CẢI TIẾN MỘT SỐ THUẬT TOÁN HEURISTIC GIẢI BÀI TOÁN CLIQUE LỚN NHẤT

Phan Tấn Quốc[1], Huỳnh Thị Châu Ái[2]

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

[2] Khoa Kỹ thuật công nghệ, Trường Đại học Văn Hiến

quocpt@sgu.edu.vn, aihtc@vhu.edu.vn

TÓM TẮT— *Clique lớn nhất (maximum clique problem) 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,... clique lớn nhất là bài toán thuộc lớp NP-hard. Hiện có nhiều hướng tiếp cận giải bài toán clique lớn nhất như các thuật toán tìm lời giải chính xác, các thuật toán heuristic, các thuật toán metaheuristic,.... Trong bài báo này, chúng tôi cải tiến hai thuật toán dạng heuristic giải bài toán clique lớn nhất; các thuật toán cải tiến của chúng tôi dựa trên hai thuật toán heuristic hiệu quả hiện biết. Chúng tôi đã cài đặt và thực nghiệm các thuật toán này trên 78 bộ dữ liệu trong hai hệ thống dữ liệu thực nghiệm chuẩn DIMACS và BHOSLIB. Kết quả thực nghiệm cho thấy rằng các thuật toán cải tiến của chúng tôi cho chất lượng lời giải phần lớn tốt hơn so với các thuật toán gốc. Các thuật toán cải tiến này và kết quả thực nghiệm tương ứng là thông tin hữu ích cho những nghiên cứu tiếp theo về bài toán clique lớn nhất.*

Từ khóa *— Maximum clique problem, community detection in social networks, heuristic algorithm, metaheuristic algorithm, NP-hard problem.*

I. GIỚI THIỆU

A. Một số định nghĩa

Mục này trình bày một số định nghĩa cơ sở cho bài toán clique lớn nhất [2].

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 còn gọi là kích thước) của một clique C ký hiệu là $|C|$. Nếu clique C chứa đỉnh v thì $|C| \leq deg(v) + 1$.

Định nghĩa 2. Clique cực đại

C được gọi là một clique cực đại nếu C là một clique và C không thuộc về bất kỳ một clique nào khác có số đỉnh nhiều hơn nó.

Định nghĩa 3. 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.

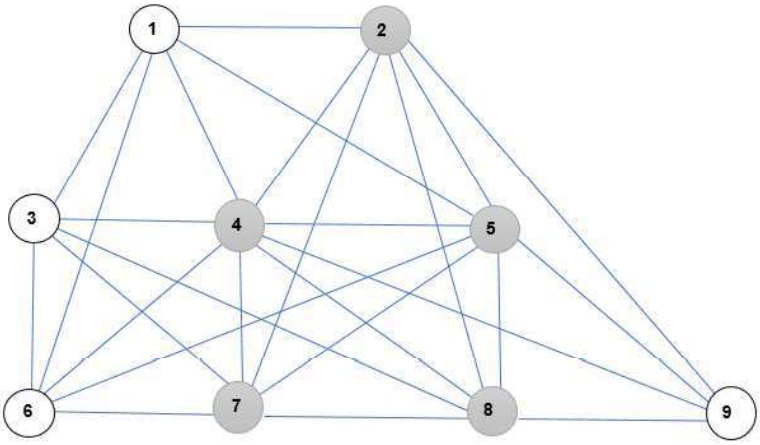
Một clique lớn nhất là một clique cực đại nhưng một clique cực đại chưa chắc đã là một clique lớn nhất. Rõ ràng một đồ thị có thể có nhiều clique có kích thước bằng clique lớn nhất.

Định nghĩa 4. 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 (Maximum Clique Problem-MCP) là bài toán tìm một clique lớn nhất trong đồ thị đã cho. Trong trường hợp tổng quát, bài toán clique lớn nhất đã được chứng minh thuộc lớp NP-hard.

Nếu $G = (V,E)$ là đồ thị không có trọng số thì $\omega(G) = max\{|C|: C$ là một clique của đồ thị $G\}$; nếu $G = (V,E)$ là đồ thị có trọng số (trên đỉnh) thì kích thước của clique C lớn nhất (clique weight) của G ký hiệu là $w(C)$ bằng tổng trọng số các đỉnh của C; khi đó $w(G) = max\{|C|: C$ là một clique của đồ thị $G\}$ [9][28]. Trong phạm vi bài báo này, chúng tôi chỉ giới hạn xem xét bài toán clique lớn nhất trong trường hợp đồ thị không có trọng số.

Ví dụ: Cho đồ thị vô hướng liên thông có 9 đỉnh và 26 cạnh như Hình 1; một clique lớn nhất tìm được ứng với đồ thị này là các đỉnh {2,4,5,7,8}.



Hình 1. Minh họa đồ thị và một clique lớn nhất với các đỉnh {2,4,5,7,8}

B. Ứng dụng của bài toán Clique lớn nhất

Bài toán MCP có thể được ứng dụng trong một số lĩnh vực khoa học và kỹ thuật; chẳng hạn như mạng xã hội, mạng viễn thông, tin sinh học, tài chính, hóa học, mã hóa, thị giác máy tính, máy học, lập lịch [7][14][26].

Ví dụ đối với mạng xã hội facebook: mỗi thành viên có thể kết nối với một hoặc nhiều thành viên khác (tiêu chí kết nối có thể là cùng sở thích chẳng hạn). Vấn đề đặt ra là cần tìm một cộng đồng có nhiều thành viên nhất (hoặc cần tìm một cộng đồng có *k*-thành viên) mà mỗi thành viên trong đó đều có kết nối (trực tiếp) đến với tất cả các thành viên còn lại. Trong các hệ thống mạng: Tìm kiếm thông tin về các cuộc gọi, tần suất cuộc gọi, tìm các nhóm người gọi có mối liên kết cao. Việc tìm kiếm và thu thập thông tin này là quan trọng vì có thể đánh giá được mẫu khách hàng và tối ưu hoá hoạt động của hệ thống. Trong tin sinh học: Thuật toán clique lớn nhất được sử dụng để tìm kiếm cộng đồng cho các mạng sinh học, bài toán so khớp cấu trúc phân tử hóa học…

C. Một số nghiên cứu liên quan bài toán MCP và vấn đề đặt ra cần giải quyết trong bài báo này

Bài toán MCP đã thu hút được sự quan tâm nghiên cứu liên tục, sâu rộng của các nhà khoa học trên thế giới trong hàng chục năm qua; đặc biệt là trong vài năm trở lại đây [1][9][10][11][14][16][17][20][24]. Ngoài những công trình khảo sát chi tiết về MCP [23], hiện đã có hàng loạt thuật toán giải bài toán MCP được đề xuất và có thể chia thành ba hướng tiếp cận chính sau đây:

Hướng thứ nhất là các giải thuật tìm lời giải đúng với các thuật toán nhánh cận [2][13][18][21], quy hoạch động [23][32]. Ưu điểm của hướng tiếp cận này là tìm được lời giải chính xác, nhược điểm của hướng tiếp cận này là chỉ giải được các bài toán ứng với đồ thị có kích thước nhỏ. Hướng tiếp cận này là một cơ sở quan trọng để đánh giá mức độ chính xác của các thuật toán giải gần đúng. Việc giải đúng bài toán MCP là một thách thức lớn trong lý thuyết tối ưu tổ hợp [1].

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 chính xác; thậm chí các thuật toán heuristic không chắc hiệu quả trên mọi loại dữ liệu đối với một bài toán cụ thể. Ưu điểm nổi bật của các thuật toán heuristic trong việc giải bài toán MCP là cho thời gian chạy nhanh. Một số thuật toán heuristic có thể được sử dụng làm điều kiện cắt nhánh trong các thuật toán nhánh cận giải bài toán MCP. Một số công trình tiêu biểu cho hướng tiếp cận này như thuật toán tham lam [5][12][14][25].

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. Một số công bố tiêu biểu giải bài toán MCP theo hướng này như thuật toán di truyền [6][23], thuật toán tabu search [23], thuật toán bầy ong [33], thuật toán tối ưu bầy kiến [19][27], thuật toán tìm kiếm lân cận biến đổi [22], thuật toán tìm kiếm địa phương [8][15][29][31]. Cho đến hiện tại, hướng tiếp cận metaheuristic giải bài toán MCP cho kết quả tốt nhất trong số các thuật toán giải gần đúng.

Do là bài toán thuộc lớp *NP-hard* nên ứng dụng của bài toán MCP cần được xem xét dưới góc độ của bài toán thiết kế hay dưới góc độ của bài toán thực thi: Nếu ở góc độ bài toán thiết kế thì cần ưu tiên hơn về chất lượng lời giải hơn; còn nếu ở góc độ bài toán thực thi thì cần ưu tiên về thời gian chạy hơn.

Ưu điểm của các thuật toán heuristic là cho thời gian chạy nhanh hơn các thuật toán metaheuristic; tuy nhiên chất lượng lời giải của các thuật toán heuristic thường kém chất lượng hơn so với metaheuristic. Bài báo này cải tiến hai thuật toán heuristic tốt hiện biết: *Thứ nhất* là heuristic của nhóm tác giả Vũ Đình Hòa và các cộng sự [30] và *thứ hai* là heuristic của nhóm tác giả Bharath Pattabiraman và các cộng sự [5].

II. CẢI TIẾN MỘT SỐ THUẬT TOÁN HEURISTIC GIẢI BÀI TOÁN CLIQUE LỚN NHẤT

Ý tưởng chung của các heuristic giải bài toán clique lớn nhất là bắt đầu từ một clique rỗng; sau đó lặp lại việc thêm dần các đỉnh để tạo thành clique lớn hơn đến khi nào không thể thêm đỉnh để tạo thành các clique lớn hơn nữa thì dừng. Các thuật toán heuristic dạng tham lam này dựa vào thông tin quan trọng nhất chính là bậc của các đỉnh ứng viên. Điểm mấu chốt của các cải tiến của chúng tôi là dựa vào yếu tố ngẫu nhiên của các lời giải tiềm năng và đồng thời cho chạy nhiều lần một bộ dữ liệu thực nghiệm để hy vọng tìm được lời giải có chất lượng tốt hơn.

A. Thuật toán heuristic 1

Trước hết chúng tôi trình bày nguyên bản thuật toán heuristic giải bài toán clique lớn nhất của nhóm tác giả Vũ Đình Hòa và các cộng sự [30] như sau:

```
1.  procedure MAXCLIQUEHEU1 (G=(V,E))   // viết tắt là HEU1
2.  begin
3.      Đặt U = ∅;
4.      while (V ≠ ∅) do
5.      begin
6.          Duyệt các đỉnh thuộc tập V; tìm đỉnh v ∉ U và v có bậc là lớn nhất;
7.          U=U ∪ {v};
8.          V=V \ {v};
9.          Xóa tất cả các đỉnh u không kề với đỉnh v ra khỏi tập V;
10.         Cập nhật bậc của các đỉnh liên quan đến các đỉnh u vừa bị xóa;
11.     end;
12.     Xuất kết quả U;
13. end;
```

HEU1 là thuật toán dạng tham lam dùng để tính cận dưới và thường được sử dụng trong các thuật toán nhánh cận giải bài toán MCP [30][32]. HEU1 là thuật toán điển hình được sử dụng nhiều trong các thuật toán dạng heuristic, metaheuristic giải bài toán MCP. Chúng tôi cải tiến thuật toán HEU1 ở các điểm sau đây: *Thứ nhất*, ở dòng 6 của thuật toán gốc, thay vì chọn đỉnh v thuộc tập V và v là đỉnh có bậc lớn nhất, thì thuật toán cải tiến (HEU1_improve) sẽ chọn v là đỉnh ngẫu nhiên chưa được chọn trước đó trong một tập Q chứa một số đỉnh có số bậc cao nhất của tập V. *Thứ hai*, cho thuật toán thực hiện nhiều lần và ghi nhận lời giải tốt nhất ở các lần chạy; biểu thị ở các dòng 2-4 và 16-23 của thuật toán HEU1_improve.

```
1.  procedure MAXCLIQUEHEU1_improve(G = (V,E)) begin        // viết tắt là HEU1_improve
2.  Bestsolution=0;
3.  T=∅;
4.      while (điều kiện dừng chưa thỏa) do
5.      begin
6.          Đặt U = ∅;
7.          while (V ≠ ∅) do
8.          begin
9.              Đặt tập Q={k đỉnh thuộc tập V có bậc lớn nhất};
10.             Chọn ngẫu nhiên một đỉnh v ∈ Q;
11.             U=U ∪ {v};
12.             V=V \ {v};
13.             Xóa tất cả những đỉnh u không kề với đỉnh v ra khỏi tập V;
14.             Cập nhật bậc của các đỉnh liên quan đến các đỉnh u vừa bị xóa;
15.         end;
16.         if |U|> Bestsolution then
17.             begin
18.                 Bestsolution=|U|;
19.                 Cập nhật T=U;
20.             end;
21.     end;
22.     Xuất kết quả Bestsolution và các đỉnh của clique T tương ứng có kích thước Bestsolution;
23. end;
```

B. Thuật toán heuristic 2

Tiếp theo chúng tôi trình bày nguyên bản thuật toán heuristic giải bài toán clique lớn nhất của nhóm tác giả Bharath Pattabiraman và các cộng sự [5] (chúng tôi chọn algorithm 2 của nhóm tác giả này).

```
1.  procedure MAXCLIQUEHEU2 (G=(V,E))        // viết tắt là HEU2
2.  for i: 1 to n do
3.  begin
4.      if d(v_i) ≥ max then     // max là kích thước của clique hiện tại
5.      begin
6.          U =∅;
```

```
7.          for v_j ∈ N(v_i) do
8.              if d(v_j) ≥ max then
9.                  U = U ∪ {v_j};
10.         while (U ≠ ∅) do
11.         begin
12.             Chọn một đỉnh v có bậc lớn nhất từ tập U;
13.             U=U \{v};
14.             Chỉ giữ lại đỉnh u ∈U sao cho u kề với đỉnh v và d(u)≥ max)
15.             max=max+1;
16.         end;
17.         if (|U| >max) then
18.             max=|U|;
19.     end;
20.     Xuất kết quả là max và các đỉnh của clique tương ứng có kích thước là max;
21. end;
```

Chúng tôi cải tiến thuật toán HEU2 trên ở các điểm sau: *Thứ nhất*, ở dòng 2 của thuật toán gốc, thay vì chọn lần lượt các đỉnh theo một thứ tự định sẵn, thuật toán cải tiến (HEU2_improve) sẽ chọn v_i là đỉnh ngẫu nhiên chưa được chọn trước đó. *Thứ hai*, ở dòng 12 của thuật toán gốc, thay vì chọn một đỉnh v có bậc lớn nhất từ tập U; thuật toán cải tiến sẽ chọn đỉnh v ngẫu nhiên từ tập Q chứa k đỉnh có bậc lớn nhất thuộc tập U. *Thứ ba*, cho thuật toán thực hiện nhiều lần và ghi nhận lời giải tốt nhất ở các lần chạy; biểu thị ở các dòng 2-4, 15 và 22-29 của thuật toán HEU2_improve.

```
1.  procedure MAXCLIQUEHEU2_improve(G = (V,E))        // viết tắt là HEU2_improve
2.  Bestsolution=0;
3.  T=∅;
4.      while (điều kiện dừng chưa thỏa) do
5.      begin
6.          Chọn v_i là đỉnh ngẫu nhiên thuộc tập V;
7.          if d(v_i) ≥ max then
8.              begin
9.                  U =∅;
10.                 for v_j ∈ N(v_i) do
11.                     if d(v_j) ≥ max then
12.                         U = U ∪ {v_j};
13.                 while (U ≠ ∅)
14.                 begin
15.                     Chọn ngẫu nhiên một đỉnh v ∈ Q với Q chứa k đỉnh có bậc lớn nhất thuộc tập U;
16.                     U=U \{v}
17.                     Chỉ giữ lại đỉnh u ∈U sao cho u kề với đỉnh v và d(u)≥ max)
18.                     max=max+1;
19.                 end;
20.                 if |U| >max then
21.                     max=|U|;
22.             end;
23.         if |max|> Bestsolution then
24.             begin
25.                 Bestsolution=max;
26.                 Cập nhật T=U; với U ứng với clique có kích thước bằng Bestsolution;
27.             end;
28.     Xuất kết quả Bestsolution và các đỉnh của clique T tương ứng có kích thước Bestsolution;
29. end;
```

III. THỰC NGHIỆM VÀ ĐÁNH GIÁ

Trong phần này chúng tôi mô tả chi tiết việc thực nghiệm các thuật toán gốc HEU1, HEU2 và các thuật toán cải tiến tương ứng HEU1_improve và HEU2_improve; đồng thời đưa ra một số đánh giá về các kết quả đạt được.

A. Dữ liệu thực nghiệm

Để thực nghiệm các thuật toán HEU1, HEU2, HEU1_improve, HEU2_improve chúng tôi sử dụng hai hệ thống dữ liệu thực nghiệm chuẩn gồm 78 bộ dữ liệu; trong đó hệ thống DIMACS [3] có 37 bộ và hệ thống BHOSLIB [4] có 41 bộ. Các đồ thị trong hệ thống DIMACS có số đỉnh trong phạm vi từ 125 đến 4000 và có số cạnh trong phạm vi từ 6963 đến 5506380; các đồ thị trong hệ thống BHOSLIB có số đỉnh trong phạm vis từ 450 đến 4000 và có số cạnh trong phạm vi từ 83151 đến 7425226.

Trong 78 bộ dữ liệu này, hệ thống DIMACS có 27 trên 37 bộ dữ liệu mà kỷ lục đã biết là lời giải tối ưu, hệ thống BHOSLIB có 41/41 bộ dữ liệu mà kỷ lục đã biết là lời giải tối ưu; khi đó kích thước clique ở cột Best known trong Bảng 1 và Bảng 2 có ghi kèm dấu*. Bài toán MCP là một thách thức lớn đối với các nhà khoa học; với hệ thống DIMACS thì kỷ lục mới và thời gian chạy là mục đích cần hướng tới, với hệ thống BHOSLIB thì thời gian chạy là mục đích cần hướng tới.

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 ở hai hệ thống dữ liệu thực nghiệm này đều là các đồ thị dày; trong đó hệ thống DIMACS luôn thỏa $m \geq 35n$ và hệ thống BHOSLIB luôn thỏa $m \geq 180n$; tức hệ thống BHOSLIB chứa các đồ thị dày hơn nhiều so với hệ thống DIMACS.

Chúng tôi thấy rằng trong 78 đồ thị của hai hệ thống DIMACS, BHOSLIB không có đồ thị nào có đỉnh có bậc nhỏ hơn kích thước của clique tìm được bằng thuật toán tham lam HEU1; do vậy nếu chúng ta rút gọn đồ thị theo cách sử dụng thuật toán tham lam để tìm ra một clique lớn nhất; sau đó duyệt qua các đỉnh của đồ thị và loại bỏ các đỉnh có bậc nhỏ hơn kích thước của clique lớn nhất là không khả thi; ít nhất cũng là với hai hệ thống DIMACS, BHOSLIB.

B. Môi trường thực nghiệm

Các thuật toán HEU1, HEU2, HEU1_improve, HEU2_improve được chúng tôi cài đặt bằng ngôn ngữ C++ sử dụng môi trường DEV C++ 5.9.2; được thực nghiệm trên máy tính Hệ điều hành Microsoft Windows 10 Pro, 64bit, RAM 4GB. Intel(R) Core(TM) i3-2330M CPU @ 2.20GHz, 2200 Mhz, 2 Core(s).

C. Kết quả thực nghiệm

Kết quả thực nghiệm của các thuật toán HEU1, HEU2, HEU1_improve, HEU2_improve trên các bộ dữ liệu DIMACS được ghi nhận ở Bảng 1; bảng này có cấu trúc như sau: Cột đầu tiên (Instance) ghi tên gọi các bộ dữ liệu trong hệ thống dữ liệu thực nghiệm chuẩn, các cột tiếp theo (NODES, EDGES, $DEGREE_{min}$, $DEGREE_{max}$, Best known) lần lượt ghi các thông tin về số đỉnh, số cạnh, số bậc nhỏ nhất, lớn nhất của các đỉnh của đồ thị, kích thước clique tốt nhất hiện biết của đồ thị (đây là kỷ lục tốt nhất được cập nhật trên các website cập nhật kỷ lục thực nghiệm bài toán MCP [3][4]; đến năm 2018, luận án tiến sĩ của Yi Zhou về bài toán MCP [32] chưa chỉ ra được một kỷ lục nào mới trên các bộ dữ liệu thuộc DIMACS mà chúng tôi đã liệt kê trong Bảng 1); bốn cột cuối ghi kích thước clique tìm được lần lượt ứng với các thuật toán HEU1, HEU2, HEU1_improve, HEU2_improve.

Chúng tôi đề xuất tham số cho hai thuật toán HEU1 improve, HEU2_improve như sau: *Điều kiện dừng* được cho lặp lại 30 lần; *Số lượng phần tử của tập Q* được cho *k* bằng 10% số đỉnh có bậc lớn nhất thuộc tập đỉnh ứng viên.

Do công trình [30] không thực nghiệm trên hai hệ thống dữ liệu thực nghiệm trên và công trình [5] chỉ thực nghiệm trên 17 trong số 37 bộ dữ liệu của hệ thống DIMACS (đó là các bộ dữ liệu mà kết quả ở cột HEU2 có đánh dấu*). Do đó chúng tôi cài đặt hai thuật toán HEU1, HEU2; và với các bộ test mà thuật toán HEU2 đã có kết quả thì chúng tôi ghi nhận như bài báo gốc [5].

Bảng 1. Kết quả thực nghiệm thuật toán trên các đồ thị trong hệ thống DIMACS

Instance	NODES	EDGES	$DEGREE_{min}$	$DEGREE_{max}$	Best known	HEU1	HEU2	HEU1_ improve	HEU2_ improve
C125.9	125	6963	102	119	34*	32	31	34	34
C250.9	250	27984	203	236	44*	40	40	42	43
C500.9	500	112332	431	468	57	50	48	53	54
C1000.9	1000	450079	868	925	68	58	57	60	64
C2000.9	2000	1799532	1751	1848	80	66	63	66	62
DSJC1000_5	1000	499652	447	551	15*	9	14	13	14
DSJC500_5	500	125248	220	286	13*	11	12	13	13
C2000.5	2000	999836	919	1074	16	12	14	14	16
C4000.5	4000	4000268	1895	2123	18	14	15	16	16
MANN_a27	378	70551	364	374	126*	125	125*	126	125
MANN_a45	1035	533115	1012	1031	345*	342	341*	343	342
MANN_a81	3321	5506380	3280	3317	1100	1096	1096	1096	1096
brock200_2	200	9876	78	114	12*	8	10*	11	12
brock200_4	200	13089	112	147	17*	13	14*	16	17
brock400_2	400	59786	274	328	29*	21	20*	24	24
brock400_4	400	59765	275	326	33*	20	22*	24	33
brock800_2	800	208166	472	566	24*	15	18*	19	20
brock800_4	800	207643	481	565	26*	15	17*	19	19
gen200_p0.9_44	200	17910	165	190	44*	37	35	38	42
gen200_p0.9_55	200	17910	164	190	55*	39	39	43	51
gen400_p0.9_55	400	71820	334	375	55*	46	46	50	50
gen400_p0.9_65	400	71820	333	378	65*	45	43	48	50
gen400_p0.9_75	400	71820	335	380	75*	43	47	52	54
hamming10-4	1024	434176	848	848	40*	32	32	34	33
hamming8-4	256	20864	163	163	16*	16	16*	16	16
keller4	171	9435	102	124	11*	10	11*	11	11
keller5	776	225990	560	638	27*	22	22*	23	27
keller6	3361	4619898	2690	2952	59	44	45*	43	49
p_hat300-1	300	10933	23	132	8*	7	8*	8	8

p_hat300-2	300	21928	59	229	25*	21	24*	25	25
p_hat300-3	300	33390	168	267	36*	33	26*	35	35
p_hat700-1	700	60999	75	286	11*	7	9*	11	11
p_hat700-2	700	121728	157	539	44*	42	26*	44	43
p_hat700-3	700	183010	408	627	62	57	57	60	62
p_hat1500-1	1500	284923	157	614	12*	8	11	11	11
p_hat1500-2	1500	568960	335	1153	65	60	59	64	53
p_hat1500-3	1500	847244	912	1330	94	86	81	91	70

Tương tự, kết quả thực nghiệm của các thuật toán HEU1, HEU2, HEU1_improve, HEU2_improve trên các bộ dữ liệu BHOSLIB được ghi nhận ở Bảng 2.

Bảng 2. Kết quả nghiệm thuật toán trên các đồ thị trong hệ thống BHOSLIB

Instance	NODES	EDGES	$DEGREE_{min}$	$DEGREE_{max}$	Best known	HEU1	HEU2	HEU1_ improve	HEU2_ improve
frb30-15-1	450	83198	327	407	30*	25	24	26	27
frb30-15-2	450	83151	333	404	30*	25	24	26	28
frb30-15-3	450	83216	327	400	30*	23	24	26	27
frb30-15-4	450	83194	339	401	30*	25	26	26	27
frb30-15-5	450	83231	321	403	30*	24	25	26	28
frb35-17-1	595	148859	462	544	35*	30	29	30	31
frb35-17-2	595	148868	460	541	35*	28	29	30	33
frb35-17-3	595	148784	429	549	35*	29	29	30	32
frb35-17-4	595	148873	444	560	35*	30	30	31	32
frb35-17-5	595	148572	460	550	35*	29	29	30	33
frb40-19-1	760	247106	581	703	40*	33	33	34	36
frb40-19-2	760	247157	588	702	40*	35	34	35	36
frb40-19-3	760	247325	600	702	40*	34	34	34	36
frb40-19-4	760	246815	595	692	40*	33	32	34	36
frb40-19-5	760	246801	585	691	40*	33	33	34	37
frb45-21-1	945	386854	756	876	45*	37	36	38	40
frb45-21-2	945	387416	753	870	45*	38	36	38	40
frb45-21-3	945	387795	739	872	45*	34	36	37	40
frb45-21-4	945	387491	732	875	45*	37	37	38	41
frb45-21-5	945	387461	764	874	45*	36	36	37	40
frb50-23-1	1150	580603	941	1065	50*	43	40	42	44
frb50-23-2	1150	579824	922	1074	50*	41	40	42	45
frb50-23-3	1150	579607	945	1078	50*	42	39	42	45
frb50-23-4	1150	580417	941	1071	50*	41	40	43	45
frb50-23-5	1150	580640	922	1080	50*	40	41	42	45
frb53-24-1	1272	714129	1039	1185	53*	43	41	43	48
frb53-24-2	1272	714067	1039	1182	53*	44	42	44	47
frb53-24-3	1272	714229	1037	1183	53*	44	43	44	47
frb53-24-4	1272	714048	1045	1189	53*	42	42	44	47
frb53-24-5	1272	714130	1065	1199	53*	42	42	44	47
frb56-25-1	1400	869624	1162	1311	56*	45	44	46	47
frb56-25-2	1400	869899	1162	1327	56*	46	45	46	46
frb56-25-3	1400	869921	1160	1301	56*	46	44	46	46
frb56-25-4	1400	869262	1158	1307	56*	46	43	47	46
frb56-25-5	1400	869699	1158	1311	56*	46	44	46	46
frb59-26-1	1534	1049256	1256	1439	59*	47	47	49	49
frb59-26-2	1534	1049648	1268	1433	59*	48	48	48	48
frb59-26-3	1534	1049729	1275	1455	59*	49	47	47	48
frb59-26-4	1534	1048800	1259	1435	59*	48	46	48	48
frb59-26-5	1534	1049829	1295	1452	59*	49	47	48	48
frb100-40	4000	7425226	3553	3864	100*	81	78	77	78

D. Đánh giá kết quả thực nghiệm

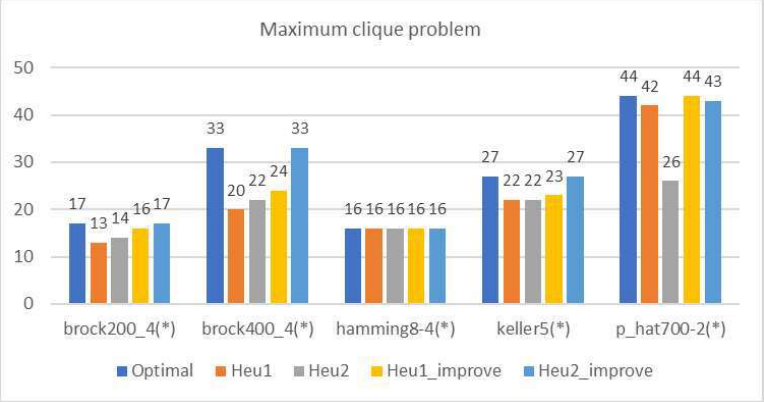
Các thuật toán heuristic được đánh giá dựa trên chất lượng lời giải (độ tốt của lời giải) và thời gian tính tương ứng để có lời giải đó thông qua việc thực nghiệm trên một số bộ dữ liệu mẫu. Mục này sẽ đưa ra một số phân tích đánh giá về chất lượng lời giải và thời gian chạy tương ứng.

Chất lượng lời giải

Với 37 bộ dữ liệu trong hệ thống DIMACS, thuật toán HEU1_improve cho chất lượng lời giải tốt hơn, bằng, kém hơn thuật toán HEU1 lần lượt là 89.2%, 8.1%, 2.7%; thuật toán HEU2_improve cho chất lượng lời giải tốt hơn, bằng, kém hơn thuật toán HEU2 lần lượt là 73.0%, 18.9%, 8.1%. Thuật toán HEU1 cho kết quả đạt từ 57.0% đến 100.0% so với kết quả tối ưu; thuật toán HEU1_improve cho kết quả đạt từ 69.0% đến 100.0% so với kết quả tối ưu; thuật toán HEU2 cho kết quả đạt từ 59.0% đến 100.0% so với kết quả tối ưu; thuật toán HEU2_improve cho kết quả đạt từ 72.0% đến 100.0% so với kết quả tối ưu.

Với 41 bộ dữ liệu trong hệ thống BHOSLIB, thuật toán HEU1_improve cho chất lượng lời giải tốt hơn, bằng, kém hơn thuật toán HEU1 lần lượt là 58.5%, 31.7%, 9.8%; thuật toán HEU2_improve cho chất lượng lời giải tốt hơn, bằng, kém hơn thuật toán HEU2 lần lượt là 97.6%, 2.4%, 0.0%. Thuật toán HEU1 cho kết quả đạt từ 75.6% đến 87.5% so với kết quả tối ưu; thuật toán HEU1_improve cho kết quả đạt từ 77.0% đến 88.6% so với kết quả tối ưu; thuật toán HEU2 cho kết quả đạt từ 76.8% đến 86.7% so với kết quả tối ưu; thuật toán HEU2_improve cho kết quả đạt từ 78.0% đến 94.3% so với kết quả tối ưu.

Chúng tôi minh họa chất lượng các thuật toán Quy hoạch động (OPTIMAL), HEU1, HEU2, HEU1_improve, HEU2_improve trên 5 bộ dữ liệu brock200_4, brock400_4, hamming8-4, keller5, p_hat700-2 như ở Hình 2.



Hình 2. Minh họa kích thước clique lớn nhất của các thuật toán qua một số bộ dữ liệu

Thời gian chạy của các thuật toán

Tiếp theo, chúng tôi ghi nhận thời gian chạy của các thuật toán Quy hoạch động (cột OPTIMAL), HEU1, HEU2, HEU1_improve, HEU2_improve qua 5 bộ dữ liệu như ở Bảng 3. Trong đó nếu cặp thuật toán và bộ dữ liệu nào mà sau 7200 giây chạy chương trình không tìm ra được kết quả thì ô tương ứng sẽ ghi ký hiệu '-'.

Bảng 3. Thời gian chạy chương trình ứng với các thuật toán trên mỗi bộ dữ liệu.

Instance	OPTIMAL	HEU1	HEU2	HEU1_improve	HEU2_improve
brock200_4	0.78	0.03	0.02	0.02	0.09
brock400_4	-	0.14	0.06	0.14	1.00
hamming8-4	0.01	0.06	0.02	0.08	0.17
keller5	-	0.55	0.31	0.44	4.78
p_hat700-2	-	0.27	0.25	0.33	2.09

Ghi chú: Thời gian tính bằng đơn vị giây, lấy đến 2 số lẻ thập phân.

Chúng tôi đã thực nghiệm thuật toán quy hoạch động trên 37 bộ dữ liệu trong hệ thống DIMACS với file nguồn chương trình được công bố tại [34]; nhận thấy chỉ 11 bộ dữ liệu cho kết quả sau 7200 giây thực hiện; đó là các bộ dữ liệu C125.9, keller4, brock200_2, **brock200_4**, gen200_p0.9_55, **hamming8-4**, p_hat300-1, p_hat300-2, DSJC500_5, p_hat700-1, p_hat1500-1.

Từ 4 cột cuối của Bảng 3 cho nhận xét rằng thuật toán HEU1_improve có thời gian chạy bằng từ 0.52 đến 1.24 lần so với thuật toán HEU1; thuật toán HEU2_improve có thời gian chạy bằng từ 6.25 đến 15.87 lần so với thuật toán HEU2. Thời gian chạy chương trình ngoài việc phụ thuộc vào độ phức tạp thời gian tính của thuật toán; còn phụ thuộc vào môi trường thực nghiệm, kỹ thuật lập trình, cấu trúc dữ liệu chọn cài đặt, các bộ tham số,…Tuy vậy, thời gian chạy của các thuật toán như ở Bảng 3 cũng cho chúng ta thông tin tham khảo cần thiết về các thuật toán này.

IV. KẾT LUẬN

Trong bài báo này chúng tôi đã cải tiến hai thuật toán heuristic để giải bài toán clique lớn nhất; các cải tiến này dựa trên hai thuật toán heuristic hiệu quả hiện biết. Chúng tôi đã cài đặt các thuật toán gốc, các thuật toán cải tiến và thực nghiệm chúng trên 78 bộ dữ liệu trong hệ thống dữ liệu thực nghiệm chuẩn. Kết quả thực nghiệm cho thấy rằng thuật toán cải tiến HEU1_improve cho chất lượng lời giải tốt hơn, bằng, kém hơn thuật toán gốc HEU1 lần lượt là 73.1%, 20.5%, 6.4%; thuật toán cải tiến HEU2_improve cho chất lượng lời giải tốt hơn, bằng, kém hơn thuật toán gốc HEU2 lần lượt là 85.9%, 10.3%, 3.8%. Các cải tiến này là hiệu quả đối với các đồ thị có kích thước lớn. Các kết quả đạt được trong bài báo này là thông tin hữu ích cho các nghiên cứu tiếp theo về bài toán clique lớn nhất.

V. TÀI LIỆU THAM KHẢO

[1] Alessio Conte and et al, "Finding all maximal cliques in very large social networks", ISSN: 2367-2005, pp.173-184, Bordeaux, France, 2016.
[2] Atul Srivastava and et al, "Maximum clique finder: MCF ", IJITEE, Volume 7, Issue 2,2018.
[3] Benchmark maximum_clique problem, http://iridia.ulb.ac.be/~fmascia/maximum_clique/DIMACS-benchmark,(last update, 26 Oct 2015).
[4] Benchmark maximum_clique problem, http://sites.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm,(last updated: Dec. 1, 2014).
[5] Bharath Pattabiraman and et al, "Fast algorithms for the maximum clique problem on massive graphs with applications to overlapping community detection", 2014.
[6] Bo Huang, "Finding maximum clique with a genetic algorithm, the pennsylvania state university", Master of Science, 2002.
[7] Đàm Thanh Phương, Ngô Mạnh Tưởng, Khoa Thu 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, Trường Đại học Thái Nguyên, ISSN: 1859-2171, Tập 102, Số 2, pp.13-17, 2013.
[8] Elena Marchiori, "Genetic, Iterated and multistart local search for the maximum clique Problem", University Amsterdam, The Netherlands, 2001.
[9] Emmanuel Hebrard, George Katsirelos, "Conflict directed clause learning for the maximum weighted clique problem", IJCAI, 2018.
[10] George Manoussakis, "New algorithms for cliques and related structures in k-degenerate graphs", Elsevier, 2018.
[11] Giannis Nikolentzos and et al, "K-clique-graphs for dense subgraph discovery", 2016.
[12] Golkar Mohammad Javad and et al, "Maximum clique problem solving using imperialist competitive algorithm and a greedy method for generating initial population", ISSN: 0976-2876, 2014.
[13] José Angel Riveaux Merino, "An exact algorithm for the maximum quasi-clique problem", Universidade Federal Fluminense, 2017.
[14] Jose L. Walteros, Austin Buchanan, "Why is maximum clique often easy in practice ?", University at Buffalo, pp.1-28,2018.
[15] Kengo Katayama, Akihiro Hamamoto, Hiroyuki Narihisa, "An effective local search for the maximum clique problem", Elsevier, 2005.
[16] Krishna Kumar Singh, Ajeet Kumar Pandey, "Survey of algorithms on maximum clique problem", IARJSET, Vol. 2, Issue 2, pp.15-20, 2015.
[17] Melisew Tefera Belachew, Nicolas Gillis, "Solving the maximum clique problem with symmetric rank-one nonnegative matrix approximation", Universit´e de Mons, Belgium, 2015.
[18] Mochamad Suyudi, Sukono and et al, "Branch and bound algorithm for finding the maximum clique problem", IEOM Society International, 2018.
[19] Mohammad Soleimani-Pouri and et al, "Finding a maximum clique using ant colony optimization and particle swarm optimization in social networks", IEEE, 2012.
[20] Muhammad Fayaz and et al, "Approximate maximum clique algorithm (amca): a clever technique for solving the maximum clique problem through near optimal algorithm for minimum vertex cover problem", IJCA, Australia, 2018.
[21] NGUYEN CANH Nam, "On globally solving the maximum weighted clique problem", Nafosted,Viet Nam, 2013.
[22] Pierre Hansen and et al, "Variable neighborhood search for the maximum clique", Elsevier, 2004.
[23] Qinghua Wua, Jin-Kao Hao, "A review on algorithms for maximum clique problems", Elsevier, 2014.
[24] Rachel Behar, Sara Cohen, "Finding all maximal connected s-cliques in social networks", ISBN 978-3-89318-078-3, 2018.
[25] Ryan A. Rossi and et al, "Fast maximum clique algorithms for large graphs", ACM, 2014.
[26] Ryan A. Rossi and et al, "Parallel maximum clique algorithms with applications to network analysis and storage", 2013.
[27] Serge Fenet, Christine Solnon, "Searching for maximum cliques with ant colony optimization", Springer-Verlag Berlin Heidelberg, pp.236-245,2003.
[28] Shaowei Cai, Jinkun Lin, "Fast solving maximum weight clique problem in massive graphs", IJCAI, 2016.
[29] Una Benlic, Jin-Kao Hao, "Breakout local search for maximum clique problems", Elsevier, 2012.
[30] Vũ Đình Hòa, Đỗ Trung 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.
[31] Wayne Pullan, Franco Mascia, Mauro Brunato, "Cooperating local search for the maximum clique problem", Springer, 2010.
[32] Yi Zhou, "Optimization algorithms for clique problems", grade de Docteur de l'Université d'Angers, pp.1-122, 2018.
[33] Yuquan Guo and et al, "Heuristic artificial bee colony algorithm for uncovering community in complex networks", Mathematical Problems in Engineering, pp.1-12,2017.
[34] https://github.com/bobogei81123/bcw_codebook/blob/master/codes/Graph/Maximum_Clique/Maximum_Clique.cpp

IMPROVED HEURISTIC ALGORITHMS FOR SOLVING MAXIMUM CLIQUE PROBLEM

Phan Tan Quoc, Huynh Thi Chau Ai

ABSTRACT— *The maximum clique problem (MCP) is known as NP-hard problem. It is also one of the most important combinatorial optimization problems with practical applications in numerous fields such as social network, machine learning, cryptography and coding, telecommunication network, scheduling, information retrieval, finance, signal transmission analysis, economics, biomedical engineering, chemistry, so on.s Many approaches have been developed to solve the MCP such as algorithms for finding exact solutions, heuristic algorithms, metaheuristic algorithms, so on. In this study, we propose two improved heuristic algorithms that are based on the effective well-known heuristic algorithms. We do experiments on the proposed approaches by using 78 datasets from two standard database, DIMACS and BHOSLIB. The experiment results claim that our approaches outperform baseline approaches. This proves that the proposed approach is effective and promising approach for solving the MCP.*