Một thuật toán tìm kiếm cục bộ hiệu quả giải bài toán phân công địa điểm thực tập cho sinh viên

Hoàng Hữu Việt(✉)
Viện Kỹ thuật và Công nghệ
Trường Đại học Vinh
Nghệ An, Việt Nam
Email: viethh@vinhuni.edu.vn

Nguyễn Thị Uyên(*)
Viện Kỹ thuật và Công nghệ
Trường Đại học Vinh
Nghệ An, Việt Nam
Email: uyennt@vinhuni.edu.vn

Cao Thanh Sơn
Viện Kỹ thuật và Công nghệ
Trường Đại học Vinh
Nghệ An, Việt Nam
Email: sonct@vinhuni.edu.vn

Lê Hồng Trang
Khoa Khoa học và Kỹ thuật máy tính
Trường Đại học Bách Khoa
Đại học Quốc gia TP.HCM, Việt Nam
Email: lhtrang@hcmut.edu.vn

***Tóm tắt nội dung*—Bài báo này giới thiệu một thuật toán hiệu quả để tìm một phép ghép hoàn chỉnh cho bài toán phân công địa điểm thực tập cho sinh viên. Chúng tôi mô hình bài toán theo dạng bài toán thỏa mãn ràng buộc và áp dụng phương pháp tìm kiếm cục bộ để giải quyết bài toán. Bằng cách chọn sinh viên theo thứ tự xoay vòng và loại bỏ cặp khối trội nhất, thuật toán sẽ loại bỏ tất cả các cặp khối tạo bởi sinh viên được chọn ở mỗi vòng lặp. Thực nghiệm chỉ ra rằng thuật toán đề xuất giải quyết bài toán kích thước lớn hiệu quả cả về thời gian thực hiện và chất lượng nghiệm.**

***Keywords*-bài toán ghép cặp; cặp khối; phép ghép ổn định; tìm kiếm cục bộ.**

I. MỞ ĐẦU

Bài toán phân công địa điểm thực tập cho sinh viên là bài toán thường gặp trong các trường đại học. Bài toán này thường được thực hiện bằng cách các khoa đào tạo phân công trực tiếp doanh nghiệp thực tập cho sinh viên hoặc sinh viên xin giấy xác nhận sẵn sàng tiếp nhận thực tập của doanh nghiệp và sau đó đăng ký đơn vị thực tập với khoa đào tạo. Khi mỗi doanh nghiệp bị hạn chế về số lượng nhận sinh viên thực tập thì hướng giải quyết là không hiệu quả vì khó đáp ứng được thỏa mãn đồng thời yêu cầu chọn sinh viên thực tập của doanh nghiệp và yêu cầu chọn doanh nghiệp thực tập của sinh viên.

Bài toán phân công địa điểm thực tập cho sinh viên là một bài toán ghép cặp được đề xuất bởi Gale và Shapley với tên "*College Admissions Problem*" [1]. Gần đây, bài toán này được xuất hiện với tên "*The Hospitals/Residents Problem*" và được gọi là bài toán HR [2], [3]. Trong ngữ cảnh của bài toán nghiên cứu, các sinh viên được xem là "*residents*" và các doanh nghiệp thực tập được xem là "*hospitals*" và do vậy, chúng tôi sẽ sử dụng lại các ký hiệu của "*residents*" và "*hospitals*" trong bài toán HR để biểu diễn các ký hiệu của "*sinh viên - students*" và "*doanh nghiệp - enterprises*". Cụ thể, bài toán HR được định nghĩa gồm một tập khác rỗng $\mathcal{R} = \{r_1, r_2, \cdots, r_n\}$ các *sinh viên*, một tập khác rỗng $\mathcal{H} = \{h_1, h_2, \cdots, h_m\}$ các *doanh nghiệp*, trong đó mỗi $r_i \in \mathcal{R}$ xếp hạng một tập con của $\mathcal{H}$ theo một thứ tự ưu tiên hay "*thích*" nghiêm ngặt (strict order of preference), mỗi $h_j \in \mathcal{H}$ xếp hạng một tập con của $\mathcal{R}$ theo một thứ tự "*thích*" nghiêm ngặt và mỗi $h_j \in \mathcal{H}$ có một số $c_j \in \mathbb{Z}^+$ chỉ ra số sinh viên tối đa mà doanh nghiệp h_j có thể nhận thực tập. Một cặp $(r_i, h_j) \in \mathcal{R} \times \mathcal{H}$ được gọi là chấp nhận (*acceptable*) lẫn nhau nếu h_j xuất hiện trong danh sách xếp hạng của r_i và ngược lại. Một phép ghép M (*matching*) là một tập các cặp $(r_i, h_j) \in \mathcal{R} \times \mathcal{H}$ thỏa mãn: (*i*) r_i và h_j chấp nhận lẫn nhau; (*ii*) r_i được gán tối đa một h_j; và (*iii*) h_j được gán tối đa c_j sinh viên r_i. Một cặp $(r_i, h_j) \in M$ được ký hiệu là $M(r_i) = h_j$ và $M(h_j)$ là tập tất cả các sinh viên được gán cho h_j trong M. Một doanh nghiệp $h_j \in \mathcal{H}$ được gọi là chưa đủ (*under-subscribed*), đủ (*full*) hoặc vượt quá (*over-subscribed*) số lượng sinh viên thực tập nếu $|M(h_j)| < c_j$, $|M(h_j)| = c_j$ hoặc $|M(h_j)| > c_j$, tương ứng. Một cặp $(r_i, h_j) \in \mathcal{R} \times \mathcal{H}$ là một cặp khối (*blocking pair*) cho phép ghép M nếu: (*i*) r_i và h_j chấp nhận lẫn nhau; (*ii*) r_i hoặc không được gán hoặc thích h_j hơn $M(r_i)$ theo nghĩa r_i xếp hạng

(*)Nghiên cứu sinh của Học viện Khoa học và Công nghệ, Viện Hàn lâm Khoa học và Công nghệ Việt Nam.

h_j ưu tiên hơn $M(r_i)$; và (*iii*) h_j hoặc là chưa đủ số lượng sinh viên được gán hoặc là xếp hạng r_i ưu tiên hơn sinh viên mà h_j xếp hạng thấp nhất trong $M(h_j)$. Một phép ghép M được gọi là ổn định (*stable matching*) nếu M không có bất kỳ cặp khối nào, ngược lại M được gọi là không ổn định (*unstable matching*). Mục tiêu giải bài toán HR là tìm một phép ghép ổn định.

Bài toán HR yêu cầu các thực thể xếp hạng các ứng viên (*applicants*) theo một thứ tự "*thích*" nghiêm ngặt, do đó khó ứng dụng được cho các bài toán thực tế. Vì vậy, một số mở rộng của bài toán HR đã được đề nghị [3], [4], [5], trong đó bài toán HR không yêu cầu các ứng viên xếp hạng nghiêm ngặt (*Hospitals/Residents problem with Ties, gọi tắt là* HRT) [2], [3] được chú ý nhiều nhất. Theo đó, các định nghĩa của phép ghép ổn định đã được xem xét gồm ổn định yếu (*weak stability*), ổn định mạnh (*strong stability*) và siêu ổn định (*super-stability*). Cho một thể hiện (*instance*) I của HRT, được biết rằng các phép ghép ổn định yếu có nhiều kích thước khác nhau và để cho mọi sinh viên được phân công một địa điểm thực tập, hiển nhiên cần phải tìm một phép ghép không chỉ ổn định mà còn có nhiều nhất số sinh viên được phân công. Đây là bài toán tìm phép ghép ổn định yếu với kích thước lớn nhất, được gọi là MAX-HRT, và được chứng minh là NP-khó (*NP-hard*) ngay cả khi mỗi $h_j \in \mathcal{H}$ có $c_j = 1$ [6], [7]. Manlove và cộng sự [8] chứng minh rằng kích thước lớn nhất của một phép ghép ổn định yếu luôn lớn hơn 2 lần kích thước bé nhất của phép ghép ổn định yếu trong mọi thể hiện của bài toán HRT. Kwanashie và cộng sự [9] đề xuất hướng tiếp cận lập trình nguyên (*integer programming*) để giải bài toán MAX-HRT. Munera và cộng sự [10] chuyển bài toán HRT thành bài toán SMTI và áp dụng giải thuật tìm kiếm thích nghi (*adaptive search*) [11] để giải bài toán MAX-HRT.

Trong bài báo này chúng tôi đề xuất một thuật toán MIN-BPS với ý tưởng dựa trên thuật toán xung đột tối thiểu (*min-conflicts algorithm*) cho bài toán thỏa mãn ràng buộc [12] để giải bài toán MAX-HRT. Cho một thể hiện I của bài toán HRT, thuật toán MIN-BPS tìm nghiệm của I từ một phép ghép ngẫu nhiên M. Tại mỗi bước lặp, thuật toán chọn một sinh viên r_i theo thứ tự xoay vòng và tìm một doanh nghiệp h_j để tạo thành một cặp khối (r_i, h_j) trội nhất (*undominated blocking pair*), sau đó loại bỏ (r_i, h_j) ra khỏi M. Bằng cách loại bỏ cặp khối trội nhất tạo bởi $r_i \in M$, thuật toán sẽ loại bỏ tất cả các cặp khối tạo bởi $r_i \in M$ để được một phép ghép mới. Thuật toán lặp lại với phép ghép cho đến khi đạt được một phép ghép hoàn chỉnh, tức là phép ghép ổn định mà mọi sinh viên đều được phân công doanh nghiệp thực tập, hoặc một số tối đa bước lặp. Thực nghiệm chỉ ra rằng thuật toán MIN-BPS giải quyết bài toán MAX-HRT kích thước lớn hiệu quả cả về thời gian thực hiện và chất lượng nghiệm.

Phần còn lại của bài báo được tổ chức như sau: Phần 2 giới thiệu các định nghĩa liên quan đến bài toán, Phần 3 mô tả thuật toán MIN-BPS, Phần 4 thảo luận các kết quả thực nghiệm và Phần 5 đưa ra các kết luận của bài báo.

II. CÁC ĐỊNH NGHĨA

Cho một thể hiện I và một phép ghép M của bài toán HRT, một số định nghĩa [2], [3], [13] được sử dụng trong bài báo gồm:

Định nghĩa 1 (*cặp khối*). Một cặp $(r_i, h_j) \in \mathcal{R} \times \mathcal{H}$ là một cặp khối cho phép ghép M nếu: (*i*) r_i và h_j chấp nhận lẫn nhau; (*ii*) r_i hoặc là không được gán hoặc là "*thích*" h_j hơn $M(r_i)$ theo nghĩa xếp hạng h_j ưu tiên hơn $M(r_i)$; và (*iii*) h_j hoặc là chưa đủ số sinh viên hoặc là xếp hạng r_i ưu tiên hơn sinh viên mà h_j xếp hạng thấp nhất trong $M(h_j)$.

Định nghĩa 2 (*phép ghép ổn định yếu*). Một phép ghép M là ổn định yếu nếu M không có cặp khối, ngược lại, M được gọi là không ổn định.

Định nghĩa 3 (*kích thước phép ghép*). Kích thước của một phép ghép ổn định yếu M là số sinh viên được phân công cho các doanh nghiệp trong M. Nếu kích thước của M là n thì M được gọi là một phép ghép hoàn chỉnh (*perfect matching*).

Định nghĩa 4 (*cặp khối vượt trội*). Một cặp khối (r_i, h_j) vượt trội (*dominate*) một cặp khối (r_i, h_k) nếu r_i thích h_j hơn h_k.

Định nghĩa 5 (*cặp khối trội nhất*). Một cặp khối (r_i, h_j) gọi là cặp khối trội nhất (*undominated blocking pair*) nếu không có cặp khối nào vượt trội cặp khối (r_i, h_j).

Khái niệm cặp khối vượt trội và cặp khối trội nhất được đề xuất trong [13] và sau đó được áp dụng để giải bài toán SMTI [13], [14]. Khi loại bỏ một cặp khối trội nhất (r_i, h_j) cho một phép ghép M, kết quả trả về một phép ghép mà tất cả các cặp khối tạo bởi r_i sẽ bị loại bỏ.

Bảng I mô tả một thể hiện của HRT gồm 8 sinh viên và 5 doanh nghiệp. Trong danh sách xếp hạng của các sinh viên, ví dụ, ký hiệu r_2: h_1 (h_4 h_5) h_3 có nghĩa là r_2 xếp hạng h_1 ưu tiên hơn h_4 và h_5, nhưng xếp hạng h_4 và h_5 là như nhau. Trong danh sách xếp hạng ưu tiên của các doanh nghiệp, ví dụ, h_2: (3) có nghĩa là $c_2 = 3$. Phép ghép M = {(2,1), (3,1), (4,2), (5,3), (6,2), (7,5), (8,4)} là ổn định yếu, trong đó r_1 không được phân công thực tập và h_2 chưa đủ số lượng sinh viên thực

Bảng I: Xếp hạng của sinh viên và doanh nghiệp

Danh sách xếp hạng ưu tiên của sinh viên	Danh sách xếp hạng ưu tiên của các doanh nghiệp
r_1: h_1 h_3 h_2	h_1: (2): r_3 (r_7 r_5 r_2) r_4 r_6 r_1
r_2: h_1 (h_5 h_4) h_3	h_2: (3): r_5 r_6 (r_3 r_4) r_1
r_3: h_1 h_5 h_2	h_3: (1): (r_5 r_2) r_6 r_1 r_7
r_4: h_1 (h_2 h_4)	h_4: (1): r_8 r_2 r_4 r_7
r_5: h_3 h_1 h_2	h_5: (1): r_3 (r_7 r_6 r_8) r_2
r_6: (h_3 h_2) h_1 h_5	
r_7: h_3 h_4 h_5 h_1	
r_8: h_5 h_4	

tập. Tuy nhiên, phép ghép M = {(1,2), (2,1), (3,1), (4,2), (5,3), (6,2), (7,5), (8,4)} là phép ghép hoàn chỉnh, nghĩa là mọi sinh viên đều được phân công địa điểm thực tập.

III. THUẬT TOÁN ĐỀ XUẤT

Trong phần này chúng tôi đề xuất thuật toán MIN-BPS để giải bài toán MAX-HRT. Ý tưởng chính của thuật toán là dựa trên thuật toán xung đột tối thiểu để giải bài toán thỏa mãn ràng buộc [12], trong đó ở mỗi bước lặp thuật toán chọn một h_j để gán cho một r_i mà kết quả trả về là một phép ghép có xung đột tối thiểu với $r_k \in \mathcal{R}, r_i \neq r_k$, theo nghĩa cặp khối. Điều này có nghĩa rằng cần phải loại bỏ một cặp khối nào đó sao cho cải tiến được tính ổn định của phép ghép. Với ý tưởng này, chúng tôi sử dụng khái niệm cặp khối vượt trội để giải quyết bài toán.

Thuật toán MIN-BPS được mô tả trong Thuật toán 1, trong đó $f(M)$ được định nghĩa là số sinh viên chưa được gán trong phép ghép M. Khởi tạo, thuật toán gán một phép ghép ngẫu nhiên, M, cho phép ghép tốt nhất, M_{best}. Ở mỗi vòng lặp, thuật toán xét một $r_i \in \mathcal{R}$ và tìm một h_j (hàm $find(r_i, M)$) trong danh sách xếp hạng của r_i sao cho cặp (r_i, h_j) là một cặp khối trội nhất của phép ghép hiện thời M. Nếu không tồn tại h_j, nghĩa là phép ghép hiện thời không có cặp khối, thuật toán đã tìm thấy một phép ghép ổn định. Nếu tất cả các sinh viên được ghép trong phép ghép, nghĩa là phép ghép là hoàn chỉnh (tức là $f(M_{best}) = 0$) và thuật toán trả về phép ghép được tìm thấy. Ngược lại, thuật toán khởi tạo lại một phép ghép ngẫu nhiên và lặp lại quá trình tìm kiếm. Nếu tìm thấy một h_j, thuật toán sẽ loại bỏ cặp khối (r_i, h_j) bằng cách gán h_j cho r_i để tạo thành một cặp $(r_i, h_j) \in M$. Tiếp theo, thuật toán kiểm tra nếu h_j vượt quá số sinh viên được xếp chỗ thì nó loại bỏ cặp $(s_w, h_j) \in M$ để h_j vừa đủ chỗ sắp xếp, trong đó s_w là sinh viên được xếp hạng ưu tiên thấp nhất của h_j trong phép ghép M. Thuật toán lặp lại cho đến khi hoặc là M_{best} là một phép ghép hoàn chỉnh hoặc đạt đến một số vòng lặp tối đa được thiết lập trước, trường hợp này thuật toán trả về hoặc là một phép ghép ổn định hoặc là một ghép không ổn định.

Algorithm 1: MIN-BPS Algorithm

Input: - An instance I of HRT.
- A maximum iter. number max_iters

Output: A matching M.

```
1.  function Main(I)
2.      M := a random matching;
3.      M_best := M;
4.      r_i := random(n);
5.      iter := 0;
6.      while (iter ≤ max_iters) do
7.          for (k = 1 ··· n) do
8.              r_i := mod(r_i, n) + 1;
9.              h_j := find(r_i, M);
10.             if (h_j ≠ ∅) then
11.                 break;
12.             end
13.         end
14.         if (h_j = ∅) then
15.             if (f(M_best) > f(M)) then
16.                 M_best := M;
17.             end
18.             if (f(M_best) = 0) then
19.                 break;
20.             else
21.                 M:= a random matching;
22.                 continue;
23.             end
24.         end
25.         M := M ∪ {(r_i, h_j)};
26.         if (h_j is over-subscribed) then
27.             s_w := h_j's worst non-empty student;
28.             M := M\{(s_w, h_j)};
29.         end
30.         iter := iter + 1;
31.     end
32.     return M_best;
33. end function
```

Chú ý rằng hàm $find(r_i, M)$ được dùng để tìm một cặp khối trội nhất (r_i, h_j) cho phép ghép M. Hàm được thực hiện bằng cách xét mỗi h_j trong danh sách xếp hạng của r_i theo thứ tự ưu tiên giảm dần và dừng lại ở cặp khối đầu tiên, khi đó (r_i, h_j) là một cặp khối trội nhất.

IV. THỰC NGHIỆM

Trong phần này chúng tôi mô tả các thực nghiệm để đánh giá hiệu quả của thuật toán MIN-BPS. Chúng tôi áp dụng thủ tục tạo bài toán SMTI [15] để sinh ra các thể hiện của HRT với 4 tham số (n, m, p_1, p_2), trong đó n là số sinh viên, m là số doanh nghiệp, p_1 là xác suất xuất hiện các ứng viên trong danh sách xếp hạng của $r_i \in \mathcal{R}$ và $h_j \in \mathcal{H}$, p_2 là xác suất cho phép $r_i \in \mathcal{R}$ và $h_j \in \mathcal{H}$ xếp hạng các ứng viên với mức ưu tiên bằng nhau. Tất cả các thực nghiệm được thực hiện bởi phần mềm Matlab phiên bản 2017a trên một máy tính cá nhân

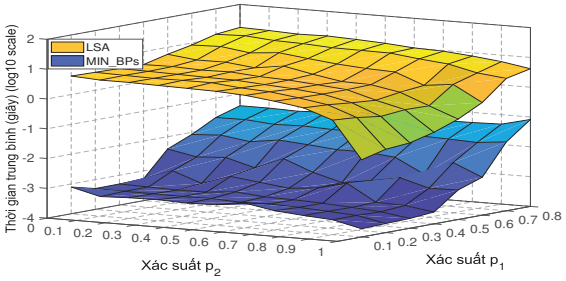
Core i7=8550U với bộ vi xử lý 1.8GHz và bộ nhớ 16Gb.

A. So sánh với thuật toán tìm kiếm cục bộ

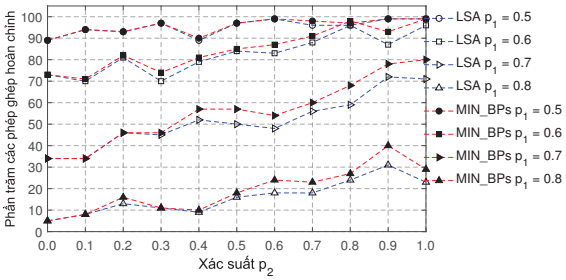
Thực nghiệm 1: Trong phần này chúng tôi mô tả kết quả thực nghiệm để so sánh hiệu quả của thuật toán MIN-BPS với thuật toán LSA [13]. Đầu tiên, chúng tôi tạo ra các thể hiện HRT với $n = 50$, $m = 10$, $p_1 = \{0.1, 0.2, \cdots, 0.8\}$ và $p_2 = \{0.0, 0.1, \cdots, 1.0\}$. Tiếp theo, chúng tôi tạo ra 100 thể hiện HRT cho mỗi tập giá trị (n, m, p_1, p_2), tính trung bình thời gian thực hiện và phần trăm phép ghép hoàn chỉnh được tìm thấy bởi MIN-BPS và LSA.

Hình 1 chỉ ra thời gian thực hiện trung bình của MIN-BPS và LSA. Khi p_1 tăng từ 0.1 đến 0.8, thời gian thực hiện trung bình của MIN-BPS tăng từ khoảng 10^{-3}(giây) đến khoảng 10^{-1}(giây), trong khi thời gian thực hiện trung bình của LSA tăng từ khoảng 10^{-1}(giây) đến khoảng $10^{1.2}$(giây). Khi p_2 tăng từ 0.0 đến 1.0, thời gian thực hiện của MIN-BPS thay đổi không đáng kể, trong khi thời gian thực hiện của LSA có xu hướng giảm dần. Ngoài ra, kết quả thực nghiệm chỉ ra rằng MIN-BPS vượt trội LSA về thời gian thực hiện (vì vậy được lấy log10) cho mọi giá trị của p_1 và p_2. Kết quả này có thể giải thích như sau. LSA là một thuật toán tìm kiếm cục bộ, ở mỗi vòng lặp, LSA phải tìm một tập các phép ghép láng giềng của phép ghép hiện thời, tính tổng số lượng cặp khối cho mỗi phép ghép láng giềng (cần $O(n^2)$ thời gian) và di chuyển phép ghép hiện thời tới phép ghép láng giềng tốt nhất theo nghĩa có số lượng cặp khối bé nhất. Một phép ghép láng giềng được sinh ra bằng cách loại bỏ một cặp khối vượt trội trong phép ghép hiện thời. Vì số lượng phép ghép láng giềng được sinh ra trong mỗi bước lặp là rất lớn, do đó LSA cần nhiều thời gian để tính giá của các phép ghép láng giềng. Ngược lại, MIN-BPS chỉ tìm một cặp khối vượt trội ở mỗi bước lặp để loại bỏ và tạo thành một phép ghép mới, khi loại bỏ một cặp khối vượt trội tạo bởi một r_i nào đó, tất cả cặp khối được tạo bởi r_i sẽ bị loại bỏ trong phép ghép kết quả. Do đó, MIN-BPS vượt trội LSA về thời gian thực hiện.

Hình 2 chỉ ra phần trăm phép ghép hoàn chỉnh được tìm thấy bởi MIN-BPS và LSA (khi p_1 thay đổi từ 0.1 đến 0.4, cả MIN-BPS và LSA đều tìm thấy 100% phép ghép hoàn chỉnh, do đó không được thể hiện trong hình này). Khi p_1 thay đổi từ 0.5 đến 0.8, MIN-BPS vượt trội LSA về chất lượng nghiệm tìm được.



Hình 1: Thời gian thực hiện của MIN-BPS và LSA.



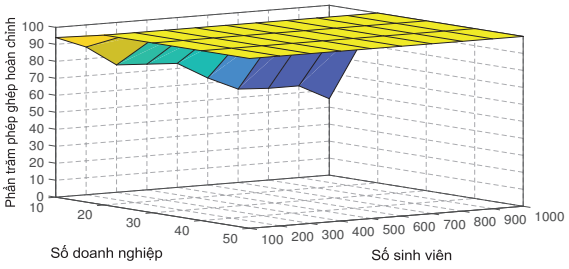
Hình 2: Phần trăm số phép ghép hoàn chỉnh tìm thấy bởi MIN-BPS và LSA.

B. Thực nghiệm với dữ liệu lớn

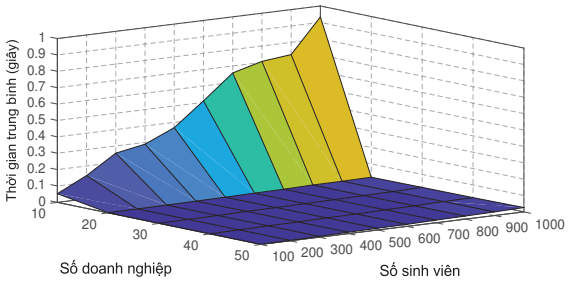
Trong phần này chúng tôi thực hiện các thực nghiệm với bài toán HRT kích thước lớn để xem xét hiệu quả của MIN-BPS. Chúng tôi tạo ra các thể hiện HRT với $n = \{100, 200, \cdots, 1000\}$, $m = \{10, 20, \cdots, 50\}$, $p_1 = 0.5$ và $p_2 = 0.5$. Với mỗi tập giá trị của (n, m, p_1, p_2), chúng tôi sinh ra 100 thể hiện HRT, chạy thuật MIN-BPS tối đa 50000 lần lặp, tính trung bình thời gian thực hiện và phần trăm phép ghép hoàn chỉnh tìm được.

Thực nghiệm 2: Thực nghiệm được thực hiện để đánh giá MIN-BPS với c_j được sinh ngẫu nhiên trong đoạn $[1, q]$, trong đó q là số lượng sinh viên được h_j xếp hạng. Hình 3 chỉ ra phần trăm phép ghép hoàn chỉnh được tìm thấy bởi MIN-BPS. Khi $m = \{20, 30, \cdots, 50\}$, MIN-BPS luôn tìm được phép ghép hoàn chỉnh với mọi n. Khi $m = 10$, phần trăm phép ghép hoàn chỉnh được tìm thấy giảm dần khi n tăng từ 100 đến 1000.

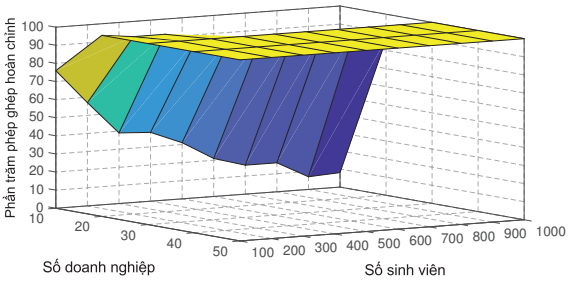
Hình 4 chỉ ra thời gian thực hiện trung bình của MIN-BPS. Khi $m = \{20, 30, 40, 50\}$, thời gian thực hiện của MIN-BPS tăng không đáng kể khi n tăng từ 100 đến 1000. Khi $m = 10$, thời gian thực hiện của MIN-BPS tăng khi n tăng từ 100 đến 1000. Điều này bởi vì phần trăm phép ghép hoàn chỉnh được tìm thấy giảm như trong Hình 3, do đó MIN-BPS phải thực hiện hết 50000 lần lặp. Hơn nữa, khi $m = 10$, thời gian thực hiện của MIN-BPS tăng so với $m = \{20, 30, 40, 50\}$. Điều này cũng vì khi đó MIN-BPS luôn tìm được phép ghép hoàn chỉnh



Hình 3: Phần trăm phép ghép hoàn chỉnh khi c_j ngẫu nhiên và $c_j \in [1, q]$.



Hình 4: Thời gian thực hiện trung bình khi c_j ngẫu nhiên và $c_j \in [1, q]$.

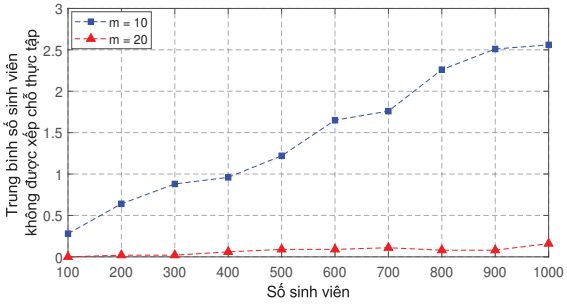


Hình 5: Phần trăm phép ghép hoàn chỉnh khi $c_j = n/m$.

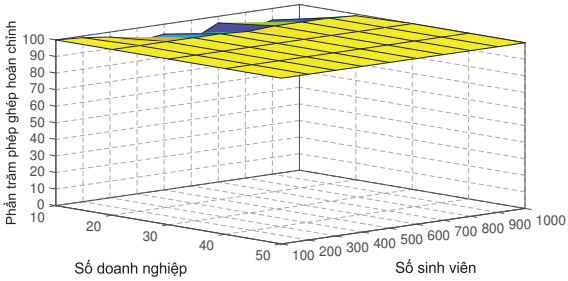
và do đó không cần sử dụng hết 50000 vòng lặp, trong khi $m = 10$, MIN-BPS vẫn còn tìm ra các phép ghép không hoàn chỉnh, do đó MIN-BPS cần lặp hết 50000 lần.

Thực nghiệm 3: Thực nghiệm được thực hiện để đánh giá MIN-BPS với $c_j = n/m$, nghĩa là tổng số chỗ thực tập của các doanh nghiệp đúng bằng số sinh viên. Hình 5 chỉ ra phần trăm phép ghép hoàn chỉnh được tìm thấy bởi MIN-BPS. Thuật toán chỉ tìm được 100% phép ghép hoàn chỉnh khi $m = \{30, 40, 50\}$. Khi $m = \{10, 20\}$, MIN-BPS tìm thấy phép ghép hoàn chỉnh ít hơn trong Thực nghiệm 1, số lượng trung bình các sinh viên không được xếp chỗ thực tập được chỉ ra trong Hình 6.

Thực nghiệm 4: Với kết quả trong Hình 6, chúng tôi tăng $c_j = 1.1 * n/m$, nghĩa là tăng c_j lên 1.1 lần so với Thực nghiệm 3. Hình 7 chỉ ra phần trăm



Hình 6: Trung bình số sinh viên không được xếp chỗ khi $c_j = n/m$ và $m = \{10, 20\}$.



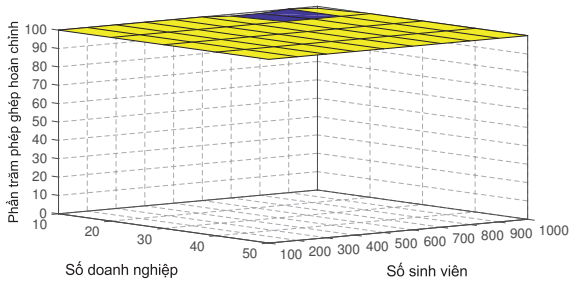
Hình 7: Phần trăm phép ghép hoàn chỉnh khi $c_j = 1.1 * n/m$.

phép ghép hoàn chỉnh được tìm thấy bởi MIN-BPS. Thuật toán luôn tìm được 100% phép ghép hoàn chỉnh khi $m = \{20, 30, 40, 50\}$ với mọi n từ 100 đến 1000. Khi $m = 10$, thuật toán tìm được trên 90% phép ghép hoàn chỉnh. Hiển nhiên, nếu c_j được tăng càng cao, MIN-BPS càng dễ tìm thấy các phép ghép hoàn chỉnh.

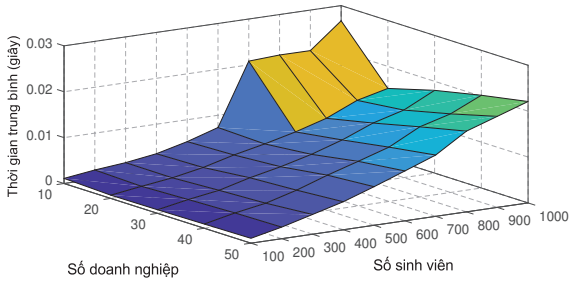
Thực nghiệm 5: Cuối cùng, chúng tôi thực nghiệm với c_j được lấy giá trị ngẫu nhiên trong đoạn $[0.2q, 0.5q]$, trong đó q là số sinh viên được h_j xếp hạng. Điều này có nghĩa rằng doanh nghiệp h_j xếp hạng 50% sinh viên (vì $p_1 = 0.5$), nhưng chỉ chọn từ 10% đến 25% trong số sinh viên được xếp hạng. Hình 8 chỉ ra phần trăm số lượng các phép ghép hoàn chỉnh được tìm thấy bởi MIN-BPS. Với thực nghiệm này, MIN-BPS tìm được 99% cho trường hợp $m = 10$ và $n = 700$ hoặc $n = 800$ và 100% cho các trường hợp còn lại. Hình 9 chỉ ra thời gian thực hiện của MIN-BPS. Thuật toán thực hiện chỉ trong thời gian 0.03 (giây) ngay cả khi $n = 1000$ với các giá trị của $m = \{10, 20, \cdots, 50\}$. Rõ ràng MIN-BPS hiệu quả với HRT kích thước lớn.

V. KẾT LUẬN

Trong bài báo này chúng tôi đề xuất một thuật toán MIN-BPS để tìm phép ghép hoàn chỉnh cho bài toán phân công địa điểm thực tập cho sinh viên. Ý tưởng chính của thuật toán là loại bỏ một cặp khối



Hình 8: Phần trăm phép ghép hoàn chỉnh với $c_j \in [0.2q, 0.5q]$.



Hình 9: Thời gian tìm phép ghép hoàn chỉnh với $c_j \in [0.2q, 0.5q]$.

vượt trội tại mỗi bước lặp để cải tiến tính ổn định của một phép ghép. Kết quả thực nghiệm trên các tập dữ liệu được tạo ngẫu nhiên chỉ ra rằng thuật toán đề xuất giải quyết hiệu quả bài toán kích thước lớn cả thời gian thực hiện và chất lượng nghiệm. Một hướng mở rộng của bài toán là cần đánh giá các điều kiện ràng buộc về số lượng sinh viên và số chỗ thực tập của doanh nghiệp để thuật toán luôn tìm được phép ghép hoàn chỉnh, đảm bảo cho mọi sinh viên có địa điểm thực tập.

TÀI LIỆU

[1] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 9, no. 1, pp. 9–15, 1962.

[2] R. W. Irving, D. F. Manlove, and S. Scott, "The hospitals/residents problem with ties," in *in Proceedings of the 7th Scandinavian Workshop on Algorithm Theory*, Bergen, Norway, Jul. 2000, pp. 259–271.

[3] D. F. Manlove, *The Hospitals/Residents Problem. In: Kao MY. (eds) Encyclopedia of Algorithms*. Boston, MA: Springer, 2008.

[4] P. Biró, D. F. Manlove, and I. McBride, "The hospitals/residents problem with couples: Complexity and integer programming models," in *in Proceeding of SEA 2014: 13th International Symposium on Experimental Algorithms*, Copenhagen, Denmark, Jun. 2014, pp. 10–21.

[5] D. F. Manlove, I. McBride, and J. Trimble, ""almost-stable" matchings in the hospitals / residents problem with couples," *Constraints*, vol. 22, no. 1, pp. 50–72, 2017.

[6] K. Iwama, S. Miyazaki, Y. Morita, and D. Manlove, "Stable marriage with incomplete lists and ties," in *in Proceedings of ICALP 1999: the 26th International Colloquium on Automata, Languages, and Programming*. Springer, 1999, pp. 443–452.

[7] R. W.Irving, D. F.Manlove, and S. Scott, "The stable marriage problem with master preference lists," *Discrete Applied Mathematics*, vol. 156, no. 15, pp. 2959–2977, 2008.

[8] D. F. Manlove, R. W. Irving, K. Iwama, S. Miyazaki, and Y. Morita, "Hardvariants of stable marriage," *Theoretical Computer Science*, vol. 276, no. 1, pp. 261–279, 2002.

[9] A. Kwanashie and D. F. Manlove, "An integer programming approach to the hospitals/residents problem with ties," in *Proceedings of the International Conference on Operations Research*, Erasmus University Rotterdam, Sep. 2013, pp. 263–269.

[10] D. Munera, D. Diaz, S. Abreu, F. Rossi, V. Saraswat, and P. Codognet, "A local search algorithm for smti and its extension to hrt problems," in *Proceedings of the 3rd International Workshop on Matching Under Preferences*, University of Glasgow, UK, Apr. 2015, pp. 66–77.

[11] P. Codognet and D. Diaz, "Yet another local search method for constraint solving," in *Proceedings of the International Symposium on Stochastic Algorithms*, Berlin, Germany, Dec. 2001, pp. 73–90.

[12] S. Russel and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009.

[13] M. Gelain, M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh, "Local search for stable marriage problems with ties and incomplete lists," in *Proceedings of 11th Pacific Rim International Conference on Artificial Intelligence*, Daegu, Korea, Aug. 2010, pp. 64–75.

[14] D. Munera, D. Diaz, S. Abreu, F. Rossi, V. Saraswat, and P. Codognet, "Solving hard stable matching problems via local search and cooperative parallelization," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Austin, Texas, Jan. 2015, pp. 1212–1218.

[15] I. P. Gent and P. Prosser, "An empirical study of the stable marriage problem with ties and incomplete lists," in *in Proceedings of the 15th European Conference on Artificial Intelligence*, Lyon, France, Jul. 2002, pp. 141–145.