TRƯỜNG ĐẠI HỌC SÀI GÒN
TẠP CHÍ KHOA HỌC
ĐẠI HỌC SÀI GÒN
ISSN 1859 - 3208
Số 77 (06/2021)

SAIGON UNIVERSITY
SCIENTIFIC JOURNAL
OF SAIGON UNIVERSITY
No. 77 (06/2021)

Email: tcdhsg@sgu.edu.vn ; Website: http://sj.sgu.edu.vn/

ĐỀ XUẤT GIẢI THUẬT BEES GIẢI BÀI TOÁN CLIQUE LỚN NHẤT

Solving maximum Clique problem using Bees algorithm

ThS. Đỗ Minh Vũ[(1)], ThS. Mai Trương Hoàng Thông[(2)]

[(1)]Trường THPT chuyên Trần Hưng Đạo, Bình Thuận

[(2)]Công ty Hệ thống thông tin FPT

TÓM TẮT

Bài toán clique lớn nhất (Maximum clique problem) là bài toán tối ưu tổ hợp được ứng dụng trong nhiều lĩnh vực như mạng xã hội, tin sinh học, tài chính, lập lịch... và đã được chứng minh là bài toán thuộc lớp NP-Hard. Nghiên cứu này đề xuất giải thuật bầy ong giải bài toán clique lớn nhất dựa trên hệ thống dữ liệu thực nghiệm chuẩn DIMACS gồm 37 bộ dữ liệu thực nghiệm. Kết quả thực nghiệm của giải thuật đề xuất cho kết quả đạt từ 72% đến 100% so với lời giải kỷ lục hiện nay.

Từ khóa: *bài toán Clique lớn nhất, Giải thuật bầy ong, Giải thuật Heuristic, Giải thuật Metaheuristic, NP-Hard*

ABSTRACT

The Maximum clique problem is the combination optimization problem with practical application in many fields such as social networking, bioinformatics, finance, scheduling... and has been proved as a NP-Hard problem. This paper proposes the Bees algorithm to solve the maximum clique problem on 37 datasets standard of datatable DIMACS. The experimental results show that the proposed algorithm achieves the results from 72.0% to 100.0% compared to the current optimal results.

Keywords: *maximum Clique problem, Bees Algorithm, Heuristic Algorithm, Metaheuristic Algorithm, NP-Hard*

1. Giới thiệu

1.1. Một số định nghĩa

Mục này trình bày một số định nghĩa về bài toán clique lớn nhất

Định nghĩa 1. Clique

Cho đồ thị vô hướng liên thông $G=(V,E)$; trong đó V là tập đỉnh, E là tập cạnh. Tập đỉnh $C \subseteq V$ được gọi là một clique của đồ thị G nếu mọi cặp đỉnh (u,v) trong C đều là cạnh thuộc tập E.

Số lượng đỉnh (hay kích thước) của một clique C ký hiệu là $|C|$ [1−2].

Định nghĩa 2. Clique lớn nhất

C được gọi là một clique lớn nhất của đồ thị G nếu C là một clique và C có số đỉnh lớn nhất trong số các clique của G. Số lượng đỉnh của clique lớn nhất trong đồ thị G ký hiệu là $\omega(G)$ và gọi là chỉ số clique của đồ thị G [1−2].

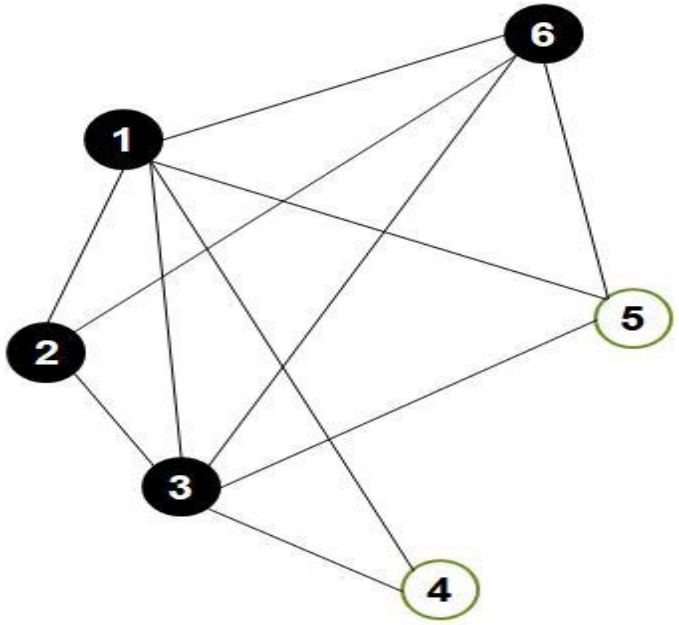
Định nghĩa 3. Bài toán clique lớn nhất

Cho đồ thị vô hướng liên thông $G=(V,E)$; trong đó V là tập đỉnh, E là tập cạnh. Bài toán clique lớn nhất là bài toán tìm một clique lớn nhất trong đồ thị đã cho.

Email: minhvu@thd.vn

Nếu $G = (V,E)$ là đồ thị không có trọng số thì $\omega(G) = max\{|C|: C$ là một clique của đồ thị $G\}$. Trong trường hợp tổng quát, bài toán clique lớn nhất đã được chứng minh thuộc lớp NP-khó [1−2].

Ví dụ: Cho đồ thị vô hướng G có 6 đỉnh và 11 cạnh như Hình 1.



Hình 1. Đồ thị vô hướng với Clique lớn nhất là {1,2,3,6}

1.2. Ứng dụng của bài toán clique lớn nhất

Clique lớn nhất là bài toán tối ưu tổ hợp có nhiều ứng dụng trong khoa học và kỹ thuật như mạng xã hội, máy học, mã hóa, thị giác máy tính, mạng viễn thông, lập lịch, thu hồi thông tin, tin sinh học, tài chính, hóa học,... [3−5].

1.3. Giải thuật bầy ong

Giải thuật bầy ong cơ bản được đề xuất bởi D. Pham và cộng sự (2005, 2006) [6], [7] và Yang (2010) [5], [8] như sau:

1: Khởi tạo quần thể ban đầu với n vùng; mỗi vùng chứa một cá thể ong;

2: Đánh giá độ thích nghi của quần thể;

3: **while** (điều kiện đừng chưa thỏa)

4: Trong n vùng ban đầu, chọn ngẫu nhiên p vùng ($p<n$) để thực hiện việc tìm kiếm lân cận và trong p vùng này chọn tiếp ra e vùng có độ thích nghi cao nhất.

5: Tuyển thêm *nep* ong để thực hiện việc tìm kiếm lân cận cho mỗi vùng quanh nó trong e vùng được chọn;

6: Tuyển thêm *nsp* ong để thực hiện việc tìm kiếm lân cận cho mỗi vùng quanh nó trong $p – e$ vùng ($nep>nsp$);

7: Mỗi ong trong $n – p$ vùng còn lại sẽ được thay thế bằng một ong ngẫu nhiên;

8: Đánh giá lại độ thích nghi cho các ong trong từng vùng;

9: Mỗi vùng sẽ chọn ra duy nhất một ong có độ thích nghi cao nhất để xây dựng quần thể ong ở thế hệ tiếp theo;

10: **end while**.

* Giải thuật Bees bao gồm một tập các tham số:

n: số ong do thám (tương ứng số

vùng); *p*: số vùng được chọn trong *n* vùng được thăm; *e*: số vùng tốt nhất trong p vùng được chọn; *nep*: số ong được cử đến mỗi vùng trong *e* vùng; *nsp*: số ong được cử đến *p* – *e* vùng; *np*: số ong được cử đến *n* – *p* vùng còn lại.

1.4. Vấn đề đặt ra cần giải quyết trong bài báo này

Clique lớn nhất (MCP - Maximum Clique Problem) là bài toán tối ưu tổ hợp. Để giải quyết bài toán này đã có rất nhiều tài liệu nghiên cứu từ nhiều lĩnh vực trong những năm qua để giải quyết bài toán trên. Đặc biệt, các hướng tiếp cận thuộc nhóm phương pháp heuristic và metaheuristic đang được quan tâm nhiều hiện nay. Trong nhóm các phương pháp metaheuristic thì Giải thuật Bees được đề xuất bởi nhóm tác giả [7−8] đang được ứng dụng nhiều trong các bài toán thuộc nhiều lĩnh vực khác nhau. Do vậy, Bài báo này **đề xuất giải thuật bầy ong giải bài toán clique lớn nhất** (Solving Maximum Clique Problem using Bee Algorithm).

2. Giải thuật bees giải bài toán clique lớn nhất

Mục này sẽ đề xuất giải thuật BEE-MCP để giải bài toán MCP. Trước hết, chúng tôi sẽ trình bày một số vấn đề liên quan khi áp dụng giải thuật vào MCP.

2.1. Mã hóa lời giải cho bài toán clique lớn nhất

Sử dụng phương pháp mã hóa dạng đỉnh để mã hóa lời giải bài toán. Lời giải sẽ là một dãy nhị phân n bít (mỗi bít ứng với mỗi đỉnh của đồ thị, nếu đỉnh i thuộc Clique thì bit thứ i bằng 1, ngược lại nếu đỉnh i không thuộc Clique thì bít thứ i bằng 0).

2.2. Tạo quần thể ban đầu

Các cá thể trong quần thể ban đầu cần được phân bố sao cho chúng có thể có mặt trong hầu hết các vùng của không gian tìm kiếm nhằm tạo được tính đa dạng của quần thể. Trong bài báo này, chúng tôi chọn cách sử dụng giải thuật heuristic HEU1 của nhóm tác giả Vũ Đình Hòa và các cộng sự [9] để khởi tạo các cá thể ban đầu cho quần thể. Đoạn mã giả tạo quần thể ban đầu *n* cá thể bởi cách thứ nhất như sau:

```
initializePopulation(G,n)
{
   population = ∅;
   while (i<=n)
        { C = HEU1(G,i); population
        = population ∪ C; }
}
```

2.3. Độ thích nghi của cá thể

Độ thích nghi của mỗi cá thể là kích thước của clique tương ứng với cá thể đó.

Đoạn mã giả để đánh giá độ thích nghi của các cá thể của quần thể được mô tả như sau:

```
caculateFitness()
{
    for (mỗi clique C thuộc quần thể)
    Tính kích thước của mỗi clique;
}
```

2.4. Phân chia vùng tìm kiếm

Trong *n* vùng ban đầu chọn ngẫu nhiên *p* vùng ($p < n$), tiếp theo sắp xếp *p* vùng này theo độ thích nghi giảm dần. Ta sử dụng hàm Quicksort để thực hiện việc sắp xếp. Khi đó trong *p* vùng này chọn ra *e* cá thể đầu tiên có độ thích nghi cao nhất sẽ thuộc *e*-vùng, *p* – *e* cá thể tiếp theo thuộc *pe*-vùng, *n* – *p* cá thể còn lại mà chưa được sắp xếp sẽ thuộc *np*-vùng. Mỗi vùng trong *e* vùng sẽ được tuyển thêm *nep* ong để tìm lân cận quanh nó. Mỗi vùng trong *p* - *e* vùng được chọn sẽ được tuyển thêm *nsp* ong để tìm lân cận (trong đó nep > nsp). Mỗi ong trong *n* - *p* vùng còn lại được thay thế bằng một ong ngẫu nhiên khác. Đánh giá độ thích nghi cho tất cả các ong ở mỗi

vùng. Mỗi vùng chọn ra duy nhất một ong có độ thích nghi cao nhất.

Đoạn mã giả chọn các cá thể cho mỗi loại vùng như sau:

***selectSites*()**
{
d=0;
populationtemp = ∅;
dachon[*population_count*]; //mảng khởi tạo các giá trị 0
while (*d*<*p*) {
k = random(population_count) +1;
if (dachon[k]=0) { populationtemp= populationtemp ∪ C_k; dachon[k]=1;}
d++ ;}

Sắp xếp các cá thể trong quần thể populationtemp và chọn các vùng như sau:

// chọn các cá thể đưa vào e-vùng
For (i=1; i<=e;i++) **e-vùng=e-vùng ∪ C_i** ;
// chọn các cá thể đưa vào pe-vùng
For (i=e+1; i<=p; i++) **pe-vùng=pe-vùng ∪ C_i** ;
// chọn các cá thể đưa vào np-vùng
For (i=1; i<=population_count; i++)
If (dachon[i] =0) **np-vùng=np-vùng ∪ C_i** ;
}

2.5. Tìm kiếm lân cận

Việc tìm kiếm lân cận *C'* cho một clique *C* được thực hiện như sau: Thêm một đỉnh *v* ngẫu nhiên vào clique hiện tại, nếu việc thêm đỉnh này vào clique hiện tại vẫn đảm bảo tập đỉnh mới là một clique thì tăng kích thước clique lên một, còn nếu sau một số lần thêm các đỉnh vào mà không thể làm tăng kích thước clique thì tiến hành loại bỏ một đỉnh mà tập các đỉnh còn lại của clique có nhiều đỉnh ứng viên nhất. Sau một số lần bớt đi mà cũng kết quả cũng không tốt hơn thì ta dừng quá trình tìm kiếm lân cận *C'* của *C*.

Ở mỗi bước lặp của giải thuật, các ong do thám sẽ được phân công vào ba loại vùng: loại *e*-vùng, loại *pe*-vùng và loại *np*-vùng. Mỗi vùng thuộc *e*-vùng sẽ tìm kiếm lân cận để sinh thêm *nep* clique. Mỗi vùng thuộc *pe*-vùng sẽ được tìm kiếm lân cận để sinh thêm *nsp* clique.

Mục đích của việc tìm kiếm lân cận là nhằm tăng cường sâu hơn khả năng tìm được kết quả tốt nhất ở những vùng có không gian lời giải tiềm năng. Sau đây là các đoạn mã giả cho việc tìm kiếm lân cận của mỗi loại vùng:

eSitesNeighSearch(C,e,nep) //Tìm kiếm lân cận để sinh thêm nep clique
{
for (mỗi clique *C* thuộc vùng **e-vùng**
for (i=1; i<= *nep* ; i++){
findNeightC(*C,C'*); // *C'* là lân cận của *C*
if (caculateFitness(C') > caculateFitness(C))
C=C';}
}

peSitesNeighSearch(C,pe,nsp) //Tìm kiếm lân cận để sinh thêm nsp clique
{
for (mỗi clique *C* thuộc vùng ***pe*-vùng)**
for (i=1; i<= *nsp* ; i++){
findNeightC(*C,C'*); // *C'* là lân cận của *C*
if (caculateFitness(C') > caculateFitness(C))
C=C';} }

findNeighSC(C,C') // Tìm clique lân cận {
while (thực hiện *k2* lần bỏ bớt đỉnh){
while (thực hiện *k1* lần thêm đỉnh) {

Thêm ngẫu nhiên một đỉnh vào clique;

Nếu có thể thêm được thì tăng kích thước clique lên một;}

Đánh giá lại độ thích nghi của clique hiện tại;

Xóa ngẫu nhiên một đỉnh bất kỳ;}

}

2.6. Tìm kiếm ngẫu nhiên

Việc tìm kiếm ngẫu nhiên diễn ra ở giai đoạn cuối của mỗi bước lặp khi $n - p$ ong được cử đi tìm kiếm ngẫu nhiên. Mỗi cá thể trong số *np*-vùng này sẽ được thay thế trực tiếp bằng một lân cận ngẫu nhiên mà bỏ qua điều kiện là cá thể ngẫu nhiên này có tốt hơn cá thể trước đó ở mỗi vùng đó. Việc thay thế mỗi cá thể trong *np-vùng* bằng một cá thể ngẫu nhiên khác nhằm mục đích khám phá các không gian tìm kiếm mới đồng thời giúp tránh việc tìm kiếm rơi vào tối ưu cục bộ. Cuối mỗi bước lặp của quá trình tìm kiếm lân cận và tìm kiếm ngẫu nhiên thì mỗi vùng chỉ chọn đúng một cá thể có độ thích nghi cao nhất để xây dựng quần thể ở thế hệ tiếp theo. Trong quá trình thực hiện giải thuật, ta cần lưu lại các cá thể tốt để tìm cá thể tốt nhất sau này.

Sau đây là đoạn mã giả thực hiện việc tìm kiếm ngẫu nhiên

npSitesRandSearch(*C,np*)
//Tìm Clique ngẫu nhiên
{
for (mỗi Clique thuộc mỗi vùng của ***np*-vùng)**
{ randSearch (*C'*); *C=C'*;}
}
randSearch(*C*) // Tạo Clique ngẫu nhiên {
Xét tập đỉnh *tempV=V*;
While (*tempV*>0)
Duyệt các đỉnh thuộc tập *V*;
Chọn ngẫu nhiên một đỉnh *v* không thuộc *U*;

Xóa tất cả các đỉnh *u* không kề với đỉnh *v* ra khỏi tập *V*

Cập nhật bậc của các đỉnh liên quan đến đỉnh vừa bị xóa;

end while;

}

2.7. Điều kiện dừng

Có nhiều điều kiện dừng đã được áp dụng trong các giải thuật metaheuristic như: Lời giải tốt nhất của bài toán được tìm thấy, số lần lặp của giải thuật đạt đến một giá trị định trước, giải thuật chạy hết một lượng thời gian. Trong giải thuật MCP, chúng tôi đề xuất điều kiện dừng là giải thuật kết thúc sau một số lần lặp định trước.

2.8. Tìm cá thể tốt nhất của quần thể

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

findBestIndi(*population*)
// Tìm cá thể tốt nhất của quần thể {
Cbest = 0;
for (mỗi clique *C* thuộc quần thể population**)**
if (calculateFitness(C)>Cbest) Cbest = calculateFitness(*C*)
return *Cbest;*}

Cuối cùng giải thuật bees giải bài toán clique lớn nhất (giải thuật ***BEE-MCP***) được mô tả như sau:

Đầu vào: Đồ thị vô hướng G(*V,E*).

Đầu ra: Kích thước Clique lớn nhất tìm được.

BEE-MCP() {
initiate_population(*G,n*);//tạo quần thể ban đầu
calculateFitness(*population*; //đánh giá độ thích nghi quần thể

while (*điều kiện lặp chưa thỏa*){
SelectSite(*population,n,p,e*); // Chọn cá thể cho mỗi vùng
eSitesNeighSearch(*C,e,nep*);//Tìm kiếm lân cận để sinh thêm *nep* clique
peSitesNeighSearch(*C,pe,nsp*);// Tìm kiếm lân cận để sinh thêm *nsp* clique
npSitesRandSearch(*C,np*); // Tìm clique ngẫu nhiên
PushIntoPopulation();// Chọn các cá thể cho quần thể ở thế hệ sau
dem++;}
findBestIndi(); // Tìm cá thể tốt nhất cho quần thể }

3. Thực nghiệm và đánh giá

3.1. Dữ liệu thực nghiệm

Giải thuật BEE – MCP được thực nghiệm trên hệ thống dữ liệu thực nghiệm chuẩn DIMACS có 37 bộ dữ liệu [10]. Đồ thị thưa là đồ thị có số cạnh thỏa mãn bất đẳng thức $m \geq 6n$. Nếu theo tiêu chuẩn này thì các đồ thị trong hệ thống DIMACS là các đồ thị dày vì luôn thỏa $m \geq 35n$.

3.2. Môi trường thực nghiệm

Giải thuật BEE-MCP được cài đặt bằng ngôn ngữ C++, sử dụng trình biên dịch C-Free 5.0 professional và chạy trên cấu hình máy tính Hệ điều hành Microsoft Windows 10 Pro, 64 bit, RAM 4GB, Intel(R) Core(TM) i3 CPU M380 @ 2.53GHz, 2.53 GHz.

3.3. Tham số thực nghiệm

Trong thực nghiệm giải thuật BEE-MCP, đề xuất các giá trị của tham số như sau: số cá thể trong quần thể là $n = 200$; số vùng được chọn để tìm kiếm lân cận $p = 2n/3$; $e = p/3$; số ong được cử đến mỗi vùng thuộc e-vùng là $nep = 7$; số ong được cử đến mỗi vùng thuộc pe-vùng là $nsp = 5$; số lần thực hiện thêm một đỉnh vào clique là $k1 = 5$; số lần thực hiện việc xóa bớt một đỉnh trong tìm kiếm lân cận là $k2 = 3$; Điều kiện dừng được lặp lại sau 30 lần.

3.4. Kết quả thực nghiệm

Kết quả thực nghiệm của giải thuật BEE-MCP và các giải thuật HEU1, HEU2, HEU1_IMPROVE, HEU2_IMPROVE [11] trên các bộ dữ liệu của hệ thống DIMACS. Với mỗi bộ dữ liệu, các chương trình được chạy 20 lần và kết quả tốt nhất của 20 lần chạy được ghi nhận làm kết quả của giải thuật.

Bảng 1. Kết quả thực nghiệm các giải thuật trên hệ thống DIMACS

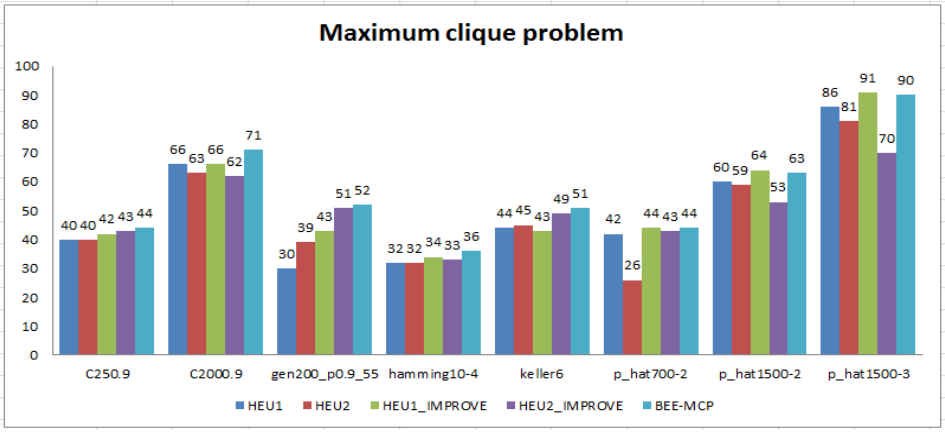
Instance	Nodes	Edges	Best Known	HEU1	HEU2	HEU1_ Improve	HEU2_ Improve	BEE-MCP
C125.9	125	6963	34*	32	31	34	34	34
C250.9	250	27984	44*	40	40	42	43	44
C500.9	500	112332	57	50	48	53	54	54
C1000.9	1000	450079	68	58	57	60	64	64
C2000.9	2000	1799532	80	66	63	66	62	71
DSJC1000_5	1000	499652	15*	9	14	13	14	14
DSJC500_5	500	125248	13*	11	12	13	13	13
C2000.5	2000	999836	16	12	14	14	16	15

Instance	Nodes	Edges	Best Known	HEU1	HEU2	HEU1_ Improve	HEU2_ Improve	BEE-MCP
C4000.5	4000	4000268	18	14	15	16	16	16
MANN_a27	378	70551	126*	125	125*	126	125	125
MANN_a45	1035	533115	345*	342	341*	343	342	343
MANN_a81	3321	5506380	1100	1096	1096	1096	1096	1096
brock200_2	200	9876	12*	8	10*	11	12	11
brock200_4	200	13089	17*	13	14*	16	17	16
brock400_2	400	59786	29*	21	20*	24	24	24
brock400_4	400	59765	33*	20	22*	24	33	24
brock800_2	800	208166	24*	15	18*	19	20	20
brock800_4	800	207643	26*	15	17*	19	19	20
gen200_p0.9_44	200	17910	44*	37	35	38	42	40
gen200_p0.9_55	200	17910	55*	39	39	43	51	52
gen400_p0.9_55	400	71820	55*	46	46	50	50	50
gen400_p0.9_65	400	71820	65*	45	43	48	50	51
gen400_p0.9_75	400	71820	75*	43	47	52	54	54
hamming10-4	1024	434176	40*	32	32	34	33	36
hamming8-4	256	20864	16*	16	16*	16	16	16
keller4	171	9435	11*	10	11*	11	11	11
keller5	776	225990	27*	22	22*	23	27	27
keller6	3361	4619898	59	44	45*	43	49	51
p_hat300-1	300	10933	8*	7	8*	8	8	8
p_hat300-2	300	21928	25*	21	24*	25	25	25
p_hat300-3	300	33390	36*	33	26*	35	35	35
p_hat700-1	700	60999	11*	7	9*	11	11	11
p_hat700-2	700	121728	44*	42	26*	44	43	44
p_hat700-3	700	183010	62	57	57	60	62	62
p_hat1500-1	1500	284923	12*	8	11	11	11	11
p_hat1500-2	1500	568960	65	60	59	64	53	63
p_hat1500-3	1500	847244	94	86	81	91	70	90

3.5. Độ phức tạp của giải thuật BEE-MCP

Thời gian tính của một vòng lặp của giải thuật BEE-MCP có thể được đánh giá như sau: Hàm sắp xếp quần thể có thời gian tính là $O(N \log N)$, Hàm xử lý cho e-vùng có thời gian tính là $O(N^2)$, Hàm xử lý cho p-vùng có thời gian tính là $O(N^2)$, Hàm xử lý cho np-vùng có thời gian tính là $O(N)$. Tổng cộng một vòng lặp của giải thuật BEE-MCP đòi hỏi thời gian tính là $O(N \log N + N^2) \sim O(N^2)$.

3.6. Minh họa kết quả đạt được bằng đồ thị



Hình 2. Minh họa kích thước clique lớn nhất của các giải thuật qua một số bộ dữ liệu của hệ thống dữ liệu thực nghiệm chuẩn DIMACS

3.7. Đánh giá kết quả thực nghiệm

Từ kết quả thực nghiệm trong bảng 1, có thể nhận thấy rằng với 37 bộ dữ liệu trong hệ thống DIMACS thì: giải thuật BEE-MCP cho chất lượng lời giải tốt hơn, bằng, kém hơn giải thuật HEU1 lần lượt là 91.9%, 8.1%, 0% tương ứng với 34/37 bộ dữ liệu có chất lượng lời giải tốt hơn, 3/37 bộ dữ liệu có chất lượng bằng. Giải thuật BEE-MCP cho chất lượng lời giải tốt hơn, bằng, kém hơn giải thuật HEU2 lần lượt là 81.1%, 18.9%, 0% tương ứng với 30/37 bộ dữ liệu có chất lượng lời giải tốt hơn, 7/37 bộ dữ liệu có chất lượng bằng. Giải thuật BEE-MCP cho chất lượng lời giải tốt hơn, bằng, kém hơn giải thuật HEU1_IMPROVE lần lượt là 43.2%, 48.6%, 8.1% tương ứng với 16/37 bộ dữ liệu có chất lượng lời giải tốt hơn, 18/37 bộ dữ liệu có chất lượng bằng và 3/37 bộ dữ liệu có chất lượng kém hơn. Giải thuật BEE-MCP cho chất lượng lời giải tốt hơn, bằng, kém hơn giải thuật HEU2_IMPROVE lần lượt là 29.7%, 56.8%, 13.5% tương ứng với 11/37 bộ dữ liệu có chất lượng lời giải tốt hơn, 21/37 bộ dữ liệu có chất lượng bằng và 5/37 bộ dữ liệu có chất lượng kém hơn. Giải thuật BEE-MCP cho kết quả đạt từ 72.0% đến 100.0% so với kết quả tối ưu.

4. Kết luận

Trong nghiên cứu này, chúng tôi đã đề xuất giải thuật Bees giải bài toán Clique lớn nhất, đồng thời chúng tôi đã cài đặt các giải thuật và thực nghiệm chúng trên hệ thống dữ liệu thực nghiệm chuẩn DIMACS. Kết quả thực nghiệm cho thấy rằng giải thuật đề xuất cho chất lượng lời giải tốt hơn các giải thuật heuristic

hiệu quả hiện biết, đặc biệt đối với các đồ thị có kích thước lớn. Các kết quả trong bài báo này cũng là thông tin có ích cho các nghiên cứu tiếp theo đối với bài toán clique lớn nhất cũng như đối với giải thuật bầy ong.

TÀI LIỆU THAM KHẢO

[1] A. Srivastava, A. Pillai và D. J. Gupta, "Maximum clique finder: MCF," *Information Technology & Electrical Engineering Journal,* tập 7, số 2, 2018.

[2] P. T. Quốc và N. Đ. 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,* tập 3, pp. 265−276, 2013.

[3] Đ. T. Phương, N. M. Tưởng và K. T. Hoài, "Bài toán clique lớn nhất - ứng dụng và những thách thức tính toán," *Tạp chí Khoa học & Công nghệ - Chuyên san Khoa học Tự nhiên - Kỹ thuật,* tập 102, số 2, pp. 13−17, 2013.

[4] R. A. Rossi, D. F. Gleich, A. H. Gebremedhin và M. M. A. Patwary, "Parallel maximum clique algorithms with applications to network analysis and storage," *SIAM Journal on Scientific Computing,* tập 37, số 5, pp. 589−616, 2015.

[5] X.-S. Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press, 2010.

[6] D. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim và M. Zaidi, "The Bees Algorithm - A Novel Tool for Complex Optimisation Problems," *Proceedings of IPROMS 2006 Conference,* pp. 454−461, 2006.

[7] D. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim và M. Zaidi, "The Bees Algorithm Technical Note," *Manufacturing Engineering Centre, Cardiff University, UK,* pp. 1-57, 2005.

[8] X.-S. Yang, Engineering Optimization: An Introduction with Metaheuristic Applications, Wiley, 2010.

[9] V. Đ. Hòa và Đ. T. Kiên, "Thuật toán song song giải bài toán xác định clique cực đại trên đồ thị," *Hội thảo Quốc gia: Một số vấn đề chọn lọc của Công nghệ thông tin và truyền thông,* pp. 426−442, 2009.

[10] F. Mascia, "DIMACS - The Maximum Clique problem," [Trực tuyến]. Available: http://iridia.ulb.ac.be/~fmascia/maximum_clique/DIMACS-benchmark. [Đã truy cập 23/04/2020].

[11] P. T. Quốc và H. T. C. Ái, "Cải tiến một số thuật toán heuristic giải bài toán Clique lớn nhất," *Hội nghị quốc gia lần thứ XII về nghiên cứu cơ bản và ứng dụng CNTT,* 2009.

[12] J. L. Walteros và A. Buchanan, "Why is maximum clique often easy in practice ?," *University at Buffalo,* pp. 1−28, 2018.

Ngày nhận bài: 26/5/2020 Biên tập xong: 15/3/2021 Duyệt đăng: 20/3/2021