nghiand@soict. hut.edu.vn