BỘ GIÁO DỤC VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC
VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

NGUYỄN THỊ UYÊN

NGHIÊN CỨU MỘT SỐ BIẾN THỂ CỦA BÀI TOÁN HÔN NHÂN ỔN ĐỊNH THEO TIẾP CẬN HEURISTIC

LUẬN ÁN TIẾN SĨ NGÀNH MÁY TÍNH

Hà Nội - 2023

BỘ GIÁO DỤC VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC
VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

NGUYỄN THỊ UYÊN

NGHIÊN CỨU MỘT SỐ BIẾN THỂ CỦA BÀI TOÁN HÔN NHÂN ỔN ĐỊNH THEO TIẾP CẬN HEURISTIC

LUẬN ÁN TIẾN SĨ NGÀNH MÁY TÍNH

Mã số: 9 48 01 01

Xác nhận của Học viện Khoa học và Công nghệ	**Người hướng dẫn 1** *(Ký, ghi rõ họ tên)*	**Người hướng dẫn 2** *(Ký, ghi rõ họ tên)*
	PGS.TS. Hoàng Hữu Việt	PGS.TS Nguyễn Long Giang

Hà Nội - 2023

LỜI CAM ĐOAN

Tôi xin cam đoan các kết quả công bố trong luận án là công trình nghiên cứu của bản thân tôi trong thời gian học tập, nghiên cứu và được hoàn thành với sự hướng dẫn của hai Thầy giáo gồm PGS.TS. Hoàng Hữu Việt và PGS.TS. Nguyễn Long Giang. Các tài liệu tham khảo được trích dẫn đầy đủ và được ghi rõ ở phần tài liệu tham khảo. Các kết quả nghiên cứu được thực nghiệm trên cùng một môi trường thực nghiệm và được ghi nhận một cách khách quan, trung thực và đã được công bố trên các tạp chí khoa học chuyên ngành.

Hà Nội, ngày 31 tháng 03 năm 2023

Nguyễn Thị Uyên

LỜI CẢM ƠN

Luận án này được hoàn thành với sự nỗ lực không ngừng của tác giả và sự giúp đỡ nhiệt tình từ các thầy giáo hướng dẫn, bạn bè và người thân trong suốt 4 năm học tập và nghiên cứu tại Viện Công nghệ thông tin - Viện Hàn lâm Khoa học và Công nghệ Việt Nam.

Đầu tiên, tác giả xin bày tỏ lòng biết ơn chân thành và sâu sắc tới hai Thầy giáo hướng dẫn PGS.TS Hoàng Hữu Việt và PGS.TS Nguyễn Long Giang. Sự tận tình chỉ bảo, hướng dẫn và động viên của các Thầy dành cho tác giả trong suốt thời gian thực hiện luận án. Tác giả xin gửi lời cảm ơn tới các Thầy, Cô giáo và Cán bộ bộ phận quản lý nghiên cứu sinh của Học viện Khoa học và Công nghệ, Viện Hàn lâm Khoa học và Công nghệ Việt Nam và bộ phận quản lý sau đại học của Viện Công nghệ thông tin đã nhiệt tình giúp đỡ và tạo ra môi trường nghiên cứu tốt để tác giả hoàn thành công trình của mình.

Tác giả xin chân thành cảm ơn tới Ban Giám hiệu Trường Đại học Vinh, các đồng nghiệp ở Viện Kỹ thuật và Công nghệ, nơi tác giả đang công tác đã luôn động viên, giúp đỡ tác giả trong công tác để tác giả có thời gian tập trung nghiên cứu và hoàn thành luận án đúng thời hạn. Cuối cùng tác giả muốn bày tỏ lòng biết ơn sâu sắc nhất tới gia đình và bạn bè đã luôn động viên, chia sẻ, ủng hộ và giúp đỡ tác giả vượt qua những khó khăn để đạt được những kết quả nghiên cứu trong luận án.

Tác giả xin trân trọng cảm ơn!

Hà Nội, ngày 31 tháng 03 năm 2023

Nguyễn Thị Uyên

MỤC LỤC

DANH MỤC CÁC CHỮ VIẾT TẮT

STT	Từ viết tắt	Tiếng Anh	Ý nghĩa
1	AS	Adaptive Search	Tìm kiếm thích nghi
2	CSP	Constraint Satisfaction Problem	Bài toán thỏa mãn ràng buộc
3	GS	Gale-Shapley	Thuật toán Gale-Shapley
4	HR	Heuristic Repair	Thuật toán sửa đổi heuristic
5	HRT	Hospitals/Residents problem with Ties	Bài toán phân bổ sinh viên thực tập tại các doanh nghiệp
6	HS	Heuristic Search	Tìm kiếm Heuristic
7	MAX-HRT	Maximum Stable Matching for HRT	Phép ghép ổn định với kích thước tối đa cho HRT
8	MAX-SMTI	Maximum Stable Matching for SMTI	Phép ghép ổn định với kích thước tối đa cho SMTI
9	MAX-SPA	Maximum Stable Matching for SPA	Phép ghép ổn định với kích thước tối đa cho SPA
10	MCA	Min-Conflicts Algorithm	Thuật toán xung đột tối thiểu
11	MCS	Max-Conflicts based heuristic Search	Thuật toán tìm kiếm heuristic dựa trên các xung đột tối đa
12	SMI	Stable Marriage problem with Incomplete lists	Bài toán hôn nhân ổn định với danh sách xếp hạng không đầy đủ
13	SMP	Stable Marriage Problem	Bài toán hôn nhân ổn định
14	SMT	Stable Marriage with Ties Problem	Bài toán hôn nhân ổn định với danh sách xếp hạng ngang hàng
15	SMTI	Stable Marriage with Ties and Incomplete lists	Bài toán hôn nhân ổn định với danh sách xếp hạng ngang hàng và không đầy đủ
16	SPA	Student-Project Allocation Problem	Bài toán phân bổ đề tài cho sinh viên
17	SPA-P	Student-Project Allocation problem with lecturer preferences over Projects	Bài toán phân bổ đề tài cho sinh viên với danh sách xếp hạng của giảng viên dựa vào đề tài
18	SPA-S	Student-Project Allocation problem with lecturer preferences over Students	Bài toán phân bổ đề tài cho sinh viên với danh sách xếp hạng của giảng viên dựa vào sinh viên
19	SPA-ST	Student-Project Allocation problem with lecturer preferences over Students containing Ties	Bài toán phân bổ đề tài cho sinh viên với danh sách xếp hạng của giảng viên dựa vào đề tài có chứa quan hệ ngang hàng
20	UBP	Undominated Blocking Pair	Cặp chặn trội nhất
21	UBPs	Undominated Blocking Pairs	Tập hợp các cặp chặn trội nhất

DANH MỤC CÁC HÌNH VẼ

DANH MỤC CÁC BẢNG BIỂU

MỞ ĐẦU

1. Tính cấp thiết của đề tài luận án

Bài toán hôn nhân ổn định (*Stable Marriage Problem*, SMP) là một bài toán ghép cặp nổi tiếng được giới thiệu lần đầu tiên bởi Gale và Shapley [1]. Bài toán SMP gồm một tập n người nam và một tập n người nữ, trong đó mỗi người xếp hạng "*thích*" những người khác giới theo một thứ tự ưu tiên từ 1 đến n trong một danh sách xếp hạng. Mục đích của bài toán là tìm một phép ghép giữa những người nam và người nữ sao cho thỏa mãn tính ổn định (*stable*) theo một tiêu chuẩn nào đó. Gần đây, bài toán SMP đã nhận được nhiều sự quan tâm của các nhà nghiên cứu trong các lĩnh vực Trí tuệ nhân tạo và Tính toán tối ưu. Năm 2012, Shapley và Roth đã được trao giải thưởng Nobel kinh tế vì các thành tựu đạt được dựa trên các mô hình xuất phát từ bài toán SMP trong lĩnh vực quản lý thị trường chứng khoán tại Mỹ. Ngoài ra, bài toán SMP có nhiều ứng dụng trong thực tế như: *(i)* bài toán phân bổ sinh viên ngành y tới thực tập tại các bệnh viện [2]; *(ii)* bài toán phân công giảng viên hướng dẫn sinh viên thực hiện các đề tài [3]; *(iii)*; *(iv)* bài toán phân bổ nhà ở cho cư dân [4, 5]; *(v)* bài toán tối ưu hóa các yêu cầu dịch vụ của người dùng Internet tới các nhà mạng viễn thông [6, 7]. Gale và Shapley đã đề xuất một thuật toán Gale-Shapley (GS) [1] để tìm ra các cặp ghép ổn định tối ưu từ một phía theo nam (*man-optimal*) hoặc theo nữ (*woman-optimal*). Tuy nhiên do những ràng buộc nghiêm ngặt trong danh sách xếp hạng, tức là yêu cầu số nam và số nữ phải bằng nhau và mỗi người phải xếp hạng tất cả các thành viên khác giới theo một thứ tự ưu tiên nhất định, vì vậy bài toán SMP thường ít gặp trong các ứng dụng thực tế. Do đó, một số biến thể của bài toán SMP đã được đề xuất gần đây như: *(i)* bài toán hôn nhân ổn định với thứ tự ưu tiên ngang bằng (*Stable Marriage problem with Ties*, SMT), tức là một người có thể xếp hạng nhiều người khác giới với thứ tự ưu tiên bằng nhau; *(ii)* bài toán hôn nhân ổn định với danh sách không đầy đủ (*Stable Marriage problem with incomplete*, SMI), tức là một người có thể xếp hạng một số người khác giới trong danh sách xếp hạng; và *(iii)* bài toán hôn nhân ổn định với thứ tự ưu tiên ngang bằng và không đầy đủ (*Stable Marriage Problem with Ties and Incomplete lists*, SMTI), là sự kết hợp của biến thể SMT và SMI. Trong bài toán SMT và SMTI, với sự xuất hiện thứ tự ngang hàng trong danh sách xếp hạng, Irving và cộng sự [8] đã chỉ ra ba điều kiện của phép ghép ổn định gồm: ổn định yếu (*weakly stable*), ổn

định mạnh (*strongly stable*) và siêu ổn định (*super-stable*). Các tác giả đã chứng minh rằng một phép ghép ổn định yếu luôn tồn tại, trong khi phép ghép ổn định mạnh và siêu ổn định có thể không tồn tại với mọi thể hiện của các bài toán SMT và SMTI. Ngoài ra, Manlove và cộng sự [9] đã chứng minh rằng việc tìm một phép ghép ổn định yếu với kích thước tối đa (MAX) là một bài toán NP-khó. Luận án này tập trung nghiên cứu tìm ra một phép ghép ổn định yếu với kích thước tối đa, do vậy để đơn giản, luận án gọi phép ghép *ổn định yếu* là phép ghép *ổn định*. Gần đây, bài toán SMTI được cộng đồng nghiên cứu quan tâm nhiều bởi những ứng dụng của nó trong thực tế như: (*i*) bài toán phân bổ sinh viên ngành y tới thực tập tại các bệnh viện (*Hospitals/Residents with Ties problem*, (HRT)) [2]; (*ii*) bài toán phân công giảng viên hướng dẫn sinh viên thực hiện đề tài (*Student-Project Allocation problem*, SPA) [3, 10]; và (*iii*) bài toán phân bổ nhà ở cho dân cư (*Stable Roommates problem*, SR) [4, 5].

Trong những năm qua, nhiều thuật toán đã được đề xuất giải quyết bài toán SMTI và các biến thể mở rộng như HRT [2] và SPA [3, 11, 10, 12, 13, 14, 15]. Các hướng tiếp cận chủ yếu là thuật toán xấp xỉ [3, 16, 17, 18], lập trình nguyên [19, 20], lập trình thích nghi [14], tìm kiếm cục bộ [11, 21, 22, 23], tìm kiếm bầy đàn [24] và một số phương pháp khác [15, 25, 26]. Bài toán SMTI và các biển thể mở rộng của nó là các bài toán NP-khó [27, 28]. Mặc dù, đã có nhiều hướng tiếp cận để giải các bài toán này, tuy nhiên, các phương pháp đã đề xuất chưa đạt được kết quả mong muốn về chất lượng nghiệm và thời gian thực hiện cho các bài toán này với kích thước lớn. Hướng tiếp cận theo heuristic là các phương pháp giải bài toán dựa trên kinh nghiệm thông qua việc khám phá một phần không gian tìm kiếm nhằm đưa ra một nghiệm chấp nhận được, do vậy hướng tiếp cận này sẽ tăng tốc độ tìm kiếm nghiệm. Ngoài ra, các phương pháp heuristic thường thể hiện khá tự nhiên với cách suy nghĩ và hành động của con người dựa trên các kinh nghiệm về sự hiểu rõ bài toán đặt ra.

2. Mục tiêu luận án

Luận án này tập trung nghiên cứu các thuật toán heuristic để tìm một phép ghép ổn định yếu với kích thước tối đa cho bài toán MAX-SMTI và các mở rộng của bài toán gồm MAX-HRT [2] và MAX-SPA [3, 10]. Mục tiêu nghiên cứu của luận án tập vào ba nội dung chính: *(i)* Đề xuất các thuật toán heuristic để giải quyết bài toán MAX-SMTI; *(ii)* Đề xuất các thuật toán heuristic để giải quyết bài toán MAX-HRT; và *(iii)* Đề xuất các thuật toán heuristic để giải quyết bài toán MAX-SPA. Với mục tiêu đã đặt ra, luận án đạt được các kết quả như sau:

- Đề xuất hai thuật toán giải quyết bài toán MAX-SMTI bao gồm thuật toán MCS và

thuật toán HR. Các đóng góp này được trình bày ở Chương 2 của luận án và công bố ở công trình [A.1] và [A.2].

- Đề xuất hai thuật toán giải quyết bài toán MAX-HRT bao gồm thuật toán MCA và thuật toán HS. Các đóng góp này được trình bày ở Chương 3 của luận án và công bố ở công trình [A.3] và [B.1] (đang gửi tạp chí).
- Đề xuất hai thuật giải quyết bài toán MAX-SPA bao gồm thuật toán SPA-P-heuristic và thuật toán HAG. Các đóng góp này được trình bày ở Chương 4 của luận án và công bố ở công trình [A.4], [B.2] (đang gửi tạp chí) và [A.5].

3. Đối tượng và phạm vi nghiên cứu

3.1. Đối tượng nghiên cứu

- Nghiên cứu tổng quan bài toán MAX-SMTI và các biến thể.
- Nghiên cứu các thuật toán tìm kiếm heuristic cho bài toán MAX-SMTI.
- Nghiên cứu các thuật toán tìm kiếm heuristic cho bài toán MAX-HRT và MAX-SPA.

3.2. Phạm vi của đề tài

- Nghiên cứu đề xuất các thuật toán tìm kiếm heuristic cho bài toán SMTI và các biến thể của bài toán SMTI.
- Nghiên cứu phương pháp tiến hành thực nghiệm, so sánh và đánh giá thuật toán đề xuất so với các hướng tiếp cận khác.

4. Phương pháp nghiên cứu

- *Nghiên cứu lý thuyết*: Nghiên cứu các tài liệu về bài toán hôn nhân ổn định và các biến thể đã công bố ở trong và ngoài nước. Phân tích ưu điểm và các tồn tại của các thuật toán đã đề xuất. Trên cơ sở đó, đề xuất các thuật toán tìm kiếm heuristic để giải quyết bài toán với kích thước lớn.
- *Nghiên cứu thực nghiệm*: Nghiên cứu các các phương pháp xử lý số liệu thực nghiệm; cài đặt các thuật toán đề xuất và nghiên cứu các phương pháp đánh giá hiệu quả của các thuật toán được thực hiện bằng ngôn ngữ lập trình MATLAB.

5. Nội dung nghiên cứu

- Nghiên cứu tổng quan bài toán SMP và các biến thể của bài toán SMP.
- Nghiên cứu các phương pháp đã đề xuất giải quyết bài toán SMTI và các biến thể của bài toán SMTI.
- Nghiên cứu các thuật toán tìm kiếm heuristic để giải quyết bài toán SMTI và các biến thể của bài toán SMTI.
- Cài đặt, thử nghiệm, so sánh và đánh giá các thuật toán đề xuất với các thuật toán khác đã công bố cho bài toán SMTI và các biến thể của bài toán SMTI với kích thước lớn.

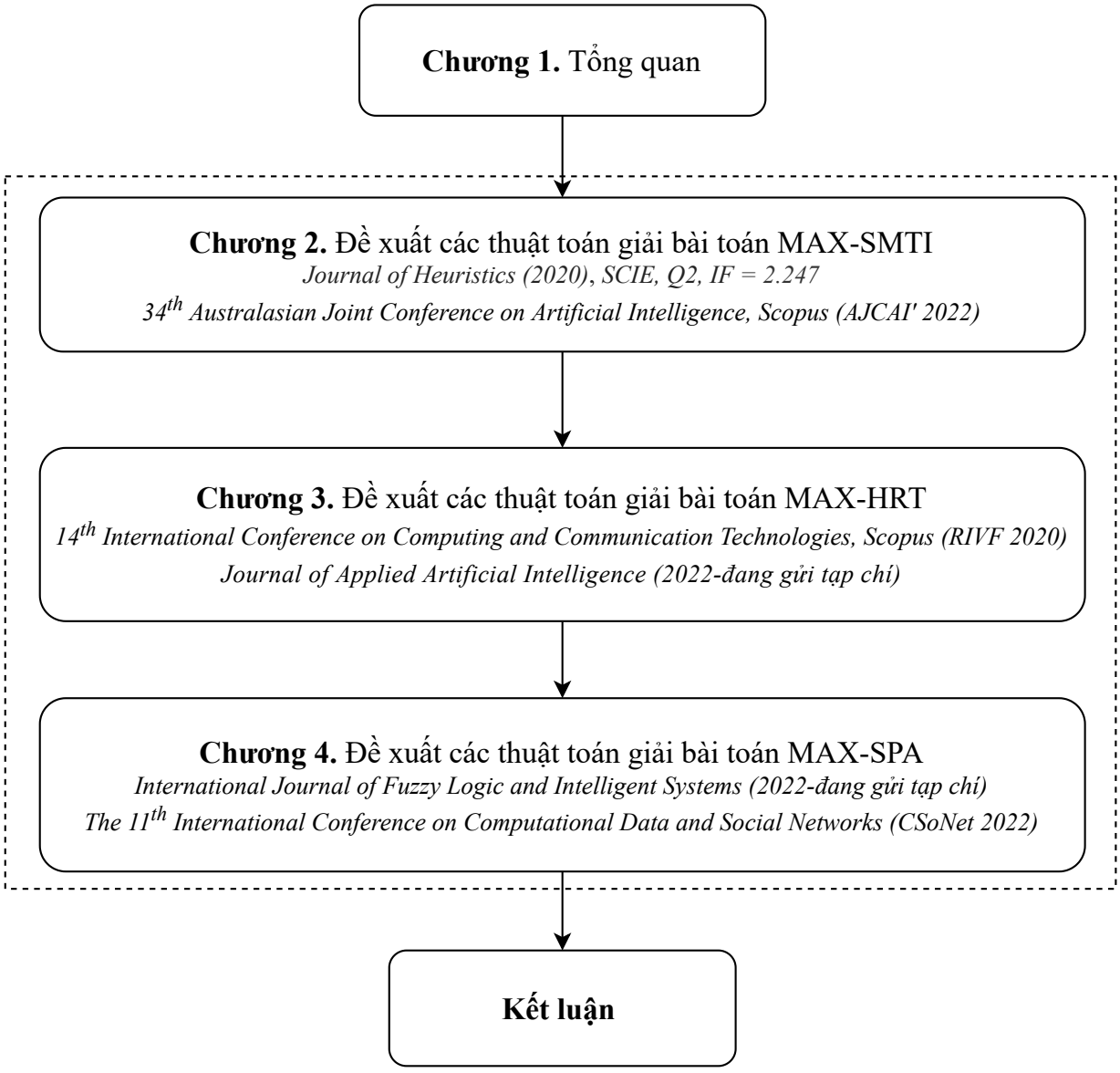
6. Ý nghĩa khoa học và thực tiễn

- **Về khoa học**: Luận án đề xuất các thuật toán tìm kiếm heuristic mới để giải quyết bài toán SMTI các biến thể của bài toán SMTI. Đóng góp của các thuật toán để giải quyết các bài toán là hiệu quả hơn về chất lượng nghiệm và thời gian thực hiện so với các thuật toán trước đây.
- **Về thực tiễn**: Các thuật toán đề xuất có thể áp dụng để giải quyết các bài toán tối ưu thỏa mãn ràng buộc, thuộc lớp bài toán NP-khó cho nhiều ứng dụng thực tế có kích thước lớn trong các lĩnh vực y tế, giáo dục, kinh tế và xã hội.

7. Bố cục của luận án

Bố cục của Luận án gồm phần mở đầu và bốn chương nội dung, phần kết luận và danh mục các tài liệu tham khảo được tổ chức như Hình 1.

- ***Mở đầu***: Trình bày tổng quan về bài toán nghiên cứu, lý do chọn đề tài, đối tượng, mục tiêu và nội dung nghiên cứu của luận án.
- ***Chương 1***: Trình bày cơ sở lý thuyết và tổng quan tình hình nghiên cứu của bài toán SMP và các biến thể.
- ***Chương 2***: Trình bày các thuật toán đề xuất giải bài toán MAX-SMTI.
- ***Chương 3***: Trình bày các thuật toán đề xuất giải bài toán MAX-HRT.
- ***Chương 4***: Trình bày các thuật toán đề xuất giải bài toán MAX-SPA.
- ***Kết luận***: Trình bày những kết quả đã đạt được và hướng phát triển trong tương lai.



Hình 1: *Cấu trúc của luận án*

CHƯƠNG 1.
TỔNG QUAN VỀ BÀI TOÁN HÔN NHÂN ỔN ĐỊNH

Chương này trình bày các khái niệm, các định nghĩa, các ví dụ và tổng quan tình hình nghiên cứu liên quan đến bài toán hôn nhân ổn định và các biến thể mở rộng của bài toán.

1.1. Bài toán hôn nhân ổn định

1.1.1. Giới thiệu

Bài toán hôn nhân ổn định (*Stable Marriage Problem*, SMP) là một bài toán ghép cặp nổi tiếng được giới thiệu bởi Gale and Shapley năm 1962 [1]. Bài toán bao gồm một tập nam và một tập nữ có số lượng bằng nhau và mục tiêu của bài toán là tìm một phép ghép giữa nam và nữ sao cho thỏa mãn một điều kiện tối ưu nào đó. Gần đây bài toán này được cộng đồng nghiên cứu rất quan tâm bởi những ứng dụng rộng lớn của nó trong thực tế như các chương trình phân bổ dân cư tại Mỹ (National Resident Matching Program-NRMP) [4]; ứng dụng phát triển của thị trường lao động cho sinh viên và nhân viên y tế (Evolution of the Labor Market for Medical Interns and Residents) [29]; chương trình đăng ký nhà ở tại Scotland (Scottish Pre-registration house officer Allocations) [30]; dịch vụ phân bố dân cư tại Canada (Resident Matching Service-CaRMS) [31]; và một số ứng dụng về việc tối ưu hóa dịch vụ của người dùng Internet [32].

Định nghĩa 1.1 (Thể hiện SMP [1]). *Một thể hiện I (instance) của bài toán* SMP *kích thước n gồm một tập $\mathcal{M} = \{m_1, m_2, \cdots, m_n\}$ người nam và một tập $\mathcal{W} = \{w_1, w_2, \cdots, w_n\}$ người nữ, trong đó mỗi người có một danh sách xếp hạng "thích" những người khác giới theo một thứ tự ưu tiên nghiêm ngặt.*

Định nghĩa 1.2 (Phép ghép [1]). *Một phép ghép M (matching) là một tập $M = \{(m_i, w_j) \in \mathcal{M} \times \mathcal{W}\}$ gồm n cặp, trong đó mỗi $m_i \in \mathcal{M}$ chỉ được ghép duy nhất với một $w_j \in \mathcal{W}$ và ngược lại.*

Một cặp $(m_i, w_j) \in M$ được gọi là một cặp ghép trong M và được ký hiệu bởi $m_i = M(w_j)$ và $w_j = M(m_i)$. Ký hiệu $rank(m_i, w_j)$ là thứ hạng của $w_j \in \mathcal{W}$ trong danh sách xếp

Bảng 1.1: *Một thể hiện của bài toán* SMP

Danh sách xếp hạng của $m_i \in \mathcal{M}$	Danh sách xếp hạng của $w_j \in \mathcal{W}$
m_1: w_4 w_7 w_3 w_8 w_1 w_5 w_2 w_6	w_1: m_1 m_3 m_5 m_4 m_2 m_6 m_8 m_7
m_2: w_5 w_3 w_4 w_2 w_1 w_8 w_6 w_7	w_2: m_8 m_2 m_4 m_5 m_3 m_7 m_1 m_6
m_3: w_3 w_8 w_2 w_4 w_6 w_7 w_5 w_1	w_3: m_5 m_8 m_1 m_4 m_2 m_3 m_6 m_7
m_4: w_5 w_6 w_8 w_3 w_4 w_7 w_1 w_2	w_4: m_2 m_4 m_3 m_6 m_5 m_8 m_1 m_7
m_5: w_1 w_3 w_5 w_2 w_8 w_6 w_4 w_7	w_5: m_6 m_5 m_4 m_8 m_1 m_7 m_2 m_3
m_6: w_8 w_6 w_2 w_5 w_1 w_7 w_4 w_3	w_6: m_7 m_4 m_2 m_5 m_6 m_8 m_1 m_3
m_7: w_2 w_5 w_8 w_3 w_6 w_4 w_7 w_1	w_7: m_3 m_8 m_6 m_5 m_7 m_2 m_1 m_4
m_8: w_5 w_7 w_4 w_1 w_6 w_2 w_8 w_3	w_8: m_4 m_7 m_1 m_3 m_5 m_8 m_2 m_6

hạng của $m_i \in \mathcal{M}$ và $rank(w_j, m_i)$ là thứ hạng của $m_i \in \mathcal{M}$ trong danh sách xếp hạng của $w_j \in \mathcal{W}$. Nếu $m_i \in \mathcal{M}$ thích $w_j \in \mathcal{W}$ hơn $w_k \in \mathcal{W}$ nghĩa là $rank(m_i, w_j) < rank(m_i, w_k)$ và nếu $w_j \in \mathcal{W}$ thích $m_i \in \mathcal{M}$ hơn $m_t \in \mathcal{M}$ nghĩa là $rank(w_j, m_i) < rank(w_j, m_t)$.

Định nghĩa 1.3 (Cặp chặn [23])**.** *Một cặp $(m_i, w_j) \in \mathcal{M} \times \mathcal{W}$ là một cặp chặn (blocking pair) cho một phép ghép M nếu thỏa mãn các điều kiện:*

1. $rank(m_i, w_j) < rank(m_i, M(m_i))$;
2. $rank(w_j, m_i) < rank(w_j, M(w_j))$.

Định nghĩa 1.4 (Phép ghép ổn định [23])**.** *Một phép ghép M là ổn định (stable) nếu không tồn tại bất kỳ cặp chặn $(m_i, w_j) \in \mathcal{M} \times \mathcal{W}$ cho M, ngược lại M được gọi là không ổn định.*

Một thể hiện SMP gồm 8 người nam và 8 người nữ được chỉ ra trong Bảng 1.1. Phép ghép $M = \{(m_1, w_3), (m_2, w_1), (m_3, w_2), (m_4, w_8), (m_5, w_7), (m_6, w_4), (m_7, w_5), (m_8, w_6)\}$ là phép ghép không ổn định vì tồn tại các cặp chặn $\{(m_2, w_2), (m_2, w_4), (m_4, w_5), (m_4, w_6), (m_5, w_1), (m_5, w_2), (m_5, w_3), (m_5, w_5), (m_5, w_6), (m_6, w_5), (m_6, w_6), (m_6, w_7), (m_8, w_5), (m_8, w_7)\}$ cho M. Phép ghép $M = \{(m_1, w_3), (m_2, w_4), (m_3, w_2), (m_4, w_5), (m_5, w_1), (m_6, w_6), (m_7, w_8), (m_8, w_7)\}$ là phép ghép ổn định.

1.1.2. Các nghiên cứu liên quan

Phần này trình bày các hướng nghiên liên quan giải quyết bài toán SMP. Năm 1962, bài toán SMP được Gale và Shapley [1] đề xuất và nghiên cứu. Họ đã trình bày một thuật toán nổi tiếng gọi là thuật toán Gale-Shapley (GS) để tìm một phép ghép tối ưu cho tập nam hoặc tập nữ trong thời gian $O(n^2)$. Tuy nhiên, phép ghép ổn định tối ưu cho tập nam và phép ghép ổn định tối ưu cho tập nữ là những phép ghép "*ích kỷ*", tức là trong thuật toán GS những người đề xuất luôn có được bạn ghép mà mình thích nhất còn người được ghép luôn nhận được bạn ghép tồi nhất trong danh sách xếp hạng của họ. Vì vậy các nghiên cứu sau

Bảng 1.2: *Các nghiên cứu liên quan giải quyết bài toán* SMP

Tác giả, năm xuất bản	Thuật toán	Mục tiêu
Thuật toán xấp xỉ		
Gale và Shapley [1], 1962	Thuật toán Gale-Shapley	Phép ghép tối ưu một phía
KazuoIwama và cộng sự [33], 2010	Thuật toán xấp xỉ	Phép ghép cân bằng giới tính
Thuật toán heuristic		
Codognet và cộng sự [43], 2001	Thuật toán tìm kiếm cục bộ	Mô hình bài toán SMP như bài toán thỏa mãn ràng buộc (CSP)
Gent và cộng sự [44], 2002	Thuật toán sinh ngẫu nhiên	Sinh ngẫu nhiên các thể hiện SMP
Nakamura và cộng sự [34], 2005	Thuật toán di truyền (GA)	Phép ghép ổn định hai phía
Vien và cộng sự [24], 2007	Thuật toán tối ưu đàn kiến (ACS)	Phép ghép ổn định hai phía
Gelain và cộng sự [45], 2010	Thuật toán tìm kiếm cục bộ (SML)	Phép ghép ổn định hai phía
Việt và các cộng sự [36], 2019	Thuật toán Short list-BilS	Phép ghép ổn định hai phía
Hướng tiếp cận khác		
Zavidovique và cộng sự [37], 2005	Thuật toán ZigZag	Phép ghép cân bằng theo giới tính
Iwama và cộng sự [38], 2010	Thuật toán xấp xỉ	Phép ghép ổn định hai phía
Everaere và cộng sự [39], 2013	Thuật toán Swing	Phép ghép tương đương theo giới tính
Giannakopoulos và cộng sự [40], 2015	Thuật toán ESMA dựa trên Swing	Phép ghép tương đương theo giới tính
Tayu và cộng sự [46], 2017	Sử dụng cấu trúc Cây	Phép ghép tối ưu từ hai phía

này thường tập trung vào việc tìm các phép ghép tối ưu tối cân bằng cho cả tập nam và tập nữ. Một số phương pháp nghiên cứu để giải quyết bài toán này đã được đề xuất như: phương pháp xấp xỉ [1, 33], phương pháp tìm kiếm heuristic [34, 24, 11, 35, 36], và một số phương pháp nghiên cứu khác [37, 38, 39, 40, 41, 42] như được trình bày trong trong Bảng 1.2.

Mặc dù, có nhiều nghiên cứu đã được đề xuất để giải quyết bài toán SMP, tuy nhiên bài toán SMP ít được ứng dụng trong thực tế bởi các yêu cầu ràng buộc nghiêm ngặt của danh sách xếp hạng, tức là mỗi $m_i \in \mathcal{M}$ phải xếp hạng tất cả $w_j \in \mathcal{W}$ và ngược lại. Do đó, những năm gần đây một số biến thể mới của bài toán SMP đã được giới thiệu và ứng dụng nhiều trong thực tế. Vì vậy, luận án tập trung nghiên cứu và đề xuất các thuật toán theo hướng tiếp cận heuristic để giải quyết các biến thể của bài toán SMP và các ứng dụng mở rộng.

1.2. Các biến thể của bài toán hôn nhân ổn định

1.2.1. Bài toán hôn nhân ổn định với danh sách xếp hạng ngang bằng

Một biến thể đầu tiên của bài toán SMP được gọi là bài toán hôn nhân ổn định với danh sách xếp hạng ngang bằng (*Stable Marriage Problem with Ties*, SMT) [47], nghĩa là một người có thể xếp hạng "*thích*" một số người khác giới với ưu tiên bằng nhau. Một thể hiện SMT gồm 8 người nam và 8 người nữ được minh họa trong Bảng 1.3, trong đó mỗi $m_i \in \mathcal{M}$ có thể xếp hạng "*thích*" một số $w_j \in \mathcal{W}$ cùng một thứ tự ưu tiên và ngược lại. Ví dụ: m_1: $(w_4\ w_3)\ w_1\ w_5\ w_2\ w_6\ w_8\ w_7$, nghĩa là m_1 xếp hạng w_4 và w_3 cùng một thứ tự ưu tiên trong danh xếp hạng của m_1. Từ đây về sau, luận án quy ước sử dụng ký hiệu "()" để mô tả thứ tự ưu tiên bằng nhau trong danh sách xếp hạng của $m_i \in \mathcal{M}$ và $w_j \in \mathcal{W}$.

Bảng 1.3: *Một thể hiện của bài toán* SMT

Danh sách xếp hạng của $m_i \in \mathcal{M}$	Danh sách xếp hạng của $w_j \in \mathcal{W}$
m_1: $(w_4\ w_3)\ w_1\ w_5\ w_2\ w_6\ w_8\ w_7$	w_1: $m_4\ (m_7\ m_3)\ m_8\ m_1\ m_5\ m_2\ m_6$
m_2: $w_2\ (w_8\ w_4)\ w_5\ w_3\ w_7\ w_1\ w_6$	w_2: $m_5\ m_3\ (m_4\ m_2)\ m_1\ m_8\ m_6\ m_7$
m_3: $w_5\ w_8\ w_1\ (w_4\ w_2)\ w_3\ w_6\ w_7$	w_3: $m_2\ m_8\ (m_6\ m_4)\ m_3\ m_7\ m_5\ m_1$
m_4: $w_6\ (w_4\ w_3\ w_2)\ w_5\ w_8\ w_1\ w_7$	w_4: $m_5\ m_6\ (m_8\ m_3)\ m_4\ m_7\ m_1\ m_2$
m_5: $w_6\ (w_5\ w_4)\ w_8\ w_1\ w_7\ w_2\ w_3$	w_5: $m_1\ m_8\ m_5\ m_2\ m_3\ (m_6\ m_4\ m_7)$
m_6: $w_7\ w_4\ (w_2\ w_5\ w_6)\ w_8\ w_1\ w_3$	w_6: $m_8\ (m_6\ m_2)\ m_5\ m_1\ m_7\ m_4\ m_3$
m_7: $w_8\ (w_5\ w_6\ w_3)\ w_7\ w_2\ w_1\ w_4$	w_7: $(m_5\ m_2)\ m_8\ m_3\ m_6\ m_4\ m_7\ m_1$
m_8: $w_4\ w_7\ (w_1\ w_3\ w_5)\ w_8\ w_2\ w_6$	w_8: $m_4\ (m_5\ m_7)\ m_1\ m_6\ m_2\ m_8\ m_3$

Trong bài toán SMT, các định nghĩa về phép ghép M tương tự như bài toán SMP đã được trình bày trong Phần 1.1. Tuy nhiên với bài toán SMT, một phép ghép M được xem xét với khả năng ổn định yếu (*weakly stable*), ổn định mạnh (*strongly stable*), và ổn định siêu mạnh (*super-stable*) [48].

Một số thuật toán xấp xỉ đã được đề xuất để giải quyết bài toán SMT. Một thuật toán T được gọi là một thuật toán r-xấp xỉ (r–approximation) cho một bài toán tối ưu nếu T luôn tìm ra một nghiệm $T(x)$ thỏa mãn $|T(x)| \geq |opt(x)|/r$ cho tất cả các thể hiện x của bài toán, trong đó $opt(x)$ là nghiệm tối ưu của thể hiện x. Manlove và cộng sự [9] đã đề xuất hai thuật toán 2-xấp xỉ để tìm nghiệm gần đúng trong thời gian đa thức cho bài toán SMT. Halldorsson và cộng sự [49] đã đề xuất một thuật toán với độ phức tạp đa thức để tìm nghiệm xấp xỉ tối ưu bình đẳng và tối ưu tương đương cho bài toán SMT dựa trên lý thuyết đồ thị. Gần đây, Daniel Marx và cộng sự [50] đã nghiên cứu các thuật toán tìm kiếm cục bộ để tìm ra nghiệm tối ưu bình đẳng. Ngoài ra, Trang và các cộng sự [22] đã đề xuất các thuật toán tìm kiếm cục bộ để giải quyết bài toán SMT nhằm tìm kiếm các phép ghép ổn định tối ưu bình đẳng và tối ưu tương đương theo giới tính. Nakamura và cộng sự [51] đã sử dụng cấu trúc cây để tìm phép ghép ổn định cho SMT trong thời gian đa thức. Nhìn chung, các thuật toán đề xuất là thuật toán xấp xỉ và thuật toán tìm kiếm cục bộ và những thuật toán này đã giải quyết tương đối tốt cho bài toán SMT. Tuy nhiên trong thực tế các ứng dụng biến thể SMT chưa được sử dụng phổ biến và vì vậy các nhà nghiên cứu tiếp tục khai thác các biến thể khác của bài toán SMP nhằm giải quyết các vấn đề có tính thực tiễn.

1.2.2. Bài toán hôn nhân ổn định với danh sách không đầy đủ

Một biến thể khác của SMP được gọi là bài toán hôn nhân ổn định với danh sách xếp hạng không đầy đủ (*Stable Marriage Problem with Incomplete*, SMI) [48], nghĩa là mỗi người có thể xếp hạng "*thích*" một số người khác giới trong danh sách xếp hạng của họ.

Một thể hiện SMI gồm 8 người nam và 8 người nữ được mô tả trong Bảng 1.4, trong đó nếu viết m_1: w_5 w_8 w_7, nghĩa là m_1 chỉ xếp hạng w_5, w_8 và w_7 trong danh sách xếp hạng của m_1.

Bảng 1.4: *Một thể hiện của bài toán* SMI

Danh sách xếp hạng của $m_i \in \mathcal{M}$	Danh sách xếp hạng của $w_j \in \mathcal{W}$
m_1: w_5 w_8 w_7	w_1: m_4 m_5 m_2 m_6
m_2: w_2 w_8 w_1 w_6	w_2: m_4 m_2 m_1
m_3: w_5 w_4 w_2 w_6 w_7	w_3: m_8 m_6 m_5 m_1
m_4: w_5 w_8 w_1 w_7	w_4: m_5 m_6 m_8 m_3 m_4
m_5: w_6 w_7 w_2 w_3	w_5: m_1 m_8 m_3 m_6 m_4 m_7
m_6: w_6 w_8 w_1 w_3	w_6: m_8 m_6 m_5 m_1
m_7: w_8 w_5 w_6 w_4	w_7: m_5 m_2 m_3 m_6
m_8: w_4 w_2 w_6	w_8: m_4 m_5 m_1 m_6

Tương tự như biến thể SMT, một số thuật toán đã đề xuất để giải quyết bài toán SMI là các thuật toán xấp xỉ [23, 52, 47]. Tuy nhiên, bài toán SMI cũng không phổ biến trong các ứng dụng thực tế. Do vậy, luận án tập trung nghiên cứu biến thể khác của bài toán SMP như bài toán SMTI [53] và các ứng dụng mở rộng của bài toán này trong các phần tiếp theo.

1.2.3. Bài toán hôn nhân ổn định với danh sách xếp hạng ngang bằng và không đầy đủ

Bài toán hôn nhân ổn định với danh sách xếp hạng ngang bằng và không đầy đủ (*Stable marriage problem with Ties and Incomplete lists*, SMTI) [53, 9] là sự kết hợp của hai biến thể SMT và SMI.

Định nghĩa 1.5 (Thể hiện SMTI [11, 53])**.** *Một thể hiện I của bài toán* SMTI *kích thước n gồm một tập $\mathcal{M} = \{m_1, m_2, \cdots, m_n\}$ người nam và một tập $\mathcal{W} = \{w_1, w_2, \cdots, w_n\}$ người nữ, trong đó mỗi người có thể xếp hạng "thích" một số người khác giới theo các thứ tự có thể ngang bằng trong danh sách xếp hạng.*

Ký hiệu $rank(m_i, w_j)$ là thứ hạng của $w_j \in \mathcal{W}$ trong danh sách xếp hạng của $m_i \in \mathcal{M}$ và $rank(w_j, m_i)$ là thứ hạng của $m_i \in \mathcal{M}$ trong danh sách xếp hạng của $w_j \in \mathcal{W}$. Nếu $m_i \in \mathcal{M}$ thực sự thích $w_j \in \mathcal{W}$ hơn $w_k \in \mathcal{W}$ nghĩa là $rank(m_i, w_j) < rank(m_i, w_k)$ và nếu $m_i \in \mathcal{M}$ thích $w_j \in \mathcal{W}$ và $w_k \in \mathcal{W}$ như nhau nghĩa là $rank(m_i, w_j) = rank(m_i, w_k)$. Tương tự, nếu $w_j \in \mathcal{W}$ thực sự thích $m_i \in \mathcal{M}$ hơn $m_k \in \mathcal{M}$ nghĩa là $rank(w_j, m_i) < rank(w_j, m_k)$ và nếu $w_j \in \mathcal{W}$ thích $m_i \in \mathcal{M}$ và $m_k \in \mathcal{M}$ như nhau nghĩa là $rank(w_j, m_i) = rank(w_j, m_k)$.

Định nghĩa 1.6 (Cặp chấp nhận [45]). *Một cặp* $(m_i, w_j) \in \mathcal{M} \times \mathcal{W}$ *là một cặp chấp nhận nếu* $rank(m_i, w_j) > 0$ *và* $rank(w_j, m_i) > 0$.

Định nghĩa 1.7 (Phép ghép [45, 14]). *Một phép ghép* M *là một tập* $M = \{(m_i, w_j) \in \mathcal{M} \times \mathcal{W} | rank(m_i, w_j) > 0$ *và* $rank(w_j, m_i) > 0\}$, *trong đó mỗi* $m_i \in \mathcal{M}$ *chỉ được ghép với một* $w_j \in \mathcal{W}$ *và ngược lại. Nếu* $(m_i, w_j) \in M$, *thì* m_i *và* w_j *được gọi là bạn ghép của nhau, ký hiệu* $m_i = M(w_j)$ *và* $w_j = M(m_i)$. *Nếu* $m_i \in \mathcal{M}$ *không được ghép trong* M, *thì* m_i *được gọi là độc thân và ký hiệu* $M(m_i) = \varnothing$. *Tương tự, nếu* $w_j \in \mathcal{W}$ *không được ghép trong* M, *thì* w_j *được gọi là độc thân và ký hiệu* $M(w_j) = \varnothing$.

Với việc xuất hiện danh sách xếp hạng ngang bằng trong các thể hiện SMTI, định nghĩa về tính ổn định của một phép ghép M gồm: ổn định yếu (*weakly stable*), ổn định mạnh (*strongly stable*), và ổn định siêu mạnh (*super-stable*) [48]. Sau đây, luận án sẽ trình bày ba định nghĩa của cặp chặn tương ứng với ba điều kiện của ba phép ghép như sau:

Định nghĩa 1.8 (Cặp chặn yếu [45, 14]). *Một cặp* $(m_i, w_j) \in \mathcal{M} \times \mathcal{W}$ *là một cặp chặn yếu (weakly blocking pair) cho một phép ghép* M *nếu thỏa mãn các điều kiện:*

1. $rank(m_i, w_j) > 0$ *và* $rank(w_j, m_i) > 0$, *tức là* (m_i, w_j) *là một cặp chấp nhận;*
2. $M(m_i) = \varnothing$ *hoặc* $rank(m_i, w_j) < rank(m_i, M(m_i))$;
3. $M(w_j) = \varnothing$ *hoặc* $rank(w_j, m_i) < rank(w_j, M(w_j))$.

Định nghĩa 1.9 (Cặp chặn mạnh [45, 14]). *Một cặp* $(m_i, w_j) \in \mathcal{M} \times \mathcal{W}$ *là một cặp chặn mạnh (strongly blocking pair) cho một phép ghép* M *nếu thỏa mãn các điều kiện:*

1. $rank(m_i, w_j) > 0$ *và* $rank(w_j, m_i) > 0$, *tức là* (m_i, w_j) *là một cặp chấp nhận;*
2. $M(m_i) = \varnothing$ *hoặc* $rank(m_i, w_j) < rank(m_i, M(m_i))$;
3. $M(w_j) = \varnothing$ *hoặc* $rank(w_j, m_i) \leq rank(w_j, M(w_j))$.

Định nghĩa 1.10 (Cặp chặn siêu mạnh [45, 14]). *Một cặp* $(m_i, w_j) \in \mathcal{M} \times \mathcal{W}$ *là một chặn siêu mạnh (super blocking pair) cho một phép ghép* M *nếu thỏa mãn các điều kiện:*

1. $rank(m_i, w_j) > 0$ *và* $rank(w_j, m_i) > 0$, *tức là* (m_i, w_j) *là một cặp chấp nhận;*
2. $M(m_i) = \varnothing$ *hoặc* $rank(m_i, w_j) \leq rank(m_i, M(m_i))$;
3. $M(w_j) = \varnothing$ *hoặc* $rank(w_j, m_i) \leq rank(w_j, M(w_j))$.

Cho một thể hiện SMTI, Irving và cộng sự [53] đã chứng minh rằng luôn tồn tại một phép ghép ổn định yếu bằng cách phá vỡ các ưu tiên ngang bằng trong danh sách xếp hạng của $m_i \in \mathcal{M}$ và $w_j \in \mathcal{W}$ và hơn nữa, một phép ghép siêu ổn định là ổn định mạnh và một phép ghép ổn định mạnh là ổn định yếu [54]. Họ cũng đã chứng minh rằng một phép ổn định

mạnh và siêu ổn định có thể không tồn tại, do đó mục tiêu của bài toán SMTI là tìm một phép ghép ổn định yếu trong đó có nhiều nhất số người nam được ghép, hay còn gọi là bài toán MAX-SMTI [53]. Manlove và cộng sự [2] đã chứng minh rằng MAX-SMTI là một bài toán NP-khó ngay cả khi danh sách xếp hạng có thứ tự ưu tiên ngang bằng chỉ từ phía người nam hoặc từ phía người nữ. Vì vậy, trong các phần tiếp theo, luận án tập trung trình bày các định nghĩa liên quan tới phép ghép ổn định yếu. Để đơn giản, luận án gọi một cặp chặn yếu là cặp chặn và một phép ghép ổn định yếu là phép ghép ổn định.

Định nghĩa 1.11 (Cặp chặn vượt trội [45, 14]). *Một cặp chặn $(m_i, w_j) \in \mathcal{M} \times \mathcal{W}$ vượt trội (dominate) một cặp chặn $(m_i, w_k) \in \mathcal{M} \times \mathcal{W}$ theo xếp hạng của m_i nếu $rank(m_i, w_j) < rank(m_i, w_k)$.*

Định nghĩa 1.12 (Cặp chặn trội nhất [45, 14]). *Một cặp chặn $(m_i, w_j) \in \mathcal{M} \times \mathcal{W}$ là cặp chặn trội nhất (undominated blocking pair) theo xếp hạng của m_i nếu không tồn tại cặp chặn (m_i, w_k) mà $rank(m_i, w_k) < rank(m_i, w_j)$.*

Định nghĩa 1.13 (Phép ghép ổn định [45, 14]). *Một phép ghép M là ổn định nếu không tồn tại bất kỳ cặp chặn $(m_i, w_j) \in \mathcal{M} \times \mathcal{W}$ cho M, ngược lại M được gọi là không ổn định.*

Định nghĩa 1.14 (Kích thước phép ghép [45, 14]). *Kích thước của một phép ghép ổn định M là tổng số cặp $(m_i, w_j) \in M$ và được ký hiệu là $|M|$.*

Định nghĩa 1.15 (Phép ghép hoàn chỉnh [45, 14]). *Một phép ghép ổn định M gọi là hoàn chỉnh nếu $|M| = n$, ngược lại M được gọi là không hoàn chỉnh.*

Một thể hiện SMTI gồm 8 người nam và 8 người nữ được mô tả trong Bảng 1.5. Cặp (m_1, w_1) là cặp chấp nhận vì $rank(m_1, w_1) = 1$ và $rank(w_1, m_1) = 1$. Một phép

Bảng 1.5: *Ví dụ của một thể hiện* SMTI

Danh sách xếp hạng của $m_i \in \mathcal{M}$	Danh sách xếp hạng của $w_j \in \mathcal{W}$
m_1: w_1	w_1: m_1 $(m_5$ $m_6)$
m_2: w_5 $(w_3$ w_4 $w_6)$ $(w_7$ $w_8)$	w_2: $(m_3$ m_5 $m_6)$
m_3: w_4 $(w_2$ $w_5)$	w_3: m_6 $(m_7$ $m_8)$ m_5 m_2
m_4: $(w_5$ $w_6)$ w_8 w_7	w_4: m_3 $(m_2$ m_6 $m_7)$ m_5
m_5: $(w_1$ $w_3)$ $(w_4$ $w_5)$ w_2	w_5: $(m_5$ m_7 $m_8)$ $(m_3$ $m_4)$ m_2
m_6: $(w_4$ $w_7)$ w_1 $(w_2$ w_3 $w_8)$	w_6: m_2 m_7 $(m_4$ $m_8)$
m_7: w_4 w_6 $(w_3$ w_5 $w_7)$	w_7: $(m_2$ $m_6)$ m_7 m_4
m_8: w_5 w_6 w_3	w_8: $(m_2$ $m_4)$ m_6

ghép là $M = \{(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), (m_5, \varnothing), (m_6, w_2), (m_7, w_7), (m_8, \varnothing),$

$(\varnothing, w_3)$, $(\varnothing, w_5)\}$, trong đó $M(m_1) = w_1$, $M(w_1) = m_1$, và $M(m_5) = M(w_3) = \varnothing$, tức là m_5 và w_3 là độc thân. Phép ghép $M = \{(m_1, w_1), (m_2, w_5), (m_3, \varnothing), (m_4, w_6), (m_5, w_2), (m_6, w_4), (m_7, w_3), (m_8, \varnothing), (\varnothing, w_7), (\varnothing, w_8)\}$ có các cặp chặn gồm $\{(m_3, w_4), (m_3, w_5), (m_5, w_5), (m_7, w_6), (m_8, w_8)\}$, trong đó cặp chặn (m_3, w_4) vượt trội cặp chặn (m_3, w_5) theo xếp hạng ưu tiên của m_3 vì $rank(m_3, w_4) < rank(m_3, w_5)$ và cặp chặn (m_3, w_4) là trội nhất vì không có cặp chặn nào vượt trội (m_3, w_4) theo xếp hạng ưu tiên của m_3. Phép ghép $M = \{(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), (m_5, w_5), (m_6, w_7), (m_7, w_3), (m_8, \varnothing), (\varnothing, w_2)\}$ là một phép ghép ổn định vì không tồn tại bất kỳ cặp chặn nào cho M, trong đó m_8 và w_2 là độc thân. Hơn nữa, M là một phép ghép không hoàn chỉnh với kích thước $|M| = 7$. Phép ghép $M = \{(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), (m_5, w_2), (m_6, w_7), (m_7, w_3), (m_8, w_5)\}$ là một phép ghép hoàn chỉnh với kích thước $|M| = 8$.

1.2.4. Các nghiên cứu liên quan

Trong những năm qua, bài toán MAX-SMTI đã nhận được nhiều sự chú ý từ các nhà nghiên cứu trong lĩnh vực Trí nhân tạo và Tính toán thông minh bởi những ứng dụng rộng rãi của nó trong thực tế như: bài toán phân công thực tập tại các doanh nghiệp (Hospitals/Residents problem) [2], bài toán sắp xếp chỗ ở cho sinh viên ký túc xá (Stable Roommate problem) [4, 5, 48], hay bài toán phân công giáo viên hướng dẫn đề tài cho sinh viên (Student-Project Allocation Problem) [3]. Phần này trình bày một số hướng nghiên cứu chính đã được đề xuất để giải quyết bài toán MAX-SMTI như được thống kê trong Bảng 1.6.

i) Thuật toán xấp xỉ: Một số thuật toán theo hướng tiếp cận xấp xỉ đã được đề xuất để giải quyết bài toán MAX-SMTI [75, 49, 52, 17, 57, 76, 2, 59, 61, 63, 77, 62]. Iwama và cộng sự [52] đã đề xuất một thuật toán xấp xỉ dựa trên hướng tiếp cận tìm kiếm cục bộ cho bài toán MAX-SMTI để đạt được tỷ lệ xấp xỉ là $(2 - c\frac{log(n)}{n})$, trong đó c hằng số dương tùy ý và n kích thước của thể hiện MAX-SMTI. Sau đó, Iwama và cộng sự [57, 76] đã cải tiến thuật toán của họ nhằm đạt được tỷ lệ xấp xỉ $(2 - c\frac{1}{\sqrt{n}})$ cho bài toán MAX-SMTI, trong đó c là hằng số dương tùy ý thỏa mãn $c \leq \frac{1}{4\sqrt{6}}$. Tiếp theo, các tác giả đã cải thiện các thuật toán trước đó của họ để đạt được tỷ lệ xấp xỉ là 1.875 [17]. Manlove và cộng sự [2] đã chứng minh rằng MAX-SMTI là một bài toán NP-khó, ngay cả khi danh sách xếp hạng chỉ có ưu tiên ngang bằng từ một phía người nam hoặc người nữ. Irving và cộng sự [75] đã giới thiệu một (p, q) MAX-SMTI, tức là danh sách xếp hạng của mỗi người nam có độ dài tối đa p và danh sách xếp hạng của tập các người nữ có độ dài tối đa q. Halldórsson và cộng sự [18] tiếp tục cải tiến các thuật toán đã đề xuất trong [49] cho bài toán MAX-SMTI với tỷ lệ xấp xỉ đạt $13/7 (< 1.858)$ nếu

Bảng 1.6: *Các nghiên cứu liên quan giải quyết bài toán* MAX-SMTI

Tác giả, năm xuất bản	Thuật toán	Mục tiêu
Thuật toán xấp xỉ		
Iwama và cộng sự [53], 1999 Manlove và cộng sự [8], 2002 Irving và cộng sự [55], 2009	Đề xuất xem xét bài toán SMTI, là một bài toán NP-khó	SMT, SMTI
Gent và Prosser [44, 56], 2001, 2002	Mô hình bài toán SMTI như bài toán CSP, Thuật toán sinh ngẫu nhiên các thể hiện SMTI	SMTI
Halldórsson và cộng sự [49], 2003	Thuật toán xấp xỉ	MAX-SMTI
Iwamavà cộng sự [52, 57, 17, 48], 2004, 2005, 2007, 2008	Thuật toán xấp xỉ	MAX-SMTI
Halldorsson và cộng sự [18], 2007	Cải tiến thuật toán xấp xỉ	MAX-SMTI
Irving và Manlove [58], 2008	Thuật toán xấp xỉ	MAX-SMTI
McDermid [59], 2009	Thuật toán xấp xỉ với tỷ lệ 3/2	MAX-SMTI
Daniel Marx và cộng sự [60] 2010	Thuật toán xấp xỉ FPT	SMT
Manlove and O'Malley [2], 2010	Mô hình bài toán SMTI từ bài toán (CSP)	SMTI
Paluch và cộng sự [61], 2012	Cải tiến thuật toán xấp xỉ với tỷ lệ 3/2	SMTI
Király [62], 2013 Paluch [63], 2014	Thuật toán xấp xỉ với tỷ lệ 3/2	MAX-SMTI
Adil và cộng sự [15], 2018	Thuật toán xấp xỉ được tham số hóa	MAX-SMTI
Adam Kunysz [12] 2019	Thuật toán xấp xỉ	SMTI
Hamada và cộng sự [64], 2019	Thuật toán xấp xỉ	MAX-SMTI
Lam và cộng sự [65], 2020	Thuật toán xấp xỉ	MAX-SMTI
Panda và cộng sự [66], 2022	Thuật toán xấp xỉ	MAX-SMTI
Thuật toán heuristic		
Gelain và cộng sự [11], 2010 Gelain và cộng sự [45], 2013	Thuật toán tìm kiếm cục bộ (LTIU)	MAX-SMTI
Munera và cộng sự [67], 2015	Thuật toán lập trình thích nghi (AS)	MAX-SMTI
Việt và cộng sự [23], 2016	Tối ưu cân bằng giới tính	SMT
Pettersson và cộng sự [68], 2021	Thuật toán tiền xử lý danh sách xếp hạng	SMTI
Haas và cộng sự [69], 2021	Thuật toán heuristic cải tiến chất lượng phép ghép ổn định	SMTI
Hướng tiếp cận khác		
Vohra và cộng sự [25], 2012	Stable matchings and linear programming	SMTI
Drummond và cộng sự [70], 2015	SAT	SMTI couples
Askalidis và cộng sự [71], 2018 Kwanashie và cộng sự [20], 2020	Áp dụng mô hình quy hoạch nguyên Áp dụng mô hình quy hoạch tuyến tính (ILP)	MAX-SMTI
Durmomod và cộng sự [70], 2020	Áp dụng mô hình SAT	MAX-SMTI
Erdem và cộng sự [72], 2020	Sử dụng phương pháp Answer Set Programming (ASP)	MAX-SMTI
Meeks và cộng sự [73], 2020	Giải quyết bài toán hôn nhân ổn định với nhóm các tác nhân	MAX-SMTI
Delorme và cộng sự [26], 2019	Áp dụng mô hình quy hoạch tuyến tính (ILP)	MAX-SMTI
Aziz và cộng sự [74], 2020	Uncertain Linear Preferences	SMTI

danh sách xếp hạng có giới hạn trên từ hai phía người nam và người nữ. McDermid [59] đã giới thiệu một thuật toán có tỷ lệ xấp xỉ là $3/2$ với thời gian thực hiện $O(n^{3/2}L)$, trong đó n là tổng số người nam và người nữ, và L là tổng độ dài của danh sách xếp hạng của chúng. Király [62] và Paluch [61, 63] đã cải tiến thuật toán được đề xuất bởi Gale và Shapley [1] để đạt được tỷ lệ xấp xỉ là $3/2$ với thời gian thực hiện là tuyến tính. Kiryaly và cộng sự [62] đã đề xuất một thuật toán, gọi là GSA2 để tìm một phép ghép ổn định với kích thước tối đa cho bài toán SMTI. Ý tưởng chính của thuật toán là dựa vào thuật toán GS [1] bằng cách cho phép mỗi người nam có thể khởi tạo lại danh sách xếp hạng của họ khi rỗng lần đầu tiên trong quá trình đề xuất người nữ trong danh sách. Quá trình khởi tạo danh sách này giúp cho việc tìm được phép ghép ổn định với kích thước tối đa hiệu quả hơn. Các thuật toán xấp xỉ đã đề xuất nhìn chung đã giải quyết khá tốt cho bài toán MAX-SMTI với chất lượng nghiệm tương đối tốt với tỷ lệ gần đúng tăng dần lên so với các thuật toán đề xuất trước đó. Hiện tại, thuật

toán xấp xỉ có tỷ lệ gần đúng tốt nhất là $3/2$ [77, 62, 63]. Tuy các thuật toán xấp xỉ đã giải quyết bài toán tương đối tốt về thời gian và chất lượng nghiệm, tuy nhiên chúng tôi nhận thấy rằng chất lượng nghiệm có thể được cải tiến tốt hơn. Ngoài ra, các thuật toán này không chỉ ra được các thực nghiệm chứng minh sự hiệu quả để giải quyết bài toán SMTI với kích thước lớn.

ii) Thuật toán heuristic: Một số phương pháp tìm kiếm heuristic [44, 78, 11, 45, 43] như lập trình ràng buộc (constraint programming) [14, 79], tìm kiếm cục bộ (local search) [11, 45] và lập trình tìm kiếm thích nghi (adaptive search) [43] đã được nghiên cứu để giải quyết bài toán MAX-SMTI với kích thước lớn. Gelain và cộng sự [11, 45] đã đề xuất một thuật toán tìm kiếm cục bộ, gọi là LTIU để giải bài toán MAX-SMTI. Ý tưởng chính của thuật toán này là bắt đầu từ một phép ghép ngẫu nhiên, thuật toán LTIU tìm một phép ghép lân cận (neighbors of matching) tốt nhất theo nghĩa tính giá trị (cost) của phép ghép so với tất cả các phép ghép lân cận mà nó sinh ra để di chuyển tiếp trong quá trình tìm phép ghép ổn định cho bài toán SMTI. Tuy nhiên, với phương pháp tiếp cận này, việc sinh ra các phép ghép lân cận cần $O(n^2)$ thời gian vì số lượng các phép ghép lân cận là rất lớn, vì vậy thuật toán LTIU không hiệu quả khi giải quyết bài toán SMTI kích thước lớn. Munera và cộng sự [14] đã mô hình hóa bài toán MAX-SMTI theo bài toán thỏa mãn ràng buộc trong đó mỗi người nam được xem là các biến (variables) và mỗi người nữ là miền giá trị (domains) của mỗi biến, các ràng buộc (constraints) là các điều kiện mà phép ghép không còn cặp chặn nào. Sau đó họ đã nghiên cứu phương pháp tìm kiếm thích nghi (adaptive search) [43] và đề xuất một thuật toán AS để giải quyết bài toán MAX-SMTI. Ý tưởng chính của AS là tại mỗi bước lặp thuật toán sẽ lựa chọn một biến có giá trị lỗi (error) cao nhất trong phép ghép hiện tại và thực hiện tạo một phép ghép mới bằng cách loại bỏ cặp chặn trong phép ghép hiện tại. Thuật toán AS sẽ đánh giá lại phép ghép hiện tại và kiểm tra có mắc kẹt cục bộ địa phương (local minimum) hay không? Nếu xảy ra trường hợp cục bộ địa phương thì AS sẽ gọi tới một hàm khởi tạo (resets) để thoát khỏi trường hợp mắc kẹt cục bộ địa phương. Thực nghiệm của họ đã chỉ ra rằng thuật toán AS hiệu quả hơn thuật toán xấp xỉ MD [59] và thuật toán tìm kiếm cục bộ LTIU [45] về thời gian và chất lượng nghiệm, tuy nhiên AS vẫn chưa hiệu quả khi giải quyết bài toán MAX-SMTI kích thước lớn.

iii) Các hướng tiếp cận khác: Ngoài ra, một số nhà nghiên cứu đã đề xuất phương pháp tiếp cận mới để giải bài toán SMTI như: sử dụng mô hình quy hoạch nguyên (integer programming - IP) [25, 71], sử dụng mô hình SAT [70], sử dụng đồ thị phân đôi (bipartite graph) [12], và sử dụng mô hình quy hoạch nguyên tuyến tính (ILP) [20, 26]. Gần đây, một

hướng tiếp cận khác [15, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89] đã được đề xuất để nghiên cứu bài toán hôn nhân ổn định.

Mặc dù, các thuật toán đề xuất trong những năm gần đây đã mở ra một hướng tiếp cận mới đề giải quyết bài toán MAX-SMTI, tuy nhiên các thuật toán này chưa thực sự hiệu quả cho bài toán SMTI kích thước lớn. Do đó, luận án tập trung nghiên cứu các thuật toán theo hướng tiếp cận tìm kiếm heuristic để giải quyết bài toán MAX-SMTI hiệu quả hơn về thời gian thực hiện và chất lượng nghiệm.

1.3. Một số bài toán mở rộng của bài toán SMTI

Bài toán SMTI là bài toán ghép cặp một-một (one-to-one matching) với điều kiện số người nam và số người nữ phải bằng nhau và do đó rất khó áp dụng vào các ứng dụng thực tế. Phần này sẽ trình bày các bài toán mở rộng của bài toán SMTI - là các bài toán ghép cặp một-nhiều (one-to-many matching) gồm Hospitals/Residents with Ties - HRT [54, 2] và Student-Project Allocation - SPA [60].

1.3.1. Bài toán Hospitals/Residents with Ties

Bài toán Hospitals/Residents with Ties problem, gọi tắt là HRT, là một biến thể của bài toán SMTI [2]. Bài toán HRT bao gồm một tập các bác sĩ thực tập (*residents*) và một tập các bệnh viện (*hospitals*). Mỗi bác sĩ thực tập xếp hạng các bệnh viện theo một thứ tự "*thích*" trong danh sách xếp hạng của họ và ngược lại. Mỗi bệnh viện có một số lượng tối đa các bác sĩ được nhận thực tập. Mục tiêu của bài toán là tìm một phép ghép ổn định giữa bác sĩ thực tập và bệnh viện sao cho tất cả các bác sĩ thực tập đều có thể nhận được vị trí thích hợp trong các bệnh viện mà không vi phạm các ràng buộc về số lượng tối đa các bác sĩ được nhận thực tập của bệnh viện. Để mở rộng tính ứng dụng của bài toán HRT trong các vấn đề hỗ trợ tìm kiếm việc làm cho sinh viên vừa mới ra trường nói chung và các bác sĩ thực tập nói riêng, luận án xem xét bài toán HRT dưới dạng một bài toán ứng dụng phân bổ sinh viên thực tập tới các doanh nghiệp và do vậy luận án xem các *sinh viên* như là các bác sĩ thực tập và các *doanh nghiệp* như là các bệnh viện.

Với danh sách xếp hạng có chứa ưu tiên ngang bằng (ties) trong của bài toán HRT, các định nghĩa của phép ghép ổn định bao gồm ổn định yếu (*weakly stable*), ổn định mạnh (*strongly stable*) và siêu ổn định (*super-stable*) [75, 10] tương tự như bài toán SMTI. Một phép ghép siêu ổn định là ổn định mạnh và một phép ghép ổn định mạnh là ổn định yếu.

Một phép ghép siêu ổn định nếu tồn tại thì tất cả các phép phép ổn định yếu có cùng kích thước [75]. Tuy nhiên, một thể hiện HRT có thể không tồn tại một phép ghép siêu ổn định. Irving và cộng sự [75] chỉ ra rằng một thể hiện HRT sẽ có các phép ghép ổn định yếu với kích thước khác nhau. Mục tiêu của bài toán HRT là tìm một phép ghép ổn định yếu sao cho tất cả các sinh viên đều có thể nhận được vị trí thực tập phù hợp tại các doanh nghiệp, gọi là bài toán MAX-HRT [75]. Với mỗi thể hiện của bài toán HRT đều tồn tại ít nhất một phép ghép ổn định yếu bằng cách phá vỡ thứ tự ưu tiên ngang bằng trong danh sách xếp hạng của cả sinh viên và doanh nghiệp. Tuy nhiên, bài toán MAX-HRT là một bài toán NP-khó [2]. Vì vậy, việc tìm ra các thuật toán để giải quyết bài toán MAX-HRT là thách thức cho các nhà nghiên cứu, do đó luận án tập trung nghiên cứu các thuật toán tìm kiếm heuristic để giải bài toán MAX-HRT. Các định nghĩa được trình bày sau đây là các định nghĩa liên quan tới phép ghép ổn định yếu. Để đơn giản, luận án xem xét một phép ghép ổn định yếu là phép ghép ổn định.

Định nghĩa 1.16 (Thể hiện HRT [75, 2, 10]). *Một thể hiện* HRT *kích thước $n \times m$ gồm một tập $\mathcal{R} = \{r_1, r_2, \cdots, r_n\}$ các sinh viên và một tập $\mathcal{H} = \{h_1, h_2, \cdots, h_m\}$ 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 không nghiêm ngặt, mỗi $h_j \in \mathcal{H}$ xếp hạng một tập con của $\mathcal{R}$ theo một thứ tự ưu tiên không nghiêm ngặt và mỗi $h_j \in \mathcal{H}$ có một số lượng tối đa $c_j \in \mathbb{Z}^+$ sinh viên có thể nhận thực tập.*

Ký hiệu $rank(r_i, h_j)$ thứ hạng của $h_j \in \mathcal{H}$ trong danh sách xếp hạng của $r_i \in \mathcal{R}$ và $rank(h_j, r_i)$ là thứ hạng của $r_i \in \mathcal{R}$ trong danh sách xếp hạng của $h_j \in \mathcal{H}$. Một cặp $(r_i, h_j) \in \mathcal{R} \times \mathcal{H}$ được gọi là chấp nhận nếu $rank(r_i, h_j) > 0$ và $rank(h_j, r_i) > 0$.

Định nghĩa 1.17 (Phép ghép [75, 2, 10]). *Một phép ghép M là một tập $M = \{(r_i, h_j) \in \mathcal{R} \times \mathcal{H}\}$ thỏa mãn các điều kiện sau:*

1. *$rank(r_i, h_j) > 0$ và $rank(h_j, r_i) > 0$, tức là (r_i, h_j) là một cặp chấp nhận;*
2. *$|M(r_i)| \leq 1$, tức là mỗi r_i chỉ được ghép tối đa một h_j;*
3. *$|M(h_j)| \leq c_j$, tức là mỗi h_j được ghép tối đa c_j sinh viên r_i.*

Nếu $(r_i, h_j) \in M$ thì ký hiệu $M(r_i) = h_j$ và $M(h_j) = \{r_k \in \mathcal{R} | M(r_k) = h_j\}$. Một $h_j \in \mathcal{H}$ được gọi là chưa đủ (*under-subscribed*), đủ (*full*) hoặc vượt quá (*over-subscribed*) số lượng $r_i \in \mathcal{R}$ nếu thỏa mãn tương ứng các điều kiện: $|M(h_j)| < c_j$, $|M(h_j)| = c_j$, hoặc $|M(h_j)| > c_j$.

Định nghĩa 1.18 (Cặp chặn [75, 2, 10]). *Một cặp $(r_i, h_j) \in \mathcal{R} \times \mathcal{H}$ là một cặp chặn cho một phép ghép M nếu thỏa mãn các điều kiện:*

1. $rank(r_i, h_j) > 0$ *và* $rank(h_j, r_i) > 0$*, tức là* (r_i, h_j) *là một cặp chấp nhận;*

2. $M(r_i) = \varnothing$ *hoặc* $rank(r_i, h_j) < rank(r_i, M(r_i))$*, tức là* r_i *hoặc là không được ghép hoặc là xếp hạng* h_j *ưu tiên hơn* $M(r_i)$*;*

3. $|M(h_j)| < c_j$ *hoặc* $rank(h_j, r_i) < rank(h_j, r_w)$*, trong đó* r_w *được* h_j *xếp hạng ưu tiên thấp nhất trong* $M(h_j)$*.*

Định nghĩa 1.19 (Phép ghép ổn định [75])**.** *Một phép ghép* M *là ổn định nếu không tồn tại bất kỳ cặp chặn* $(r_i, h_j) \in \mathcal{R} \times \mathcal{H}$ *cho* M*, ngược lại* M *được gọi là không ổn định.*

Định nghĩa 1.20 (Kích thước phép ghép [75])**.** *Kích thước của một phép ghép ổn định* M *là tổng số cặp* $(r_i, h_j) \in M$ *và được ký hiệu là* $|M|$*.*

Định nghĩa 1.21 (Phép ghép hoàn chỉnh [75])**.** *Một phép ghép ổn định* M *gọi là hoàn chỉnh nếu* $|M| = n$*, ngược lại* M *gọi là không hoàn chỉnh.*

Các định nghĩa khác như cặp chặn vượt trội và cặp chặn trội nhất tương tự như trong định nghĩa của bài toán SMTI [45].

Một thể hiện HRT gồm 8 sinh viên và 5 doanh nghiệp được mô tả trong Bảng 1.7. 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 ngang nhau. Các ký hiệu này cũng được sử dụng tương tự cho danh sách xếp hạng của các doanh nghiệp. Phép ghép

Bảng 1.7: *Ví dụ một thể hiện* HRT

Danh sách xếp hạng của $r_i \in \mathcal{R}$	Danh sách xếp hạng của $h_j \in \mathcal{H}$
r_1: h_1 h_3 h_2	h_1: 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: r_5 r_6 $(r_3$ $r_4)$ r_1
r_3: h_1 h_5 h_2	h_3: $(r_5$ $r_2)$ r_6 r_1 r_7
r_4: h_1 $(h_2$ $h_4)$	h_4: r_8 r_2 r_4 r_7
r_5: h_3 h_1 h_2	h_5: 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	
Số sinh viên tối đa của $h_j \in \mathcal{H}$: $c_1 = 2$, $c_2 = 3$, $c_3 = c_4 = c_5 = 1$	

$M = \{(r_1, h_2), (r_2, h_1), (r_3, h_1), (r_4, h_2), (r_5, h_3), (r_6, h_2), (r_7, h_5), (r_8, h_4)\}$ là phép ghép hoàn chỉnh, nghĩa là mọi sinh viên đều được phân công địa điểm thực tập tại các doanh nghiệp.

1.3.2. Bài toán Student-Project Allocation

Trong trường đại học các sinh viên phải thực hiện các đề tài do các giảng viên hướng dẫn trong quá trình học tập. Để thực hiện đề tài, các trường cần phân công giảng viên hướng dẫn sinh viên thực hiện đề tài. Tuy nhiên, trong một số trường hợp, việc phân công trực tiếp giảng viên hướng dẫn sinh viên sẽ không đáp ứng được nhu cầu thỏa mãn của giảng viên và sinh viên về yêu cầu chuyên môn, sở thích, v.v. Để giải quyết bài toán này, Manlove và Malley [60] đã mô hình hóa bài toán theo dạng bài toán ghép cặp và gọi là bài toán phân công giảng viên hướng dẫn sinh viên thực hiện đề tài (Student-Project Allocation problem - SPA) [3, 16]. Bài toán SPA là một biến thể mở rộng của bài toán HRT [1, 62] và mục tiêu của bài toán là tìm một phép ghép ổn định giữa các sinh viên và đề tài sao cho nhiều sinh viên nhận được đề tài nhất trên cơ sở dựa vào danh sách xếp hạng, gọi là bài toán MAX-SPA [3].

Một số biến thể của bài toán SPA đã được giới thiệu như: Student-Project Allocation problem with lecturer preferences over Students (SPA-S) [16]; Student-Project Allocation problem with lecturer preferences over Projects (SPA-P) [60, 19]; Student-Project Allocation problem with lecturer preferences over Students containing Ties (SPA-ST) [1, 3, 60, 90, 91]; và Student-Project Allocation problem with lecturer preferences over Student-Project pairs (SPA-(S,P)) [16, 92, 93].

Gần đây, các biến thể SPA-P và SPA-ST đã được cộng đồng nghiên cứu quan tâm nhiều trong các ứng dụng thực tế. Vì vậy, luận án trình bày lại các định nghĩa liên quan tới hai biến thể SPA-P và SPA-ST [20, 3, 19, 94].

Định nghĩa 1.22 (Thể hiện SPA-P [3])**.** *Một thể hiện* SPA-P *gồm một tập* $\mathcal{S} = \{s_1, s_2, \cdots, s_n\}$ *các sinh viên, một tập* $\mathcal{P} = \{p_1, p_2, \cdots, p_q\}$ *các đề tài, và một tập* $\mathcal{L} = \{l_1, l_2, \cdots, l_m\}$ *các giảng viên. Mỗi sinh viên* $s_i \in \mathcal{S}$ *xếp hạng một tập các đề tài* $A_i \subseteq \mathcal{P}$ *theo thứ tự ưu tiên trong danh sách xếp hạng. Mỗi giảng viên* $l_k \in \mathcal{L}$ *đề xuất một tập* $\mathcal{P}_k$ $(k = 1, 2, \cdots, m)$ *các đề tài được xếp hạng theo thứ tự ưu tiên trong danh sách xếp hạng. Mỗi giảng viên* $l_k \in \mathcal{L}$ *có một số lượng tối đa* $d_k \in \mathbb{Z}^+$ *sinh viên được ghép với* l_k*. Mỗi đề tài* $p_j \in \mathcal{P}$ *được đề xuất bởi một giảng viên và có một số lượng tối đa* $c_j \in \mathbb{Z}^+$ *sinh viên được ghép với* p_j*.*

Định nghĩa 1.23 (Thể hiện SPA-ST [3])**.** *Một thể hiện* SPA-ST *gồm một tập* $\mathcal{S} = \{s_1, s_2, \cdots, s_n\}$ *các sinh viên, một tập* $\mathcal{P} = \{p_1, p_2, \cdots, p_q\}$ *các đề tài, và một tập* $\mathcal{L} = \{l_1, l_2, \cdots, l_m\}$ *các giảng viên. Mỗi sinh viên* $s_i \in \mathcal{S}$ *xếp hạng một tập các đề tài* $A_i \subseteq \mathcal{P}$ *theo thứ tự ưu tiên trong danh sách xếp hạng. Mỗi giảng viên* $l_k \in \mathcal{L}$ *đề xuất một tập* $\mathcal{P}_k$ $(k = 1, 2, \cdots, m)$ *các đề tài và xếp hạng các sinh viên theo một thứ tự ưu tiên trong danh*

sách xếp hạng. Mỗi giảng viên $l_k \in \mathcal{L}$ có một số lượng tối đa $d_k \in \mathbb{Z}^+$ sinh viên được ghép với l_k. Mỗi đề tài $p_j \in \mathcal{P}$ được đề xuất bởi một giảng viên và có một số lượng tối đa $c_j \in \mathbb{Z}^+$ sinh viên được ghép với p_j.

Định nghĩa 1.24 (Phép gán [3]). *Một phép gán (assignment) là một tập $M = \{(s_i, p_j) \in \mathcal{S} \times \mathcal{P} | p_j \in A_i\}$. Nếu $l_k \in \mathcal{L}$ đề xuất $p_j \in \mathcal{P}$ và $(s_i, p_j) \in M$ thì có nghĩa s_i được gán cho p_j và l_k, p_j được gán cho s_i và l_k được gán cho s_i trong M.*

Với mọi $s_i \in \mathcal{S}$, nếu $(s_i, p_j) \in M$, ký hiệu là $M(s_i) = p_j$, ngược lại s_i không được gán trong M, ký hiệu $M(s_i) = \varnothing$. Với mọi $p_j \in \mathcal{P}$, ký hiệu $M(p_j) = \{s_i \in \mathcal{S} | M(s_i) = p_j\}$. Một đề tài $p_j \in \mathcal{P}$ là chưa đủ (*under-subscribed*), đủ (*full*) hoặc vượt quá (*over-subscribed*) số sinh viên tối đa được gán nếu thỏa mãn lần lượt các điều kiện: $|M(p_j)| < c_j$, $|M(p_j)| = c_j$, hoặc $|M(p_j)| > c_j$. Tương tự, với mọi $l_k \in \mathcal{L}$, ký hiệu $M(l_k) = \{s_i \in \mathcal{S} | M(s_i) = l_k\}$. Một giảng viên l_k là chưa đủ, đủ, hoặc vượt quá số sinh viên tối đa được gán nếu thỏa mãn lần lượt các điều kiện: $|M(l_k)| < d_k$, $|M(l_k)| = d_k$, hoặc $|M(l_k)| > d_k$.

Định nghĩa 1.25 (Phép ghép [3]). *Một phép ghép M là một phép gán thỏa mãn các điều kiện:*

1. *$|M(s_i)| \leq 1$ với mọi $s_i \in \mathcal{S}$;*
2. *$|M(p_j)| \leq c_j$ với mọi $p_j \in \mathcal{P}$ và $|M(l_k)| \leq d_k$ với mọi $l_k \in \mathcal{L}$, trong đó $l_k \in \mathcal{L}$ đề xuất $p_j \in \mathcal{P}$.*

Ký hiệu $rank(s_i, p_j)$ là thứ hạng của $p_j \in \mathcal{P}$ trong danh sách xếp hạng của $s_i \in \mathcal{S}$. Trong bài toán SPA-P, ký hiệu $rank(l_k, p_j)$ là thứ hạng của $p_j \in \mathcal{P}$ trong danh sách xếp hạng của $l_k \in \mathcal{L}$. Trong bài toán SPA-ST, ký hiệu $rank(l_k, s_i)$ là thứ hạng của $s_i \in \mathcal{S}$ trong danh sách xếp hạng của $l_k \in \mathcal{L}$.

Định nghĩa 1.26 (Cặp chặn của SPA-P [3]). *Một cặp $(s_i, p_j) \in \mathcal{S} \times \mathcal{P}$ là một cặp chặn cho một phép ghép M nếu thỏa mãn các điều kiện:*

1. *$p_j \in A_i$;*
2. *$M(s_i) = \varnothing$ hoặc $rank(s_i, p_j) < rank(s_i, M(s_i))$;*
3. *$|M(p_j)| < c_j$ và thỏa mãn một trong các điều kiện sau:*
   - *(a) $s_i \in M(l_k)$ và $rank(l_k, p_j) < rank(l_k, M(s_i))$, hoặc*
   - *(b) $s_i \notin M(l_k)$ và $|M(l_k)| < d_k$, hoặc*
   - *(c) $s_i \notin M(l_k), |M(l_k)| = d_k$ và $rank(l_k, p_j) < rank(l_k, p_t)$, trong p_t là đề tài được xếp hạng ưu tiên thấp nhất bởi l_k.*

Định nghĩa 1.27 (Cặp chặn của SPA-ST [3]). *Một cặp $(s_i, p_j) \in \mathcal{S} \times \mathcal{P}$ là một cặp chặn của một phép ghép M nếu thỏa mãn các điều kiện sau:*

1. $p_j \in A_i$;
2. $rank(s_i, p_j) < rank(s_i, M(s_i))$ *hoặc* $M(s_i) = \varnothing$;
3. *Thỏa mãn một trong ba điều kiện sau:*
   (a) $|M(p_j)| < c_j$ *và* $|M(l_k)| < d_k$;
   (b) $|M(p_j)| < c_j$, $|M(l_k)| = d_k$ *và (i)* $s_i \in M(l_k)$ *hoặc (ii)* $rank(l_k, s_i) < rank(l_k, s_t)$, *trong đó* s_t *là sinh viên được xếp hạng ưu tiên thấp nhất trong* $M(l_k)$.
   (c) $rank(l_k, s_i) < rank(l_k, s_z)$, *trong đó* s_z *là sinh viên được xếp hạng ưu tiên thấp nhất trong* $M(p_j)$.

Định nghĩa 1.28 (Phép ghép ổn định [3]). *Một phép ghép M là ổn định nếu không tồn tại bất kỳ cặp chặn $(s_i, p_j) \in \mathcal{S} \times \mathcal{P}$ cho M, ngược lại M được gọi là không ổn định.*

Định nghĩa 1.29 (Kích thước phép ghép [3]). *Kích thước của phép ghép ổn định M là tổng số cặp $(s_i, p_j) \in M$ và được ký hiệu là $|M|$.*

Định nghĩa 1.30 (Phép ghép hoàn chỉnh [3]). *Một phép ghép ổn định M gọi là hoàn chỉnh nếu $|M| = n$, ngược lại M được gọi là không hoàn chỉnh.*

Một thể hiện SPA-P gồm 5 sinh viên, 2 giảng viên và 5 đề tài được chỉ ra trong Bảng 1.8. Trong danh sách xếp hạng của sinh viên, ví dụ ký hiệu s_1: p_1 p_3 p_4 nghĩa là s_1 thích p_1 hơn p_3 và thích p_3 hơn p_4, tức là $rank(s_1, p_1) = 1$, $rank(s_1, p_3) = 2$ và $rank(s_1, p_4) = 3$. Trong danh sách xếp hạng của giảng viên, ví dụ ký hiệu l_1: p_1 p_2 p_3 nghĩa là l_1 đề xuất p_1, p_2 và p_3 và xếp hạng theo thứ tự ưu tiên là $rank(l_1, p_1) = 1$, $rank(l_1, p_2) = 2$ và $rank(l_1, p_3) = 3$. Ta có:

Bảng 1.8: *Ví dụ một thể hiện* SPA-P

Danh sách xếp hạng của $s_i \in \mathcal{S}$	Danh sách xếp hạng của $l_k \in \mathcal{L}$
s_1: p_1 p_3 p_4	l_1: p_1 p_2 p_3
s_2: p_5 p_1	l_2: p_4 p_5
s_3: p_2 p_5	
s_4: p_4 p_2	
s_5: p_5	
Số sinh viên tối đa của $p_j \in \mathcal{P}$: $c_1 = c_2 = c_3 = c_4 = 1, c_5 = 2$	
Số sinh viên tối đa của $l_k \in \mathcal{L}$: $d_1 = 3, d_2 = 2$	

1. Phép ghép $M = \{(s_1, p_3), (s_3, p_5), (s_4, p_2), (s_5, p_5)\}$ là một phép ghép không ổn định vì tồn tại các cặp chặn $\{(s_1, p_1), (s_2, p_1), (s_4, p_4)\}$ cho M.

2. Phép ghép $M = \{(s_1, p_1), (s_3, p_2), (s_4, p_4), (s_5, p_5)\}$ là một phép ghép ổn định với $|M| = 4$, trong đó s_2 chưa được ghép hay $M(s_2) = \varnothing$.

3. Phép ghép $M = \{(s_1, p_3), (s_2, p_1), (s_3, p_2), (s_4, p_4), (s_5, p_5)\}$ là một phép ghép hoàn chỉnh với $|M| = 5$.

Một thể hiện SPA-ST gồm 7 sinh viên, 3 giảng viên và 8 đề tài được chỉ ra trong Bảng 1.9. Trong danh sách xếp hạng của sinh viên và giảng viên, ký hiệu () thể hiện thứ tự xếp hạng ưu tiên ngang bằng. Ví dụ ký hiệu s_7: $(p_5\ p_3)\ p_8$ nghĩa là s_7 thích p_5 và p_3 hơn p_8, tức là $rank(s_7, p_5) = 1$, $rank(s_7, p_3) = 1$ và $rank(s_7, p_8) = 2$. Ví dụ ký hiệu l_3: $(s_1\ s_7)\ s_6$ nghĩa là l_3 thích s_1 và s_7 hơn s_6, tức là $rank(l_3, s_1) = 1$, $rank(l_3, s_7) = 1$ và $rank(l_3, s_6) = 2$.

Bảng 1.9: *Ví dụ một thể hiện* SPA-ST

Danh sách xếp hạng của $s_i \in \mathcal{S}$	Danh sách xếp hạng của $l_k \in \mathcal{L}$
s_1: $(p_1\ p_7)$	l_1: $(s_7\ s_4)\ s_1\ s_3\ (s_2\ s_5)\ s_6$
s_2: $p_1\ p_3\ p_5$	l_2: $s_3\ s_2\ s_7\ s_5$
s_3: $(p_2\ p_1)\ p_4$	l_3: $(s_1\ s_7)\ s_6$
s_4: p_2	
s_5: $p_1\ p_4$	l_1 đề xuất p_1, p_2, p_3
s_6: $p_2\ p_8$	l_2 đề xuất p_4, p_5, p_6
s_7: $(p_5\ p_3)\ p_8$	l_3 đề xuất p_7, p_8
Số sinh viên tối đa của $p_j \in \mathcal{P}$: $c_1 = 2, c_j = 1, (2 \leq j \leq 8)$	
Số sinh viên tối đa của $l_k \in \mathcal{L}$: $d_1 = 3, d_2 = 2, d_3 = 2$	

1. Phép ghép $M = \{(s_1, p_7), (s_3, p_4), (s_4, p_2), (s_5, p_1), (s_6, p_6), (s_7, p_8)\}$ là một phép ghép không ổn định vì tồn tại các cặp chặn $\{(s_2, p_1), (s_2, p_3), (s_3, p_1), (s_7, p_3)\}$ cho M.

2. Phép ghép $M = \{(s_1, p_7), (s_3, p_1), (s_4, p_2), (s_5, p_1), (s_6, p_8), (s_7, p_5)\}$ là một phép ghép ổn định với $|M| = 6$, trong đó s_2 chưa được ghép hay $M(s_2) = \varnothing$.

3. Phép ghép $M = \{(s_1, p_7), (s_2, p_1), (s_3, p_1), (s_4, p_2), (s_5, p_4), (s_6, p_8), (s_7, p_5)\}$ là một phép ghép hoàn chỉnh với $|M| = 7$.

1.3.3. Các nghiên cứu liên quan

Gần đây, các mở rộng của bài toán SMTI như HRT và SPA đã được cộng đồng nghiên cứu quan tâm bởi những ứng dụng quan trọng trong thực tế như: chương trình phân bố nhà ở cho nhân viên ở Scotland [30], dịch vụ quản lý phân bổ cư trú tại Canada, dự án phân bổ sinh viên ở Đại học Glasgow [20], Đại học Southern Denmark [95], và Đại học New York [96]. Một số nghiên cứu liên quan tới bài toán HRT và SPA được trình bày trong Bảng 1.10.

Một số thuật toán xấp xỉ đã được đề xuất để giải quyết bài toán MAX-HRT trong các

Bảng 1.10: *Các nghiên cứu liên quan của bài toán* HRT *và* SPA

Tác giả, năm xuất bản	Thuật toán	Mục tiêu
Bài toán HRT		
Irving và cộng sự [75], 2000	Mô hình hóa bài toán HRT	HRT
Irving và cộng sự [10], 2003	Xem xét các loại ổn định yếu, ổn định mạnh và siêu ổn định	HRT
Manlove và cộng sự [9], 2002	Thuật toán xấp xỉ tìm phép ghép ổn định mạnh	HRT
Kavitha và cộng sự [97], 2004	Sử dụng cấu trúc đồ thị phân đôi tìm phép ghép ổn định mạnh	HRT
Manlove và cộng sự [60], 2008	Thuật toán xấp xỉ cho bài toán MAX-HRT Chứng minh rằng MAX-HRT là bài toán NP-khó	MAX-HRT
Biro' và cộng sự [98], 2010	Thuật toán xấp xỉ, tìm phép ghép ổn định	HRT với ràng buộc
Kwanashie và cộng sự [20], 2013	Hướng tiếp cận quy hoạch nguyên	MAX-HRT
Munera và cộng sự [67], 2015	Sử dụng lập trình thích nghi	MAX-HRT
Hamada và cộng sự[64], 2016	Thuật toán xấp xỉ, tìm phép ghép ổn định	HRT với ràng buộc
Diebold và Bichler [99], 2017	Hướng tiếp cận quy hoạch nguyên	MAX-HRT
Bài toán SPA		
Abraham và cộng sự [3], 2003	Mô hình bài toán SPA Thuật toán xấp xỉ tìm phép phép ổn định yếu	MAX-SPA-S
Harper và cộng sự [100], 2005	Thuật toán di truyền tìm phép ghép ổn định	MAX-SPA-S
Pan và cộng sự [101], 2009	Sử dụng goal programming tìm phép ghép ổn định	MAX-SPA-S
Iwama và cộng sự [102], 2012	Mô hình bài toán SPA-P, Thuật toán xấp xỉ tìm phép ghép ổn định yếu	MAX-SPA-P
Kassa và cộng sự [103], 2013 Manlove và cộng sự [19], 2018	Mô hình IP tìm phép ghép ổn định yếu	MAX-SPA-P MAX-SPA-S
Cooper và Manlove [94], 2018	Thuật toán xấp xỉ tìm phép ghép ổn định yếu	MAX-SPA-ST
Chown và cộng sự [104], 2018	Sử dụng thuật toán luyện thép	MAX-SPA-P
Olaosebikan và cộng sự [13], 2020	Thuật toán xấp xỉ tìm phép ghép siêu ổn định	SPA-ST

tài liệu [17, 77, 9, 75]. Ki'raly [62] đã đề xuất thuật toán HP cho bài toán MAX-HRT với tỷ lệ xấp xỉ tốt nhất là $3/2$. Ý tưởng của thuật toán HP dựa vào thuật toán GSA2 cho bài toán MAX-SMTI. Ngoài ra một số độ phức tạp tham số của bài toán MAX-HRT cũng đã được nghiên cứu trong các bài báo [59, 61, 62]. Marx và cộng sự [50] đã đề xuất một thuật toán FPT với tham số là tổng độ dài của các ràng buộc trong danh sách xếp hạng cho bài toán HRT. Ngoài ra, một số hướng tiếp cận khác dựa trên tìm kiếm heuristic [79], quy hoạch nguyên [20, 99] đã được đề xuất để giải quyết bài toán MAX-HRT. Irving và cộng sự [10] đã đề xuất một thuật toán với thời gian thực hiện là $O(a^2)$ để tìm một phép ghép ổn định mạnh, trong đó a là số cặp giữa sinh viên và doanh nghiệp chấp nhận lẫn nhau. Tuy nhiên, các thuật toán này chỉ mới xem xét giải quyết tối ưu từ một phía (tối ưu cho sinh viên hoặc cho doanh nghiệp) mà chưa giải quyết được bài toán tối ưu từ hai phía, vì vậy các thuật toán này chưa thực sự hiệu quả và đáp ứng được các yêu cầu giải bài toán HRT trong thực tế. Manlove và cộng sự [9] đã đề xuất một thuật toán để tìm một phép phép ổn định mạnh cho bài toán HRT và chứng minh rằng kích thước lớn nhất của một phép ghép ổn định luôn lớn hơn 2 lần kích thước bé nhất của phép ghép ổn định trong mọi thể hiện của bài toán HRT. Irving và cộng sự [75] đã đề xuất hai thuật toán để tìm một phép ghép siêu ổn định cho bài toán HRT. Ý tưởng chính của thuật toán

là dựa trên thuật toán GS [1], tức là xem xét một chuỗi các đề xuất từ sinh viên hoặc doanh nghiệp để tạo ra một phép ghép tối ưu từ phía sinh viên hoặc từ phía doanh nghiệp. Gần đây, Kavitha và cộng sự [97] đã nghiên cứu một thuật toán với thời gian thực hiện là $O(nm)$ sử dụng cấu trúc đồ thị phân đôi (bipartite graph), trong đó n là số đỉnh và m số cạnh để tìm một phép ghép ổn định mạnh cho bài toán HRT. Kwanashie và cộng sự [20] đề xuất hướng tiếp cận quy hoạch nguyên để giải bài toán MAX-HRT. Munera và cộng sự [79] chuyển bài toán HRT thành bài toán SMTI và áp dụng thuật toán tìm kiếm thích nghi (AS) [43] để giải bài toán MAX-HRT. Diebold and Bichler [99] đã thực hiện một nghiên cứu thử nghiệm gồm 8 thuật toán cho bài toán HRT và so sánh các thuật toán này khi áp dụng cho các dữ liệu trong thực tế được lấy từ hệ thống phân bổ khóa học tại Đại học Kỹ thuật Munich. Các tác giả đã đo lường một số thuộc tính của các thuật toán, bao gồm cả kích thước của các phép ghép ổn định tìm được. Các thuật toán của họ đã trình bày để giải quyết bài toán MAX-HRT dựa trên mô hình ILP [105]. Ngoài ra, bài toán HRT còn được phát triển với nhiều ràng buộc như Hospitals/Residents Problem with Quota Lower Bounds, hay Hospitals/Residents Problem with Quota Upper Bounds [64]. Mục tiêu bài toán chú trọng vào tìm một phép ghép ổn định có thể chấp nhận (as stable as possible), tức là một phép ghép ổn định có số cặp chặn là nhỏ nhất. Hamada và cộng sự [64] đã chỉ ra rằng bài toán này là một bài toán NP-khó với tỷ lệ xấp xỉ là $(|H| + |R|)^{1-\epsilon}$, trong đó $\mathcal{H}$ và $\mathcal{R}$ là tập hợp các doanh nghiệp và sinh viên tương ứng. Biro' và cộng sự [98] đã xem xét bài toán HRT với giới hạn dưới (lower bounds). Trong mô hình này tác giả cho phép một số doanh nghiệp có thể đóng cửa vì không nhận được sinh viên. Họ cũng đã chứng minh rằng bài toán đặt ra liệu có khả thi khi giải pháp là NP-đầy đủ. Vì vậy, luận án tập trung nghiên cứu và đề xuất các thuật toán tìm kiếm heuristic để giải quyết bài toán MAX-HRT với kích thước lớn.

Tiếp theo, luận án xem xét một số nghiên cứu được đề xuất để giải quyết bài toán SPA như sau: Abraham và cộng sự [3] đã đề xuất một thuật toán $O(m)$ để tìm một phép ghép ổn định cho bài toán SPA-S. Thuật toán đã chỉ ra rằng với một thể hiện SPA, một phép ghép ổn định nhất định tồn tại. Ý tưởng chính của thuật toán là tạo ra phép ghép ổn định tối ưu cho sinh viên *(student-optimal)*, trong đó mỗi sinh viên tìm được đề tài tốt nhất của họ so với tất cả các phép ghép ổn định khác. Ngoài ra, họ cũng đã trình bày một thuật toán để tìm phép ghép ổn định tối ưu cho giảng viên *(lecturer-optimal)*, trong đó mỗi giảng viên có được nhóm sinh viên tốt nhất mà họ có thể có được trong bất kỳ phép ghép ổn định nào cho trước. Trong khi với bài toán SPA-P, một phép ghép ổn định có thể có các kích thước khác nhau, việc tìm được một phép ghép vừa ổn định vừa có kích thước tối đa là một bài toán NP-khó. Abu El-Atta và Moussa [92] đã chỉ ra rằng một phép ghép ổn định có thể được tìm thấy trong

thời gian $O(m)$, trong đó m là tổng độ dài danh sách xếp hạng của sinh viên.

Ngoài ra, một số phương pháp mới được đề xuất để giải quyết bài toán SPA-S và SPA-P như quy hoạch nguyên (*Integer Programming*-IP) [106, 103], lập trình ràng buộc (*Constraint Programming*-CP) [107, 108], lập trình mục tiêu (*Goal Programming*-GP) [101], và di truyền (*Genetic Algorithms*-GA) [100]. Saber và cộng sự [109] đã trình bày một mô hình IP cho SPA nhằm mục đích tối ưu hóa tổng thể sự hài lòng của sinh viên và giảng viên khi thực hiện các đề tài (tức là, giảm thiểu chi phí chung của cả hai bên). Anwar và cộng sự [106] đã giới thiệu một mô hình IP khác cho các bài toán SPA liên quan đến các đề tài cá nhân và nhóm. Manlove và cộng sự [19] đã mô tả một mô hình IP để tìm một phép ghép ổn định với kích thước tối đa. Manlove và O'Malley [60] đã đề xuất thuật toán SPA-P-approx và chứng minh rằng MAX-SPA-P là một bài toán NP-khó. Sau đó, Iwama và cộng sự [102] đã mô tả một thuật toán xấp xỉ, gọi là SPA-P-heuristic với tỷ lệ được cải tiến so với [60], ngoài ra họ cũng đã chỉ ra rằng MAX-SPA-P không thể gần đúng với tỷ lệ trong khoảng $21/9 - \epsilon$, với mọi $\epsilon > 0$, trừ khi P = NP. Gần đây, bài toán SPA được quan tâm theo nghĩa tìm một phép ghép ổn định yếu với kích thước tối đa hoặc phép ghép siêu ổn định. Cooper và Manlove [94] đã đề xuất một thuật toán xấp xỉ với tỷ lệ $3/2$, gọi là APX để tìm một phép ghép ổn định yếu với kích thước tối đa cho bài toán SPA-ST. Chown và cộng sự [104] đã sử dụng thuật toán mô phỏng luyện kim (*simulated annealing*) để tìm một phép ghép ổn định tối ưu cho sinh viên. Olaosebikan và cộng sự [13] đã đề xuất một thuật toán để tìm một phép ghép siêu ổn định. Họ đã chứng minh được rằng nếu một phép ghép siêu ổn định tồn tại thì tất cả các phép ghép ổn định yếu có kích thước bằng nhau, và trong thực tế thì một phép ghép siêu ổn định có thể không tồn tại.

1.4. Vấn đề tồn tại

Bài toán SMTI và các ứng dụng mở rộng đã được nhiều nhà nghiên cứu quan tâm bởi những ứng dụng quan trọng của nó trong thực tế. Với việc danh sách xếp hạng có thứ tự ưu tiên ngang bằng, một phép ghép sẽ có các tiêu chuẩn ổn định khác nhau như ổn định yếu, ổn định mạnh và ổn định siêu mạnh [53]. Một ghép ổn định yếu có thể có các kích thước khác nhau [28], trong khi một phép ghép siêu ổn định vừa là một phép ghép ổn định yếu và ổn định mạnh có kích thước bằng nhau [13]. Vì vậy, trong thực tế các ứng dụng, người ta thường quan tâm nhiều tới việc tìm một phép ghép ổn định yếu với kích thước tối đa - là bài toán NP-khó [9]. Tuy nhiên, các thuật toán đã đề xuất trong giai đoạn này chưa giải quyết hiệu quả về thời gian và chất lượng nghiệm cho mục tiêu tìm một phép ghép vừa ổn định nhưng phải

thỏa mãn ghép được nhiều người nhất với kích thước lớn. Đối với hướng tiếp cận heuristic, các thuật toán tuy đã giải quyết được bài toán với chất lượng nghiệm tương đối tốt, tuy nhiên chưa hiệu quả về thời gian thực hiện do phải xem xét tập các phép ghép lân cận quá lớn do tập các cặp chặn là quá lớn.

1.5. Định hướng nghiên cứu

Luận án tập trung nghiên cứu các thuật toán tìm kiếm heuristic để tìm một phép ghép ổn định với kích thước tối đa cho các biến thể SMTI, HRT và SPA, tức là giải quyết các bài toán MAX-SMTI, MAX-HRT và MAX-SPA.

- Nghiên cứu tổng quan bài toán SMTI và đề xuất các thuật toán tìm kiếm heuristic để giải quyết hiệu quả bài toán MAX-SMTI với kích thước lớn (*Công bố trong Chương 2*).
- Nghiên cứu tổng quan bài toán HRT và đề xuất các thuật toán tìm kiếm heuristic để giải quyết hiệu quả bài toán MAX-HRT với kích thước lớn (*Công bố trong Chương 3*).
- Nghiên cứu tổng quan bài toán SPA và đề xuất các thuật toán tìm kiếm heuristic để giải quyết hiệu quả bài toán MAX-SPA với kích thước lớn (*Công bố trong Chương 4*).

1.6. Phương pháp thực nghiệm và đánh giá

1.6.1. Bộ dữ liệu

Để tiến hành các thực nghiệm nhằm mục đích đánh giá chất lượng nghiệm và thời gian thực hiện của các thuật toán, luận án đã áp dụng phương pháp sinh bộ dữ liệu ngẫu nhiên do Gent và cộng sự [44] đề xuất cho bài toán SMTI được trình bày trong Thuật toán A.2 tại Phụ lục A. Với mỗi bộ dữ liệu, các tham số đầu vào cho các bài toán như sau:

- **Bài toán SMTI**: Luận án tạo ngẫu nhiên các thể hiện SMTI với 3 tham số (n, p_1, p_2), trong đó n là kích thước, p_1 là xác suất tạo danh sách xếp hạng không đầy đủ và p_2 là xác suất tạo các thứ tự ưu tiên ngang bằng. Chú ý rằng việc tạo các thể hiện SMTI trong đó danh sách xếp hạng của người nam và người nữ chỉ chứa các cặp chấp nhận.
- **Bài toán HRT**: Luận án tạo ngẫu nhiên các thể hiện HRT với 5 tham số (n, m, p_1, p_2, c_j), trong đó n là số sinh viên, m là số doanh nghiệp, p_1 là xác suất không đầy đủ, p_2 là xác suất của xếp hạng ưu tiên ngang bằng và c_j là số lượng sinh viên tối đa của doanh nghiệp $h_j \in \mathcal{H}$. Vì phép ghép ổn định của các thể hiện HRT bao gồm các cặp chấp nhận, nên luận án tạo ra các thể hiện HRT trong đó danh sách

xếp hạng của các sinh viên và doanh nghiệp chỉ có các cặp chấp nhận.

- **Bài toán SPA-P**: Luận án tạo ngẫu nhiên các thể hiện SPA-P với 6 tham số (n, m, q, p, c_j, d_k), trong đó n là số sinh viên, m là số giảng viên, q là số đề tài, p là xác suất không đầy đủ ($p = |A_i|/q$), c_j là số sinh viên tối đa của mỗi đề tài $p_j \in \mathcal{P}$ và d_k là số sinh viên tối đa của mỗi giảng viên $l_k \in \mathcal{L}$.
- **Bài toán SPA-ST**: Luận án tạo ngẫu nhiên các thể hiện SPA-ST với 7 tham số $(n, m, q, p_1, p_2, c_j, d_k)$, trong đó n là số sinh viên, m là số giảng viên, q là số đề tài, p_1 là xác suất tạo danh sách xếp hạng không đầy đủ, p_2 là xác suất tạo thứ tự xếp hạng ưu tiên ngang bằng, c_j là số sinh viên tối đa của mỗi đề tài $p_j \in \mathcal{P}$ và d_k là số sinh viên tối đa của mỗi giảng viên $l_k \in \mathcal{L}$.

1.6.2. Ngôn ngữ và cấu hình cài đặt

Các thuật toán đề xuất được mô phỏng bằng ngôn ngữ lập trình MATLAB2019a để tiến hành các thực nghiệm đánh giá các thuật toán đề xuất cho các bài toán SMTI, HRT và SPA. Luận án sử dụng một máy tính cá nhân với cấu hình CPU Core i7-8550U 1.8GHz và 16 GB RAM để chạy tất cả các thuật toán trong các thực nghiệm.

1.7. Kết luận chương 1

Bài toán SMTI là một biến thể quan trọng của bài toán hôn nhân ổn định được giới thiệu bởi Gale-Shaply năm 1962 [1]. Bài toán gồm một tập người nam và một tập người nữ với kích thước n. Mỗi người nam hoặc nữ xếp hạng một số người khác trong một danh sách xếp hạng có chứ xếp hạng ngang bằng. Mục tiêu của bài toán là tìm một phép ghép ổn định với kích thước tối đa, tức là có nhiều nhất số người nam được ghép tới người nữ (MAX-SMTI). Gần đây một số biến thể khác của bài toán SMTI đã được cộng đồng nghiên cứu quan tâm như HRT và SPA. Mặc dù, có nhiều thuật toán đã được đề xuất để giải quyết bài toán MAX-SMTI và các biến thể như thuật toán xấp xỉ, thuật toán heuristic và một số phương pháp khác. Tuy nhiên, việc tìm một phép ghép ổn định với kích thước tối đa là bài toán NP-khó, vì vậy thách thức của các nhà khoa học là tiếp tục nghiên cứu và đề xuất các thuật toán để giải quyết bài toán MAX-SMTI và các biến thể như HRT và SPA hiệu quả hơn về thời gian và chất lượng nghiệm. Chương này đã trình bày tổng quan các khái niệm cơ bản và phân tích tình hình nghiên cứu liên quan tới bài toán hôn nhân ổn định và các biến thể của bài toán hôn nhân ổn định. Các lý thuyết được trình bày trong chương này là cơ sở để luận án nghiên cứu các phương pháp giải quyết các bài toán trong các chương tiếp theo.

CHƯƠNG 2.
ĐỀ XUẤT THUẬT TOÁN GIẢI BÀI TOÁN MAX-SMTI

Chương này trình bày hai thuật toán heuristics để giải quyết bài toán MAX-SMTI, tức là tìm một phép ghép ổn định với kích thước tối đa cho các thể hiện của bài toán SMTI. Các thuật toán này đã được công bố tại các công trình [A.1] và [A.2] trong phần "*Các công bố sử dụng trong Luận án*"".

2.1. Giới thiệu

Bài toán SMTI là một biến thể quan trọng của bài toán SMP [1, 55, 53, 9]. Với các thứ tự ưu tiên được đưa ra trong danh sách xếp hạng, bài toán SMTI được xem xét ba tiêu chí ổn định của phép ghép gồm ổn định yếu, ổn định mạnh, và siêu ổn định [48, 9], trong đó phép ổn định yếu được chú ý nhiều nhất trong các ứng dụng thực tế [15, 11, 9, 14, 93]. Irving và cộng sự [110] đã chỉ ra rằng một phép ghép ổn định yếu của một thể hiện SMTI tìm được bằng cách áp dụng thuật toán GS [1] và chứng minh rằng một phép ghép ổn định yếu có thể có các kích thước khác nhau. Mục đích của bài toán là tìm một phép ghép ổn định yếu với kích thước tối đa, được gọi là bài toán MAX-SMTI [44, 9]. Với việc xuất hiện danh sách xếp hạng ngang bằng và không đầy đủ trong các thể hiện của SMTI, các thuật toán đã đề xuất cho các bài toán như SMP, SMT và SMI không hiệu quả để giải quyết bài toán MAX-SMTI. Bài toán MAX-SMTI là một bài toán NP-khó [53], do đó việc tìm ra một thuật toán hiệu quả để giải quyết bài toán MAX-SMTI là một thách thức đối với các nhà nghiên cứu. Nhiều thuật toán xấp xỉ [18, 52, 57, 76, 17, 18, 59] và thuật toán tìm kiếm cục bộ [45, 14] đã đề xuất để giải quyết bài toán MAX-SMTI. Tuy nhiên, các thuật toán đã đề xuất thường giải quyết với bài toán MAX-SMTI có kích thước nhỏ. Dựa vào tình hình các nghiên cứu bài toán MAX-SMTI, Chương này trình bày hai thuật toán tìm kiếm heuristic để giải quyết bài toán MAX-SMTI. Kết quả thực nghiệm chỉ ra hai thuật toán đề xuất hiệu quả về thời gian thực hiện và chất lượng nghiệm cho bài toán MAX-SMTI có kích thước lớn so với các hướng tiếp cận gần đây [11, 14, 62].

2.2. Đề xuất thuật toán MCS

2.2.1. Ý tưởng

Phần này đề xuất một thuật toán tìm kiếm heuristic dựa trên các xung đột tối đa (Max-Conflicts based heuristic search, viết tắt MCS) để giải quyết bài toán MAX-SMTI. Ý tưởng chính của MCS là bắt đầu từ một phép ghép ngẫu nhiên, thuật toán sẽ tìm một tập các cặp chặn vượt trội UBP và định nghĩa một hàm heuristic để chọn một cặp chặn UBP tốt nhất và loại bỏ khỏi phép ghép hiện tại. MCS lặp lại cho đến khi tìm được phép ghép hoàn chỉnh hoặc đạt đến số bước lặp tối đa.

2.2.2. Hàm heuristic

Cho một phép ghép không ổn định $M = \{(\cdots), (m_i, M(m_i)), (\cdots), (M(w_j), w_j), (\cdots), (M(w_k), w_k), (\cdots)\}$, trong đó chúng ta giả định rằng tồn tại hai cặp chặn (m_i, w_j), (m_i, w_k) và (m_i, w_j) vượt trội (m_i, w_k) theo xếp hạng ưu tiên của m_i. Nếu chúng ta xóa cặp chặn (m_i, w_j) từ M để thu được M', nghĩa là m_i được ghép với w_j, cả $M(m_i)$ và $M(w_j)$ trở thành độc thân và các cặp khác không thay đổi, tức là $M' = \{(\cdots), (m_i, w_j), (\cdots), (M(w_j), \varnothing), (\cdots), (M(w_k), w_k), (\varnothing, M(m_i))\}$, thì cặp chặn (m_i, w_k) của M' sẽ bị xóa. Điều này bởi vì nếu (m_i, w_k) là một cặp chặn cho M', thì $rank(m_i, w_k) < rank(m_i, w_j)$ và $rank(w_k, m_i) < rank(w_k, M'(m_k))$ và dẫn đến mâu thuẫn vì (m_i, w_j) vượt trội (m_i, w_k), tức là $rank(m_i, w_j) < rank(m_i, w_k)$. Do đó, khi (m_i, w_j) là một cặp chặn trội nhất (UBP) theo m_i của M và nếu chúng ta loại bỏ (m_i, w_j) thì tất cả các cặp chặn được tạo bởi m_i sẽ bị xóa trong phép ghép M'. Tương tự, nếu chúng ta xóa cặp chặn trội nhất (m_i, w_j) theo w_j thì tất cả các cặp chặn tạo bởi w_j sẽ bị xóa trong phép ghép M'.

Ký hiệu $X = \{(m_i, w_j) \mid m_i \in \mathcal{M}, w_j \in \mathcal{W}\}$ là một tập hợp các cặp chặn trội nhất theo xếp hạng ưu tiên của m_i cho một phép ghép không ổn định M. Theo đó, mỗi m_i xuất hiện một lần, trong khi w_j có thể xuất hiện nhiều lần trong X. Gọi $ubp(w_j)$ là số các cặp chặn được tạo bởi $w_j \in X$, thuật toán MCS định nghĩa một hàm heuristic như sau:

$$h(m_i) = n \times ubp(w_j) - rank(w_j, m_i), \forall (m_i, w_j) \in X. \tag{2.1}$$

Vì (m_i, w_j) là một cặp chặn trội nhất nên (m_i, w_j) là một cặp được chấp nhận tức là $0 < rank(w_j, m_i) \leq n$. Điều này có nghĩa là $h(m_i) \geq 0$ cho tất cả các cặp $(m_i, w_j) \in X$. Nếu một cặp (m_i, w_j) được chọn sao cho $h(m_i)$ là lớn nhất, nghĩa là $n \times ubp(w_j)$ là lớn nhất, trong khi $rank(w_j, m_i)$ là bé nhất. Hơn nữa, vì $ubp(w_j)$ được nhân với n, có nghĩa là w_j

sinh ra số lượng các cặp chặn nhiều nhất sẽ được chọn đầu tiên. Sau đó, trong số các cặp chặn $(m_i, w_j) \in X$ do w_j tạo ra, m_i được chọn sao cho $rank(w_j, m_i)$ là bé nhất. Bằng cách loại bỏ một cặp (m_i, w_j) như vậy, tất cả các cặp chặn do m_i tạo ra đều bị xóa. Hơn nữa, vì w_j thích m_i hơn những người khác trong tập các cặp chặn do w_j tạo ra, nếu chúng ta xóa (m_i, w_j), thì tất cả các cặp chặn (w_j, m_k), trong đó w_j thích m_i hơn m_k sẽ bị xóa, có nghĩa là chúng ta loại bỏ nhiều nhất cặp chặn tạo ra bởi w_j trong phép ghép hiện tại.

Mệnh đề 2.1. *Cho một phép ghép không ổn định M, một phép ghép ổn định sẽ đạt được sau một số bước hữu hạn xóa các cặp chặn trội nhất (m_i, w_j) của M ứng với giá trị lớn nhất của hàm $h(m_i)$.*

Chứng minh. Thật vậy, khi chúng ta xóa một cặp $(m_i, w_j) \in X$ của phép ghép M để nhận được phép ghép M', thì $M(w_j)$ và $M(m_i)$ trở thành độc thân trong M'.

i) Nếu $w_j \in \mathcal{W}$ là độc thân trong M, tức là $M(w_j) = \varnothing$, thì m_i được ghép với w_j và m_i không tạo ra bất kỳ cặp chặn nào cho M'.

ii) Nếu $w_j \in \mathcal{W}$ không độc thân trong M, thì $M(w_j)$ sẽ độc thân trong M' và giữ độc thân cho đến một phép ghép M'' nào đó, tức là $rank(M(w_j), \varnothing) = n + 1$, trong đó:

- Nếu $\nexists w_z \in M''$ sao cho $rank(w_z, M(w_j)) < rank(w_z, M''(w_z))$ thì $M(w_j)$ giữ nguyên và không tạo ra bất kỳ cặp chặn nào cho M''.
- Nếu $\exists w_z \in M''$ sao cho $rank(w_z, M(w_j)) < rank(w_z, M''(w_z))$, thì $M(w_j)$ và w_z tạo thành một cặp chặn cho M''. Sau khi xóa cặp chặn $(M(w_j), w_z)$ cho M'', $M(w_j)$ được ghép với w_z và tất cả cặp chặn được tạo bởi $M(w_j)$ sẽ bị xóa cho M''.

Do vậy, nếu tồn tại các cặp chặn được tạo bởi m_i cho một phép ghép M, nghĩa là có một cặp chặn trội nhất $(m_i, w_j) \in X$ cho M. Sau một số bước hữu hạn xóa các cặp chặn trội nhất, $M(w_j)$ sẽ trở thành độc thân hoặc được ghép với một $w_k \in \mathcal{W}$ và cả m_i và $M(w_j)$ không tạo ra bất kỳ cặp chặn nào. Điều này có nghĩa rằng một phép ghép ổn định sẽ đạt được sau một số bước hữu hạn xóa các cặp chặn trội nhất (m_i, w_j) của M ứng với giá trị lớn nhất của hàm $h(m_i)$. □

Ví dụ xét một thể hiện SMTI được chỉ ra trong Bảng 2.4. Giả sử rằng $M = \{(m_1, w_1)$, (m_2, w_6), (m_3, w_4), (m_4, w_8), $(m_5, \varnothing)$, (m_6, w_2), (m_7, w_7), $(m_8, \varnothing)$, $(\varnothing, w_3)$, $(\varnothing, w_5)\}$, thì $X = \{(m_2, w_5)$, (m_4, w_5), (m_5, w_3), (m_6, w_7), $(m_8, w_5)\}$. Do đó, chúng ta có $ubp(w_3) = 1$, $ubp(w_5) = 3$, $ubp(w_7) = 1$ và $h(m_2) = 21$, $h(m_4) = 22$, $h(m_5) = 5$, $h(m_6) = 7$, $h(m_8) =$

Bảng 2.1: *Ví dụ của một thể hiện* SMTI

Danh sách xếp hạng của $m_i \in \mathcal{M}$	Danh sách xếp hạng của $w_j \in \mathcal{W}$
m_1: w_1	w_1: m_1 $(m_5\ m_6)$
m_2: w_5 $(w_3\ w_4\ w_6)$ $(w_7\ w_8)$	w_2: $(m_3\ m_5\ m_6)$
m_3: w_4 $(w_2\ w_5)$	w_3: m_6 $(m_7\ m_8)$ m_5 m_2
m_4: $(w_5\ w_6)$ w_8 w_7	w_4: m_3 $(m_2\ m_6\ m_7)$ m_5
m_5: $(w_1\ w_3)$ $(w_4\ w_5)$ w_2	w_5: $(m_5\ m_7\ m_8)$ $(m_3\ m_4)$ m_2
m_6: $(w_4\ w_7)$ w_1 $(w_2\ w_3\ w_8)$	w_6: m_2 m_7 $(m_4\ m_8)$
m_7: w_4 w_6 $(w_3\ w_5\ w_7)$	w_7: $(m_2\ m_6)$ m_7 m_4
m_8: w_5 w_6 w_3	w_8: $(m_2\ m_4)$ m_6

Bảng 2.2: *Xóa cặp chặn* UBP *cho phép ghép* M

Xóa UBP	M'	X'
(m_2, w_5)	(m_1, w_1), (m_2, w_5), (m_3, w_4), (m_4, w_8), $(m_5, \varnothing)$, (m_6, w_2), (m_7, w_7), $(m_8, \varnothing)$, $(\varnothing, w_3)$, $(\varnothing, w_6)$	(m_4, w_5), (m_5, w_3), (m_6, w_7), (m_7, w_6), (m_8, w_5)
(m_4, w_5)	(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_5), $(m_5, \varnothing)$, (m_6, w_2), (m_7, w_7), $(m_8, \varnothing)$, $(\varnothing, w_3)$, $(\varnothing, w_8)$	(m_5, w_3), (m_6, w_7), (m_8, w_5)
(m_5, w_3)	(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), (m_5, w_3), (m_6, w_2), (m_7, w_7), $(m_8, \varnothing)$, $(\varnothing, w_5)$	(m_2, w_5), (m_4, w_5), (m_6, w_7), (m_8, w_5)
(m_6, w_7)	(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), $(m_5, \varnothing)$, (m_6, w_7), $(m_7, \varnothing)$, $(m_8, \varnothing)$, $(\varnothing, w_3)$, $(\varnothing, w_5)$, $(\varnothing, w_2)$	(m_2, w_5), (m_4, w_5), (m_5, w_3), (m_7, w_3), (m_8, w_5)
(m_8, w_5)	(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), $(m_5, \varnothing)$, (m_6, w_2), (m_7, w_7), (m_8, w_5), $(\varnothing, w_3)$	(m_5, w_3), (m_6, w_7)

23. Bảng 2.2 chỉ ra 5 trường hợp xóa cặp $(m_i, w_j) \in X$ để tạo ra phép ghép M' và một tập hợp X' của tập các cặp chặn trội nhất cho M'. Rõ ràng, nếu (m_8, w_5) được chọn để loại bỏ, tức là $h(m_8)$ là lớn nhất, thì sẽ đạt được M' có số UBP nhỏ nhất.

2.2.3. Mô tả thuật toán

Mục đích của bài toán MAX-SMTI là tìm một phép ghép ổn định có kích thước tối đa, nghĩa là cần xác định một hàm để đánh giá chất lượng của các phép ghép. Cho một phép ghép M, hàm đánh giá $f(M)$ được định nghĩa nếu M là một phép ghép ổn định thì $f(M)$ là số lượng độc thân trong M, ngược lại $f(M) = n$. Như vậy, một phép ghép ổn định M có kích thước lớn nhất sẽ có giá trị $f(M)$ nhỏ nhất và nếu M là hoàn chỉnh thì $f(M) = 0$. Thuật toán MCS được chỉ ra trong Thuật toán 2.1. MCS bắt đầu tìm một phép ghép ổn định với kích thước lớn nhất, M_{best}, từ một phép ghép được tạo ngẫu nhiên M (dòng 2-3). Ở mỗi bước lặp, MCS thực hiện như sau: Đầu tiên, MCS tìm một tập hợp X của các cặp chặn trội nhất cho M bằng Thuật toán 2.2 (dòng 7). Thứ hai, MCS kiểm tra nếu X là rỗng, tức là M

Algorithm 2.1: Thuật toán MCS

Input: - Thể hiện I của SMTI.
- Xác suất nhỏ, p.
- Bước lặp tối đa, max_iters.

Output: Phép ghép ổn định, M.

1. **function** `Main(`I`)`
2. Khởi tạo ngẫu nhiên M;
3. $M_{best} := M$;
4. $f_{best} := n$;
5. *iter* := 0;
6. **while** $(iter \leq max\_iters)$ **do**
7. $X :=$ `Find_UBPs(`M`)`;
8. **if** $(X = \emptyset)$ **then**
9. **if** $(f_{best} > f(M))$ **then**
10. $M_{best} := M$;
11. $f_{best} := f(M)$;
12. **if** $(f_{best} > 0)$ **then**
13. $M :=$ `Escape_Local_Minima(`M`)`;
14. continue;
15. **else**
16. break;
17. **for** (*mỗi* $m_i \in X$) **do**
18. $ubp(w_j) := ubp(w_j) + 1$, với $(m_i, w_j) \in X$;
19. **for** (*mỗi* $m_i \in X$) **do**
20. $h(m_i) := n * ubp(w_j) - rank(w_j, m_i)$, với $(m_i, w_j) \in X$;
21. **if** *(với xác suất nhỏ p)* **then**
22. $m_j :=$ một $m_i \in X$ ngẫu nhiên;
23. **else**
24. $m_j := \text{argmax}(h(m_i)), \forall m_i \in X$;
25. Xóa cặp chặn $(m_j, w_k) \in X$;
26. $iter := iter + 1$;
27. **return** M_{best};
28. **end function**

là một phép ghép ổn định, và nếu: (*i*) M_{best} có số lượng người độc thân nhiều hơn M thì M được gán cho M_{best} (dòng 9-11); (*ii*) M_{best} không phải là phép ghép hoàn chỉnh, tức là MCS bị kẹt ở điểm tối thiểu cục bộ, thì MCS gọi Thuật toán 2.3 để vượt qua điểm tối thiểu cục bộ và thực hiện vòng lặp tiếp theo (dòng 12-14), ngược lại, MCS trả về một phép ghép hoàn chỉnh M_{best}. Thứ ba, MCS đếm số lượng các UBP, $ubp(w_j)$, được tạo bởi mỗi $w_j \in X$ (dòng 17-18) và xác định các giá trị heuristic, $h(m_i)$, cho mọi $m_i \in X$ (dòng 19-20). Thứ tư, MCS lấy ngẫu nhiên một $m_j \in X$ với xác suất p nhỏ hoặc lấy một $m_j \in X$ tương ứng với giá trị lớn nhất $h(m_j)$ (dòng 21-24). Tuy nhiên, nếu tồn tại $m_k \in X$ mà $h(m_k) = h(m_j)$ thì MCS

Algorithm 2.2: Tìm tập các cặp chặn trội nhất, X

Input: Phép ghép M.
Output: Tập các cặp chặn trội nhất X của phép ghép M.

```
1.  function Find_UBPs(M)
2.      X := ∅;
3.      for (mỗi m_i ∈ 𝓜) do
4.          w_j := M(m_i);
5.          while (∃w_k ∈ 𝓦 | rank(m_i, w_k) > 0) do
6.              w_k := argmin(rank(m_i, w_k) > 0);
7.              if (rank(m_i, w_k) = rank(m_i, w_j)) then
8.                  break;
9.              if ((m_i, w_k) là một cặp chặn) then
10.                 X := X ∪ (m_i, w_k);
11.                 break;
12.             else
13.                 Xóa w_k trong danh sách xếp hạng của m_i;
14.     return X;
15. end function
```

chọn ngẫu nhiên m_j hoặc m_k. Cuối cùng, MCS loại bỏ UBP $(m_j, w_k) \in X)$ cho M để nhận được một phép ghép mới (dòng 25). MCS thực hiện lặp lại cho đến khi tìm được phép ghép hoàn chỉnh (dòng 16) hoặc đạt đến số bước lặp tối đa. Trong trường hợp thứ hai, MCS trả về phép ghép ổn định có kích thước tối đa hoặc phép ghép không ổn định. Chú ý rằng luôn tồn tại một phép ghép ổn định cho một thể hiện SMTI [110] và do đó, nếu MCS trả về một phép ghép không ổn định, thì chúng ta phải tăng số bước lặp tối đa để MCS trả về phép ghép không hoàn chỉnh hoặc hoàn chỉnh.

Hàm để xác định một tập hợp X của các cặp chặn trội nhất cho một phép ghép M được chỉ ra trong Thuật toán 2.2. Với mỗi $m_i \in \mathcal{M}$, hàm xem xét w_k có thứ hạng nhỏ nhất trong danh sách xếp hạng của m_i, tức là $rank(m_i, w_k) < rank(m_i, w_j)$ với mọi w_j. Nếu (m_i, w_k) tạo thành một UBP thì hàm sẽ thêm cặp (m_i, w_k) vào X, ngược lại hàm xem xét w_k tiếp theo trong danh sách xếp hạng của m_i. Chú ý rằng nếu $m_i \in \mathcal{M}$ đang độc thân trong M thì $M(m_i) = \varnothing$ và $rank(m_i, \varnothing) = n + 1$.

Trong cách tiếp cận này, MCS xem xét hai trường hợp mắc kẹt cục bộ. Thứ nhất, khi MCS mắc kẹt tại một phép ghép không ổn định, nó chọn một m_i ngẫu nhiên với một xác suất nhỏ p để vượt qua điểm mắc kẹt cục bộ. Thứ hai, khi MCS tìm được một phép ghép không hoàn chỉnh, nó thực hiện hàm chỉ ra trong Thuật toán 2.3 để tìm một phép ghép mới với kích thước lớn hơn. Cụ thể, trong một phép ghép không hoàn chỉnh, số người nam độc thân sẽ bằng số người nữ độc thân. Do đó, hàm kiểm tra nếu xác suất là $p \leq 0.5$, thì hàm sẽ lấy ngẫu

Algorithm 2.3: Vượt qua cục bộ địa phương

Input: Phép ghép M.
Output: Phép ghép M.

```
1.  function Escape_Local_Minima(M)
2.      if (với xác suất p ≤ 0.5) then
3.          U := {m_i | M(m_i) = ∅};
4.          Chọn ngẫu nhiên một m_j ∈ U;
5.          for (mỗi w_k ∈ danh sách xếp hạng của m_j) do
6.              if (M(w_k) ≠ ∅) then
7.                  Xóa cặp (M(w_k), w_k) thành hai người độc thân, M(w_k) và w_k;
8.      else
9.          V := {w_j | M(w_j) = ∅};
10.         Chọn ngẫu nhiên một w_j ∈ V;
11.         for (mỗi m_k ∈ danh sách xếp hạng của w_j) do
12.             if (M(m_k) ≠ ∅) then
13.                 Xóa cặp (m_k, M(m_k)) thành hai người độc thân, m_k và M(m_k);
14.     return M;
15. end function
```

nhiên một m_j độc thân (dòng 3-4). Ngược lại, hàm sẽ lấy ngẫu nhiên một w_j độc thân (dòng 9-10). Nếu m_j được chọn, hàm sẽ chọn mọi w_k trong danh sách xếp hạng của m_j và phá cặp $(M(w_k), w_k)$ thành hai người độc thân là $M(w_k)$ và w_k (dòng 5-7). Ngược lại, nếu w_j được chọn, hàm sẽ chọn mọi m_k trong danh sách xếp hạng của w_j và phá cặp $(m_k, M(m_k))$ thành hai người độc thân là m_k và $M(m_k)$ (dòng 11-13). Theo đó, MCS sẽ tạo ra một phép ghép mới, trong đó số lượng các cặp chặn cho phép ghép nhỏ hơn nhiều so với một phép ghép được tạo ngẫu nhiên và do vậy MCS tìm được phép ghép ổn định có kích thước tối đa nhanh hơn nhiều so với việc khởi động lại từ phép ghép được tạo ngẫu nhiên.

2.2.4. Ví dụ

Xét lại thể hiện SMTI gồm 8 nam và 8 nữ với danh sách xếp hạng được chỉ ra trong Bảng 2.1. Giả định rằng xác suất để chọn ngẫu nhiên một cặp chặn trội nhất của MCS là $p = 0$ và MCS bắt đầu từ phép ghép ngẫu nhiên M = $\{(m_1, w_1)$, (m_2, w_6), (m_3, w_4), (m_4, w_8), $(m_5, \varnothing)$, (m_6, w_2), (m_7, w_7), $(m_8, \varnothing)$, $(\varnothing, w_3)$, $(\varnothing, w_5)\}$, trong đó $f(M) = 8$. MCS gán M cho M_{best} và chạy các bước lặp được chỉ ra trong Bảng 2.3. Ở bước lặp đầu tiên, MCS tìm một tập hợp X của các cặp chặn trội nhất cho phép ghép M, đếm số cặp chặn được tạo bởi mỗi $w_j \in X$ và tính $h(m_i)$ cho mỗi $m_i \in X$. Vì $h(m_8) = 23$ có giá trị lớn nhất, nên MCS xóa cặp (m_8, w_5) trong M để tìm được một phép ghép mới ổn định M = $\{(m_1, w_1)$, (m_2, w_6), (m_3, w_4), (m_4, w_8), $(m_5, \varnothing)$, (m_6, w_2), (m_7, w_7), (m_8, w_5), $(\varnothing, w_3)\}$. MCS lặp lại cho đến

Bảng 2.3: *Ví dụ thực hiện của thuật toán* MCS *cho* MAX-SMTI

Lặp	M	X	$ubp(w_j \in X)$	$h(m_i \in X)$	(m_i, w_j)
1	(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), $(m_5, \varnothing)$, (m_6, w_2), (m_7, w_7), $(m_8, \varnothing)$, $(\varnothing, w_3)$, $(\varnothing, w_5)$	(m_2, w_5), (m_4, w_5), (m_5, w_3), (m_6, w_7), (m_8, w_5)	$ubp(w_3) = 1$ $ubp(w_5) = 3$ $ubp(w_7) = 1$	$h(m_2) = 21$ $h(m_4) = 22$ $h(m_5) = 5$ $h(m_6) = 7$ $h(m_8) = 23$	(m_8, w_5)
2	(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), $(m_5, \varnothing)$, (m_6, w_2), (m_7, w_7), (m_8, w_5), $(\varnothing, w_3)$	(m_5, w_3), (m_6, w_7)	$ubp(w_3) = 1$ $ubp(w_7) = 1$	$h(m_5) = 5$ $h(m_6) = 7$	(m_6, w_7)
3	(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), $(m_5, \varnothing)$, (m_6, w_7), $(m_7, \varnothing)$, (m_8, w_5), $(\varnothing, w_2)$, $(\varnothing, w_3)$	(m_5, w_3), (m_7, w_3)	$ubp(w_3) = 2$	$h(m_5) = 13$ $h(m_7) = 14$	(m_7, w_3)
4	(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), $(m_5, \varnothing)$, (m_6, w_7), (m_7, w_3), (m_8, w_5), $(\varnothing, w_2)$	(m_5, w_2)	$ubp(w_2) = 1$	$h(m_5) = 7$	(m_5, w_2)
5	(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), (m_5, w_2), (m_6, w_7), (m_7, w_3), (m_8, w_5)	$\{\}$			
6	MCS trả về M = (m_1,w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8),(m_5, w_2),(m_6, w_7), (m_7, w_3), (m_8, w_5) với $\|M\| = 8$				

bước lặp thứ 5, khi đó X rỗng và $f(M) = 0$. Vì $f_{best} > f(M)$, M được gán cho M_{best}, tức là $f_{best} = 0$, và do đó MCS trả về phép ghép hoàn chỉnh M_{best} = $\{(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), (m_5, w_2), (m_6, w_7), (m_7, w_3), (m_8, w_5)\}$.

2.2.5. Các kết quả thực nghiệm

Phần này trình bày các kết quả thực nghiệm để đánh giá hiệu quả của thuật toán MCS bằng cách so sánh thời gian thực hiện và chất lượng nghiệm của MCS với LTIU [11, 45] và thuật toán AS [14]. Cả hai thuật toán LTIU và AS được chọn để so sánh với MCS vì chúng là các thuật toán tìm kiếm cục bộ hiệu quả để giải quyết bài toán MAX-SMTI. Ngoài ra, luận án thực hiện các thực nghiệm để đánh giá hiệu quả của MCS với bài toán MAX-SMTI kích thước lớn.

Dữ liệu thực nghiệm là các thể hiện SMTI được tạo ngẫu nhiên với các tham số gồm n, $p_1 \in [0.1, 0.8]$ với bước tăng 0.1, $p_2 \in [0.0, 1.0]$ với bước tăng 0.1. Bởi vì MCS, LTIU và AS

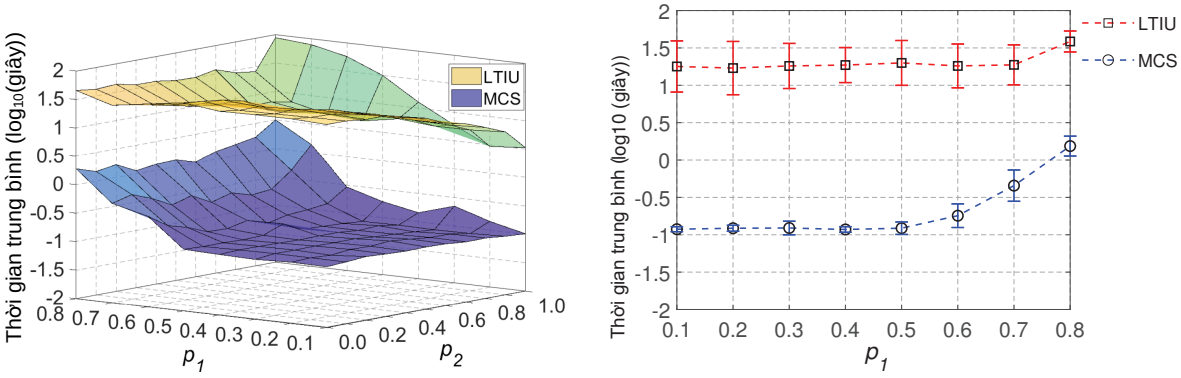
đều tìm phép ghép ổn định với kích thước tối đa từ một phép ghép ngẫu nhiên, với mỗi biến thể của SMTI, một phép ghép ngẫu nhiên được tạo ra để làm đầu vào cho cả ba thuật toán MCS, LTIU và AS. Xác suất cho MCS để chọn ngẫu nhiên một cặp chặn trội nhất là $p = 0.03$.

2.2.5.1. So sánh với thuật toán LTIU

Thực nghiệm 2.1. Đầu tiên, luận án thực hiện so sánh thời gian thực hiện và chất lượng nghiệm của MCS và LTIU. Với mỗi bộ tham số (p_1, p_2), luận án tạo ra 50 thể hiện SMTI với kích thước $n = 100$ và tính trung bình kết quả. Bước lặp tối đa cho cả hai thuật toán là 3000.

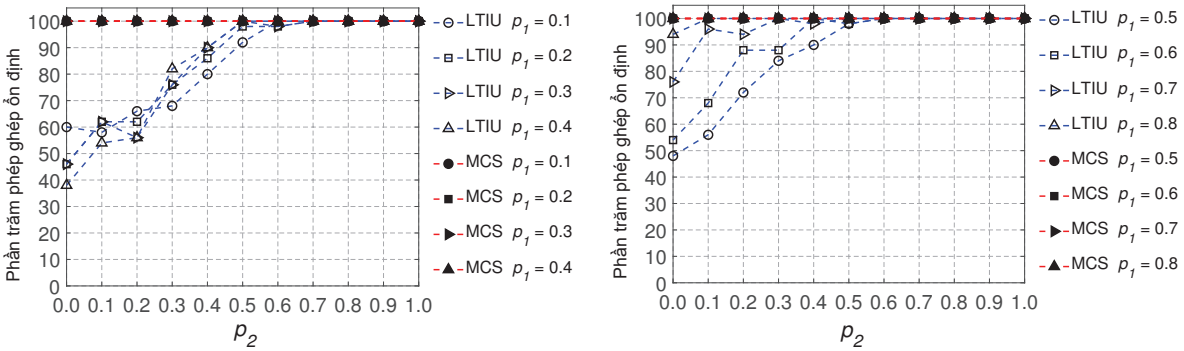
Hình 2.1(a) chỉ ra rằng thời gian thực hiện của MCS và LTIU để tìm phép ghép ổn định với kích thước tối đa. Thời gian thực hiện của cả MCS và LTIU tăng đáng kể khi p_1 tăng từ 0.1 lên 0.8, nhưng tăng nhẹ khi p_2 tăng từ 0.0 lên 1.0. Hơn nữa, thực nghiệm này chỉ ra rằng MCS chạy nhanh hơn LTIU khoảng 90 lần với giá p_1 thay đổi từ 0.1 đến 0.8, như được chỉ ra trong Hình 2.1(b) (do đó, hình vẽ sử dụng thang đo thời gian bằng log10). Những kết quả này có thể được giải thích như sau: Với SMTI có kích thước n, tại mỗi bước lặp LTIU cần $O(n^2)$ thời gian để xác định tập cặp chặn trội nhất cặp chặn trội nhất cho một phép ghép M. Bằng cách xóa từng cặp chặn trong tập hợp cặp chặn trội nhất để tạo các phép ghép lân cận (neighbours), LTIU sẽ đạt được tập hợp các phép ghép lân cận của phép ghép hiện tại. Vì chi phí của một phép ghép được tính cần $O(n^2)$ thời gian, LTIU cần $O(n^3)$ để xác định chi phí của các phép lân cận để tìm ra một phép ghép lân cận tốt nhất trong tập hợp các phép lân cận của phép ghép hiện tại. Rõ ràng, kích thước của các thể hiện SMTI càng lớn thì LTIU càng chạy chậm. Tuy nhiên ở mỗi bước lặp, MCS chỉ cần $O(n^2)$ thời gian để xác định tập cặp chặn trội nhất và chỉ xóa một cặp chặn trong tập cặp chặn trội nhất dựa trên hàm heuristic để tạo một phép ghép mới cho bước lặp tiếp theo. Điều này cho phép MCS chạy nhanh hơn nhiều so với LTIU.

Hình 2.2 chỉ ra phần trăm phép ghép ổn định tìm được của MCS và LTIU. MCS tìm được 100% các phép ghép ổn định, trong khi LTIU không phải lúc nào cũng tìm được các phép ghép ổn định, đặc biệt khi p_2 thay đổi từ 0.1 đến 0.6. Hình 2.3 chỉ ra phần trăm phép ghép hoàn chỉnh tìm được bởi MCS và LTIU. Khi p_1 thay đổi từ 0.1 đến 0.5, MCS luôn tìm được 100% phép ghép hoàn chỉnh. Khi p_1 thay đổi từ 0.6 đến 0.8, MCS tìm được phần trăm phép ghép hoàn chỉnh cao hơn so với LTIU. Điều này bởi vì khi phép ghép tìm được là không hoàn chỉnh, LTIU tạo một phép ghép ngẫu nhiên mới cho bước lặp tiếp theo. Do vậy nếu số bước lặp tối đa rất lớn, LTIU có thể tìm được các phép ghép hoàn chỉnh với tỷ lệ phần trăm



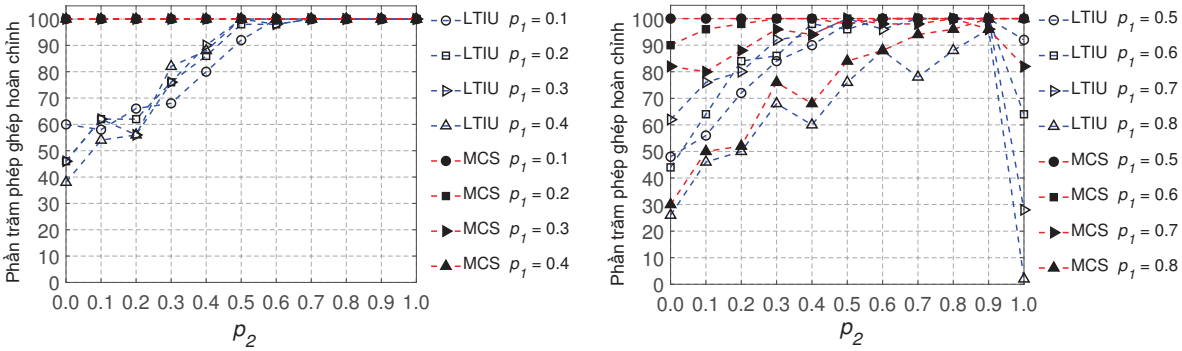
(a) MCS và LTIU- Thời gian thực hiện (b) MCS và LTIU với $p_2 \in [0.0, 1.0]$

Hình 2.1: *Thời gian thực hiện trung bình* MCS *và* LTIU



(a) MCS và LTIU với $p_1 \in [0.1, 0.4]$ (b) MCS và LTIU với $p_1 \in [0.5, 0.8]$

Hình 2.2: *Phần trăm phép ghép ổn định của* MCS *và* LTIU



(a) MCS và LTIU với $p_1 \in [0.1, 0.4]$ (b) MCS và LTIU với $p_1 \in [0.5, 0.8]$

Hình 2.3: *Phần trăm phép ghép hoàn chỉnh của* MCS *và* LTIU

cao hơn, nhưng cũng cần nhiều thời gian chạy hơn để tìm một phép ghép hoàn chỉnh. Tuy nhiên, trong các thực nghiệm này, số bước lặp lại tối đa là 3000, do đó, LTIU không tìm được tỷ lệ phần trăm phép ghép hoàn chỉnh cao hơn, nhưng với số bước lặp tối đa như vậy, MCS vẫn vượt trội hơn LTIU về thời gian thực hiện và chất lượng nghiệm.

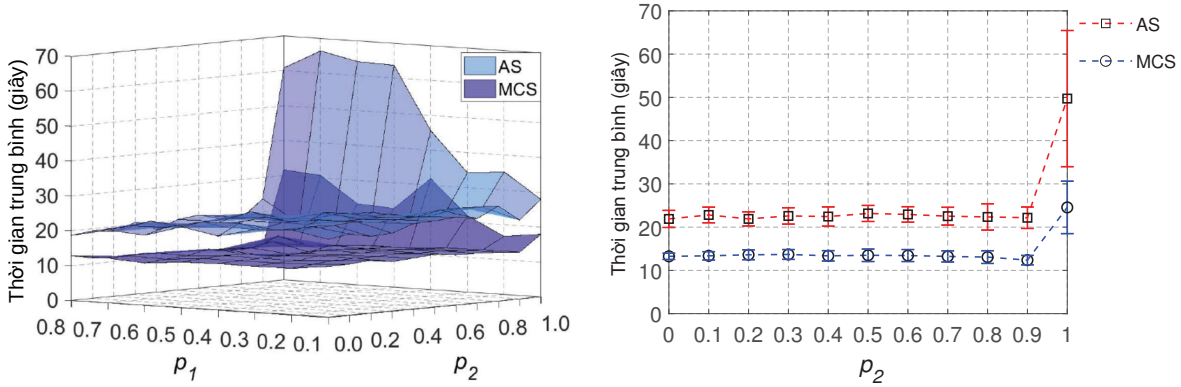
2.2.5.2. So sánh với thuật toán AS

Thực nghiệm 2.2. Trong phần này, với mỗi bộ giá trị tham số (p_1, p_2), luận án tạo ra 50 thể hiện SMTI với kích thước $n = 500$ và tính trung bình kết quả để so sánh MCS với AS. Bước lặp tối đa của hai thuật toán MCS và AS là 3000.

Hình 2.4 chỉ ra thời gian thực hiện trung bình của MCS và AS để tìm phép ghép ổn định với kích thước tối đa của các thể hiện SMTI. Khi p_2 tăng từ 0.0 đến 0.9, MCS chạy nhanh gấp 2 lần so với AS. Khi $p_2 = 1.0$, cả MCS và AS chạy chậm hơn so với khi p_2 tăng từ 0.0 đến 0.9. Điều này bởi vì tại mỗi bước lặp, AS cần $O(n^2)$ thời gian để xác định tập cặp chặn trội nhất và chi phí của một lần ghép. Sau khi loại bỏ một cặp chặn trội nhất tương ứng với giá trị hàm lỗi cao nhất trong tập cặp chặn trội nhất để có được phép ghép mới, AS phải tính chi phí của phép ghép mới để so sánh với phép ghép hiện tại để kiểm tra xem thuật toán đã gặp điểm tối thiểu cục bộ không. Do vậy AS cần $O(n^2)$ thời gian để tìm UBP và cộng thêm $O(n^2)$ thời gian để đánh giá chất lượng của hai phép ghép tại mỗi bước lặp. Ngược lại, như đã giải thích trước đây, MCS chỉ cần $O(n^2)$ thời gian để tìm một tập cặp chặn trội nhất cho một phép ghép, tức là MCS chạy nhanh hơn khoảng 2 lần so với AS.

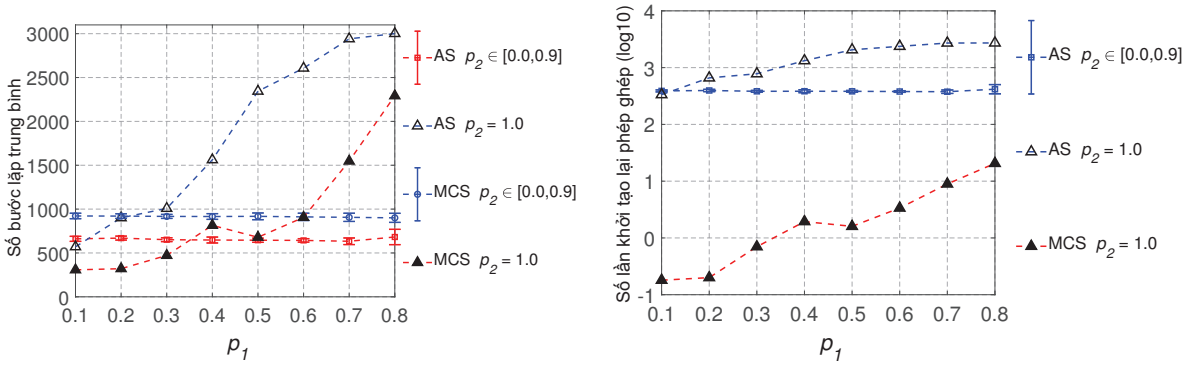
Hình 2.5(a) chỉ ra số bước lặp trung bình của MCS và AS để tìm phép ghép ổn định với kích thước tối đa của các thể hiện SMTI. Ngoại trừ $p_2 = 1.0$, AS cần ít hơn số bước lặp (khoảng 650 bước lặp) so với MCS (khoảng 850 bước lặp). Tuy nhiên, thời gian thực hiện trung bình của MCS nhỏ hơn thời gian thực hiện của AS, nghĩa là ở mỗi bước lặp AS mất nhiều thời gian tính toán hơn MCS. Hình 2.5(b) chỉ ra số lần trung bình gọi hàm để thoát khỏi điểm cục bộ trong MCS và AS (tức là số lần khởi tạo lại). Khi p_2 thay đổi từ 0.0 đến 0.9, MCS luôn vượt qua điểm cục bộ địa phương mà không cần gọi hàm khởi tạo. Khi $p_2 = 1.0$, MCS chỉ gọi hàm khởi tạo một vài lần. Ngược lại, AS gọi hàm khởi tạo khoảng 400 lần với mọi giá trị p_2. Đây cũng là lý do góp phần làm tăng thời gian thực hiện của AS so với MCS.

Tiếp theo, luận án so sánh chất lượng nghiệm tìm được của MCS và AS. Kết quả thực nghiệm chỉ ra rằng với mọi giá trị của các tham số (p_1, p_2), MCS luôn tìm được tất cả các phép ghép ổn định, trong khi AS vẫn tìm được một số phép ghép không ổn định của các thể hiện SMTI. Khi p_2 tăng từ 0.0 đến 0.9, MCS luôn tìm được tất cả các phép ghép hoàn chỉnh, nhưng AS vẫn tìm được một số phép ghép không hoàn chỉnh của các thể hiện SMTI. Khi $p_2 = 1.0$ và p_1 tăng từ 0.1 đến 0.7, MCS tìm được hơn 90% các phép ghép hoàn chỉnh như trong Hình 2.6(a) và các phép ghép khác là các phép ghép không hoàn chỉnh mà chỉ có một người độc thân duy nhất như được chỉ ra trong Hình 2.6(b). Ngược lại, AS tìm được ít hơn 90% các phép ghép hoàn chỉnh như được chỉ ra trong Hình 2.6(a) và các phép ghép khác là



(a) MCS và AS- Thời gian thực hiện (b) MCS và AS với $p_1 \in [0.1, 0.8]$

Hình 2.4: *Thời gian thực hiện trung bình của* MCS *và* AS



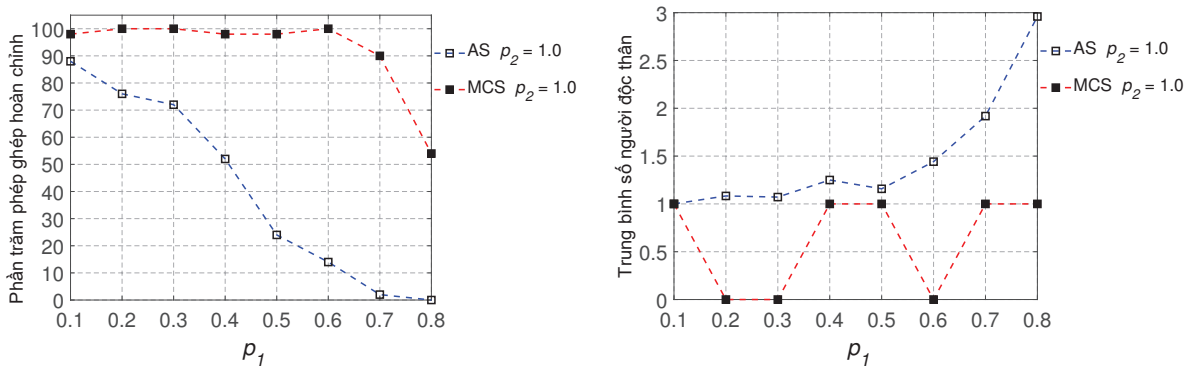
(a) Số bước lặp của MCS và AS (b) Số bước khởi tạo lại của MCS và AS

Hình 2.5: *Trung bình số bước lặp và khởi tạo lại của* MCS *và* AS

các phép ghép không hoàn chỉnh mà có nhiều hơn một người độc thân như được chỉ ra trong Hình 2.6(b). Đặc biệt, khi $p_1 = 0.8$ và $p_2 = 1.0$, AS không tìm được phép ghép hoàn chỉnh, trong khi MCS tìm được khoảng 55% phép ghép hoàn chỉnh như chỉ ra trong Hình 2.6(a). Kết quả thực nghiệm chỉ ra rằng mặc dù AS luôn đánh giá chi phí của phép ghép mới để so sánh với phép ghép trước đó để tránh bị mắc kẹt ở điểm cục bộ ở mỗi bước lặp, MCS vượt trội AS về tìm phép ghép hoàn chỉnh. Điều này bởi vì tại mỗi bước lặp, MCS luôn chọn một $w_j \in \mathcal{W}$ có số lượng tối đa các cặp chặn trội nhất và một $m_i \in \mathcal{M}$ trong danh sách xếp hạng của w_j sao cho w_j thích m_i hơn những người khác trong tập cặp chặn trội nhất. Sau đó, MCS loại bỏ cặp chặn trội nhất được tạo bởi (m_i, w_j). Do vậy MCS có thể xóa nhiều cặp chặn nhất cho một phép ghép để đạt được phép ghép mới không chỉ nhanh hơn AS mà còn tốt hơn so với phép ghép tìm được bởi AS.

2.2.5.3. Thực nghiệm với SMTI *kích thước lớn*

Thực nghiệm 2.3. Trong các thực nghiệm trên, LTIU và AS chạy tương đối chậm so với MCS khi kích thước bài toán là $n = 100$ cho LTIU và $n = 500$ cho AS. Do vậy, phần



(a) MCS và AS- % phép ghép hoàn chỉnh

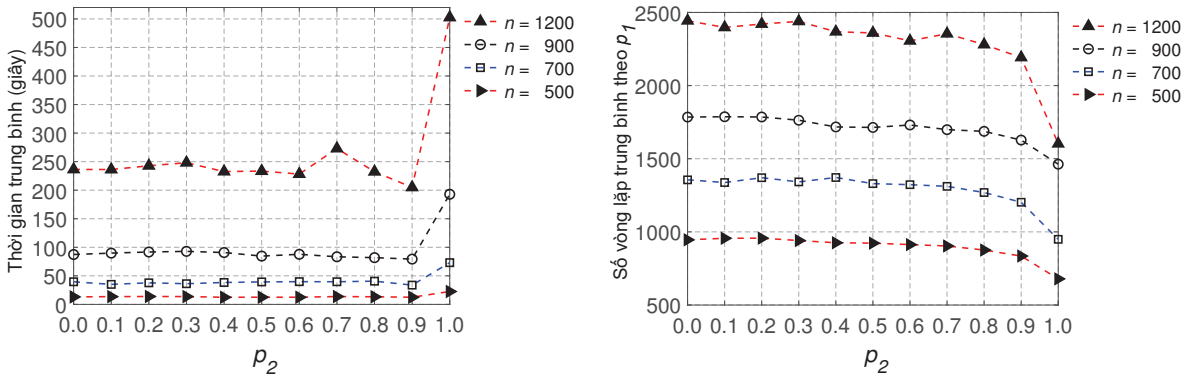
(b) MCS và AS- số người độc thân

Hình 2.6: *Chất lượng nghiệm của* MCS *và* AS *khi* $p_2 = 1$

này trình bày các kết quả thực nghiệm để đánh giá MCS cho các thể hiện SMTI kích thước lớn hơn. Cụ thể, thực nghiệm chạy với các thể hiện SMTI có kích thước n = 500, 700, 900 và 1200, trong đó $p_1 = 0.5$ và p_2 thay đổi từ 0.0 đến 1.0 với bước tăng 0.1. Với mỗi thể hiện SMTI của các tham số (n, p_1, p_2), luận án tạo ngẫu nhiên 50 thể hiện SMTI và tính trung bình kết quả. Số bước lặp lại tối đa của MCS là 5000.

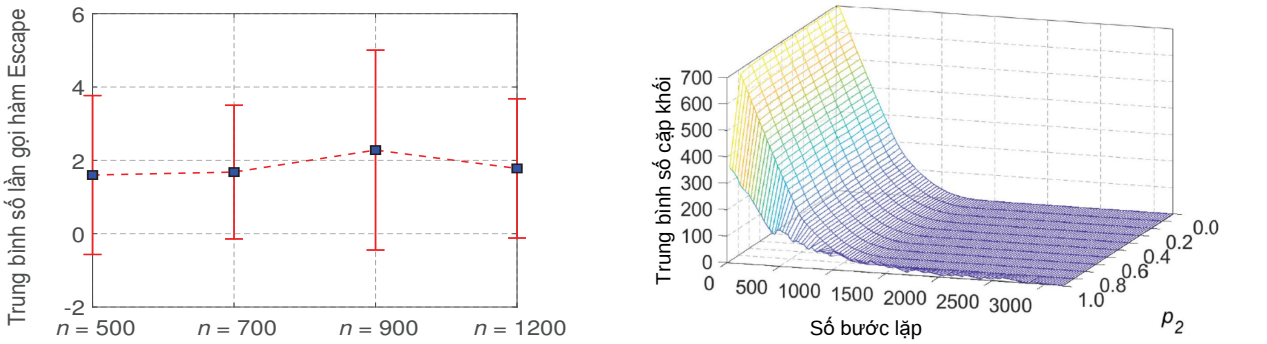
Hình 2.7(a) chỉ ra thời gian thực hiện trung bình của MCS. Với mỗi giá trị của n, thời gian thực hiện trung bình của MCS gần như giống nhau khi p_2 thay đổi từ 0.0 đến 0.9. Tuy nhiên, thời gian thực hiện trung bình của MCS tăng nhanh khi $p_2 = 1.0$, đặc biệt với $n = 1200$. Cần nhấn mạnh rằng khi kích thước của các thể hiện SMTI là 1200, SMTI có không gian tìm kiếm rất lớn ($1200! = 10^{3175}$) nhưng MCS chỉ cần khoảng 230 giây khi p_2 thay đổi từ 0.0 thành 0.9 và khoảng 500 giây khi $p_2 = 1.0$. Hình 2.7(b) chỉ ra số bước lặp trung bình của MCS. Số bước lặp lại trung bình của MCS giảm khi p_2 tăng. Hơn nữa, MCS không vượt quá 2500 bước lặp với $n = 1200$. Cần lưu ý rằng, như trong Hình 2.7(a), khi $p_2 = 1.0$, thời gian thực hiện trung bình của MCS tăng nhanh, đặc biệt với $n = 1200$, trong khi số bước lặp trung bình giảm. Điều này là vì trong MCS, tại mỗi bước lặp, độ phức tạp về thời gian để xác định tập cặp chặn trội nhất là $O(n^2)$, trong khi độ phức tạp về thời gian để xác định $ubp(w_j)$ và $h(m_i)$ là $O(n)$. Do đó, độ phức tạp về thời gian của MCS phụ thuộc nhiều vào độ phức tạp về thời gian của việc xác định tập cặp chặn trội nhất cho một phép ghép. Khi $p_2 = 1.0$, tức là các mức độ xếp hạng ưu tiên ngang bằng của tất cả những người nam có giá trị bằng 1. Do đó, để tìm các cặp chặn trội nhất cho một phép ghép, MCS phải xem xét tất cả w_k trong danh sách xếp hạng của mỗi $m_i \in \mathcal{M}$ để kiểm tra xem m_i có thích w_k hơn bạn ghép của w_j hay không. Điều này đã làm cho thời gian thực hiện của MCS tăng lên khi $p_2 = 1.0$.

Tiếp theo, luận án xem xét số lần MCS bị kẹt tại các điểm cục bộ trong các bước lặp.



(a) Trung bình thời gian (b) Trung bình số bước lặp

Hình 2.7: *Trung bình thời gian thực hiện và số bước lặp của* MCS



(a) Số lần gọi hàm Escape (b) Trung bình số cặp chặn trội nhất ($n = 700, p_1 = 0.5$)

Hình 2.8: *Trung bình số lần gọi hàm khởi tạo và số cặp chặn vượt trội*

Hình 2.8(a) chỉ ra rằng trong một số lượng nhỏ các bước lặp như trong Hình 2.7(b), MCS chỉ gọi vài lần hàm khởi tạo lại để vượt qua điểm mắc kẹt cục bộ. Hình 2.8(b) chỉ ra số lượng trung bình của UBP cho các phép ghép tìm được trong các bước lặp. Số lượng trung bình của UBP không vượt quá kích thước của SMTI. Khi p_2 thay đổi từ 0.0 đến 0.9, số lượng của UBP giảm khi số bước lặp tăng lên, nghĩa là MCS không bị kẹt tại bất kỳ điểm cực tiểu cục bộ nào trong khi tìm kiếm các phép ghép hoàn chỉnh cho các thể hiện SMTI. Tuy nhiên, khi p_2 = 1.0, số lượng UBP tăng nhanh ở một vài bước lặp, nghĩa là MCS bị mặc kẹt tại điểm cực tiểu cục bộ và MCS phải gọi hàm khởi tạo lại để tránh bị mắc kẹt trong các trường hợp. Tuy nhiên, số lượng UBP cho một phép ghép được tạo bởi hàm khởi tạo lại nhỏ hơn nhiều so với phép ghép được tạo ngẫu nhiên ở đầu MCS và trung bình số lần gọi hàm khởi tạo của MCS rất ít như được chỉ ra trong Hình 2.8(a). Những lý do này góp phần tăng nhanh quá trình tìm kiếm phép ghép hoàn chỉnh của các thể hiện SMTI kích thước lớn.

Cuối cùng, luận án xem xét chất lượng của các phép ghép mà MCS tìm được cho các thể hiện SMTI. Hình 2.3 chỉ ra rằng khi kích thước của các thể hiện SMTI là 100, MCS tìm được 100% phép ghép hoàn chỉnh chỉ với p_1 thay đổi từ 0.1 đến 0.5 và p_2 thay đổi từ 0.0 đến 1.0. Nhưng khi kích thước của các thể hiện SMTI là 500, MCS tìm được 100% của các phép

ghép hoàn chỉnh cho p_1 thay đổi từ 0.1 đến 0.8 và chỉ p_2 thay đổi từ 0.0 đến 0.9. Tuy nhiên, kết quả thực nghiệm cho các thể hiện SMTI có kích thước 700, 900 và 1200 chỉ ra rằng MCS luôn tìm được 100% phép ghép hoàn chỉnh. Điều này có nghĩa là MCS dễ dàng tìm được các phép ghép hoàn chỉnh cho các thể hiện SMTI có kích thước lớn. Điều này bởi vì nếu kích thước của các thể hiện SMTI nhỏ, khi giá trị của p_1 và p_2 tăng lên, danh sách xếp hạng của $m_i \in \mathcal{M}$ và $w_j \in \mathcal{W}$ trở nên thưa (chứa nhiều giá trị 0) và có nhiều thứ tự xếp hạng ngang bằng. Tuy nhiên, nếu kích thước của các thể hiện SMTI lớn, khi giá trị của p_1 và p_2 tăng lên, danh sách xếp hạng của $m_i \in \mathcal{M}$ và $w_j \in \mathcal{W}$ có nhiều thứ tự ưu tiên ngang bằng nhưng chúng không thưa như trong các thể hiện SMTI có kích thước nhỏ và do đó MCS dễ tìm được các phép ghép hoàn chỉnh hơn so với các thể hiện SMTI có kích thước nhỏ.

2.3. Đề xuất thuật toán HR

2.3.1. Ý tưởng

Thuật toán HR được đề xuất bao gồm thuật toán GS [1, 62] để tìm một phép ghép ổn định và một hàm heuristic để cải thiện kích thước của phép ghép tìm được bởi GS cho một thể hiện của SMTI. Cụ thể, nếu GS không tìm thấy một phép ghép hoàn chỉnh, HR sử dụng một hàm heuristic để hoán đổi bạn ghép của những người nam cho những người nam độc thân trong phép ghép hiện tại và tiếp tục áp dụng lại GS. HR kết thúc khi tìm được một phép ghép hoàn chỉnh hoặc đạt đến số bước lặp tối đa. Thực nghiệm chỉ ra rằng HR hiệu quả về thời gian thực hiện và chất lượng nghiệm cho bài toán MAX-SMTI kích thước lớn.

2.3.2. Mô tả thuật toán

Xét thuật toán GS [62] được mô tả chi tiết trong Thuật toán A.1 tại Phụ lục A cho một thể hiện I của SMTI. Thuật toán GS bắt đầu từ phép ghép rỗng. Tại mỗi bước lặp một người nam m_i độc thân có danh sách xếp hạng chưa rỗng sẽ đề xuất tới người nữ w_j có xếp hạng ưu tiên cao nhất. Nếu w_j chưa được ghép hoặc thích m_i hơn $M(w_j)$ thì w_j sẽ được ghép với m_i, ngược lại w_j từ chối m_i và m_i xoá w_j khỏi danh sách xếp hạng. Thuật toán kết thúc khi toàn bộ người nam được ghép hoặc những người nam chưa được ghép có danh sách xếp hạng rỗng. Giả sử rằng GS đưa ra một phép ghép ổn định M_1 và M_1 không hoàn chỉnh, nghĩa là có một $m_i \in \mathcal{M}$ trong đó $M_1(m_i) = \varnothing$ và danh sách xếp hạng ưu tiên của m_i = $\{\}$. Chúng ta xét hai trường hợp sau:

Trường hợp 1: Nếu khôi phục danh sách xếp hạng ban đầu cho m_i, thiết lập m_i hoạt

động (active) và chạy lại GS, thì GS trả về phép ghép M_2 giống như M_1. Điều này bởi vì: (*i*) nếu $m_k \neq m_i$ và m_k đã được ghép cho w_j trong M_1, thì m_k sẽ luôn giữ bạn ghép hiện tại w_j trong M_2 vì không tồn tại m_i nào mà $rank(w_j, m_i) < rank(w_j, m_k)$; (*ii*) nếu m_i độc thân trong M_1, thì m_i cũng sẽ độc thân trong M_2 do lần chạy đầu tiên trong thuật toán GS, m_i đã bị từ chối bởi mọi w_j trong danh sách xếp hạng của m_i, tức là mọi w_j trong danh sách xếp hạng của m_i đã được ghép cho một $m_k \in \mathcal{M}$ hoặc $rank(w_j, m_k) < rank(w_j, m_i)$ và vì thế w_j giữ bạn ghép hiện tại m_k và từ chối m_i ở lần chạy thứ hai trong thuật toán GS.

Trường hợp 2: Nếu khôi phục danh sách xếp hạng ban đầu cho m_i, gọi w_j là một trong những người nữ trong danh sách xếp hạng của m_i mà $rank(m_i, w_j) \leq rank(m_k, w_j)$ hoặc $rank(w_j, m_i) = rank(w_j, m_k)$, trong đó $m_k = M_1(w_j)$, và nếu: (*i*) đổi m_i cho m_k trong M_1, tức là: $M_1(m_k) = \varnothing$; $M_1(m_i) = w_j$; (*ii*) xóa w_j từ danh sách xếp hạng của m_k; (*iii*) thiết lập m_k hoạt động; (*iv*) chạy lại GS với đầu vào M_1, thì GS trả về M_2, trong đó $M_2(m_i) = w_j$ và m_k có thể được ghép với một người nữ trong danh sách xếp hạng của m_k, như vậy có nghĩa là $|M_2| > |M_1|$. Đây là ý tưởng của thuật toán HR để cải thiện kích thước của phép ghép ổn định.

Thuật toán HR được mô tả trong Thuật toán 2.4. HR sử dụng một thủ tục `repair`(m_i, m_k) để sửa trạng thái của $m_i \in \mathcal{M}$ và $m_k \in \mathcal{M}$, bao gồm các bước: (*i*) hoán đổi bạn ghép của m_k cho m_i trong phép ghép hiện tại, tức là: $M(m_i) := w_j$, trong đó $w_j = M(m_k)$; (*ii*) thiết lập m_k trở thành độc thân, tức là $M(m_k) := \varnothing$; (*iii*) thiết lập m_i không hoạt động, tức là $a(m_i) := 0$; và (*iv*) thiết lập m_k hoạt động trở lại, tức là $a(m_k) := 1$. Ban đầu HR tạo phép ghép M rỗng cho mỗi $m_i \in \mathcal{M}$, thiết lập m_i ở trạng thái hoạt động và gán một biến đếm $c(m_i) = 0$ (dòng 2-5). Ở mỗi bước lặp, nếu HR không tìm được $m_i \in \mathcal{M}$ đang hoạt động, thì HR sẽ cải thiện phép ghép M để có được một phép ghép có kích thước tốt hơn (dòng 9-13), ngược lại, HR sẽ chạy thuật toán GS để tìm phép ghép ổn định cho thể hiện SMTI (dòng 8, 18-30). Trong trường hợp đầu, HR kiểm tra nếu M là phép ghép hoàn chỉnh thì HR trả về M, nếu không, HR cải thiện $|M|$ bằng cách gọi Thuật toán 2.5 và thực hiện bước lặp tiếp theo. Trong trường hợp sau, HR kiểm tra nếu danh sách xếp hạng của m_i là rỗng (tức là $rank(m_j, w_j) = 0, \forall w_j \in \mathcal{W}$) thì HR thiết lập m_i không hoạt động, tăng biến đếm $c(m_i)$ và thực hiện bước lặp tiếp theo. Nếu không, m_i chọn một w_j độc thân mà m_i thích nhất. Nếu tồn tại một w_j như vậy, thì w_j được ghép cho m_i. Tuy nhiên, nếu không tồn tại w_j như vậy, nghĩa là w_j có một bạn ghép là m_k. Theo đó, w_j được ghép với m_i nếu m_k có một w_t độc thân mà $rank(m_k, w_t) = rank(m_k, w_j)$ hoặc w_j thích m_i hơn m_k. Nếu w_j được ghép với m_i thì m_i sẽ không hoạt động (tức là $a(m_i) = 0$), ngược lại, m_i sẽ xóa w_j khỏi danh

Algorithm 2.4: Thuật toán HR

Input: - Thể hiện I của SMTI.
- Bước lặp tối đa, max_iters.

Output: Phép ghép ổn định, M.

function `Main(`I`)`
  for (*mỗi* $m_i \in \mathcal{M}$) **do**
    $M(m_i) := \varnothing$;
    $a(m_i) := 1$; ▷ gán m_i hoạt động
    $c(m_i) := 0$; ▷ gán biến đếm cho m_i bằng 0
  $iter := 1$;
  while $iter \leq max\_iters$ **do**
    $m_i :=$ một người nam hoạt động, tức là $a(m_i) = 1$;
    if $\nexists a(m_i) = 1$ **then**
      if $|M| = n$ **then** break;
      $iter := iter + 1$;
      $M :=$ `Improve(`M`)`;
      continue;
    if $(\nexists w_j \in \mathcal{W} | rank(m_i, w_j) > 0)$ **then**
      $a(m_i) := 0$; ▷ gán m_i không hoạt động
      $c(m_i) := c(m_i) + 1$; ▷ tăng biến đếm của m_i
      continue;
    if *tồn tại một người nữ độc thân* w_j *người mà* m_i *thích nhất* **then**
      $M(m_i) := w_j$;
      $a(m_i) := 0$;
    else
      $w_j :=$ một người nữ người mà m_i thích nhất;
      $m_k := M(w_j)$;
      if *tồn tại một người nữ độc thân* w_t *mà* $rank(m_k, w_t) = rank(m_k, w_j)$ **then**
        `repair(`m_i, m_k`)`;
      if $M(m_i) = \varnothing$ *và* $rank(w_j, m_i) < rank(w_j, m_k)$ **then**
        `repair(`m_i, m_k`)`;
        $rank(m_k, w_j) := 0$;
      else
        $rank(m_i, w_j) := 0$;
  return M;
end function

sách xếp hạng của m_i (tức là $rank(m_i, w_j) = 0$). Nếu w_j loại bỏ m_k và được ghép với m_i, thì m_k được thiết lập hoạt động lại và m_k xóa w_j khỏi danh sách xếp hạng của m_k, trừ khi m_k có một w_t duy nhất mà $rank(m_k, w_t) = rank(m_k, w_j)$.

Hàm để cải thiện kích thước phép ghép M được chỉ ra trong Thuật toán 2.5. Với mỗi $m_i \in \mathcal{M}$ độc thân, tất cả w_j trong danh sách xếp hạng của m_i từ chối m_i, tức là danh sách xếp hạng của m_i rỗng, do đó m_i sẽ khôi phục danh sách xếp hạng ban đầu. Tiếp theo, m_i tìm một

Algorithm 2.5: Cải tiến kích thước của phép ghép ổn định M

Input: Phép ổn định M.
Output: Phép ghép M.

1. **function** `Improve(`M`)`
2. **for** *mỗi người nam* $m_i \in M$ *sao cho* $M(m_i) = 0$ **do**
3. Khôi phục lại danh sách xếp hạng của m_i;
4. $X := \{\}$;
5. **for** *mỗi* $w_j \in$ *danh sách xếp hạng của* m_i **do**
6. $m_k := M(w_j)$;
7. **if** $rank(m_i, w_j) \leq rank(m_k, w_j)$ ***hoặc*** $rank(w_j, m_i) = rank(w_j, m_k)$ **then**
8. $X := X \cup \{w_j\}$;
9. **if** $X = \emptyset$ **then** continue;
10. **for** *mỗi* $w_j \in X$ **do**
11. $m_k := M(w_j)$;
12. $k :=$ số lượng w_t sao cho $rank(m_k, w_t) = rank(m_k, w_j)$;
13. $h(w_j) := 1/k + (rank(w_j, m_i) - rank(w_j, m_k)) \times (1 - c(m_k))$;
14. $w_j := argmin(h(w_j)), \forall w_j \in X$;
15. `repair(`m_i, m_k`)`, với $m_k := M(w_j)$;
16. $rank(m_k, w_j) := 0$;
17. **return** M;
18. **end function**

tập X gồm các w_j trong danh sách xếp hạng của m_i sao cho $rank(m_i, w_j) \leq rank(m_k, w_j)$ hoặc $rank(w_j, m_i) = rank(w_j, m_k)$, trong đó $m_k = M(w_j)$ (dòng 5-8). Nếu không tồn tại w_j, thì hàm sẽ tiếp tục xét $m_i \in \mathcal{M}$ độc thân tiếp theo. Ngược lại, với mỗi $w_j \in X$ xác định giá trị heuristic $h(w_j)$ theo công thức 2.2:

$$h(w_j) = 1/k + (rank(w_j, m_i) - rank(w_j, m_k)) \times (1 - c(m_k)). \tag{2.2}$$

Sau đó, một $w_j \in \mathcal{W}$ tương ứng với giá trị nhỏ nhất của $h(w_j)$ được chọn để ghép cho m_i và $m_k = M(w_j)$ sẽ xóa w_j khỏi danh sách xếp hạng của m_k. Bằng cách như vậy, m_k có cơ hội được ghép cho w_t khác trong danh sách xếp hạng của m_k trong các bước lặp tiếp theo của HR. Chú ý rằng một $w_j \in \mathcal{W}$ được chọn sao cho $h(w_j)$ nhỏ nhất, nghĩa là: (*i*) m_k có nhiều nhất số w_t mà $rank(m_k, w_t) = rank(m_k, w_j)$; (*ii*) w_j xếp hạng m_i gần với m_k nhất; và (*iii*) $c(m_k)$ nhỏ nhất.

2.3.3. Ví dụ

Xét một thể hiện SMTI gồm 8 nam và 8 nữ với danh sách xếp hạng trong Bảng 2.4, thuật toán HR thực hiện các bước như sau:

Bảng 2.4: *Ví dụ một thể hiện* SMTI

Danh sách xếp hạng của $m_i \in \mathcal{M}$	Danh sách xếp hạng của $w_j \in \mathcal{W}$
m_1: w_3 w_8 w_5 w_2 $(w_1$ $w_7)$	w_1: m_8 m_1 m_5 m_7
m_2: w_5	w_2: m_5 $(m_1$ $m_8)$ m_3
m_3: w_8 $(w_2$ w_3 $w_7)$ w_5 w_4	w_3: m_1 $(m_4$ m_7 $m_8)$ m_3
m_4: w_8 w_5 w_3	w_4: $(m_3$ $m_8)$
m_5: $(w_1$ w_2 $w_7)$	w_5: $(m_1$ $m_3)$ m_8 m_4 m_2
m_6: $(w_6$ $w_8)$	w_6: m_8 m_6
m_7: w_1 w_3 w_8 w_7	w_7: m_5 $(m_3$ $m_7)$ m_1 m_8
m_8: $(w_1$ $w_4)$ $(w_7$ $w_8)$ $(w_2$ w_3 w_5 $w_6)$	w_8: m_8 m_7 m_6 m_1 $(m_3$ $m_4)$

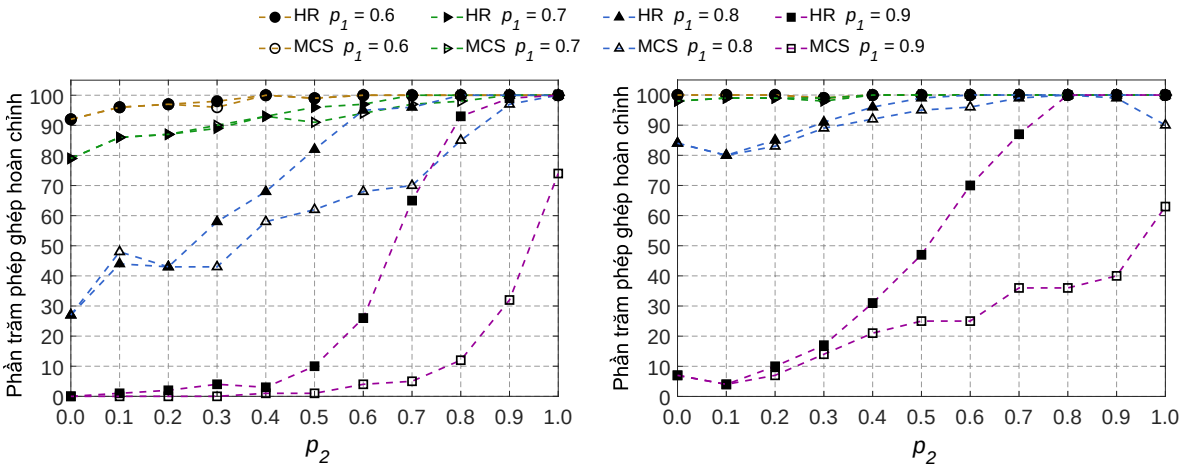
Bước 1: HR thực hiện thuật toán GS lần một và tìm được một phép ghép ổn định $M = \{(m_1, w_3), (m_2, \varnothing), (m_3, w_8), (m_4, w_5), (m_5, w_2), (m_6, w_6), (m_7, w_1), (m_8, w_4)\}$ sau 11 bước lặp. Tại bước lặp thứ 12, vì không tồn tại $m_i \in \mathcal{M}$ hoạt động và $|M| = 7$, hàm `improve`(M) được gọi để cải thiện kích thước $|M|$. Cụ thể, vì m_2 độc thân, m_2 khôi phục danh sách xếp hạng ban đầu. Tiếp theo, m_2 tìm w_5 để tạo thành một bạn ghép, vì w_5 có một bạn ghép m_4, w_5 từ chối m_4 để ghép với m_2 và m_4 xóa w_5 trong danh sách xếp hạng. Vì vậy, hàm trả về $M = \{(m_1, w_3), (m_2, w_5), (m_3, w_8), (m_4, \varnothing), (m_5, w_2), (m_6, w_6), (m_7, w_1), (m_8, w_4)\}$.

Bước 2: HR thực hiện thuật toán GS lần hai và tìm được một phép ghép ổn định $M = \{(m_1, w_3), (m_2, w_5), (m_3, w_8), (m_4, \varnothing), (m_5, w_2), (m_6, w_6), (m_7, w_1), (m_8, w_4)\}$ tại bước lặp thứ 14. Tại bước lặp 15, vì không tồn tại $m_i \in \mathcal{M}$ hoạt động và $|M| = 7$, hàm `improve`(M) được gọi để cải thiện $|M|$. Cụ thể, vì m_4 độc thân, m_4 khôi phục danh sách xếp hạng ban đầu. Tiếp theo, m_4 tìm w_8 để tạo thành một bạn ghép, vì w_8 có một bạn ghép m_3, w_8 từ chối m_3 để ghép với m_4 và m_3 xóa w_8 trong danh sách xếp hạng. Vì vậy, hàm trả về $M = \{(m_1, w_3), (m_2, w_5), (m_3, \varnothing), (m_4, w_8), (m_5, w_2), (m_6, w_6), (m_7, w_1), (m_8, w_4)\}$.

Bước 3: HR thực hiện thuật toán GS lần ba và tìm được một phép ghép hoàn chỉnh $M = \{(m_1, w_3), (m_2, w_5), (m_3, w_7), (m_4, w_8), (m_5, w_2), (m_6, w_6), (m_7, w_1), (m_8, w_4)\}$ với kích thước 8 tại bước lặp 17.

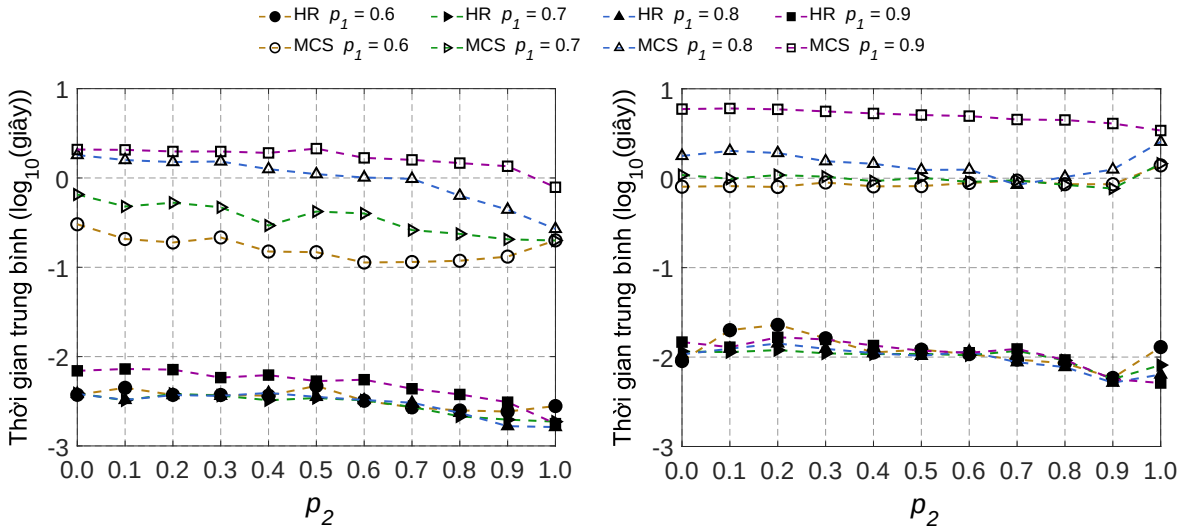
2.3.4. Các kết quả thực nghiệm

Phần này trình bày các kết quả thực nghiệm để so sánh hiệu quả của thuật toán HR với hiệu quả của các thuật toán MCS và GSA2 [62] cho bài toán MAX-SMTI.



(a) $n = 100$ (b) $n = 200$

Hình 2.9: *Phần trăm phép ghép hoàn chỉnh của* HR *và* MCS



(a) $n = 100$ (b) $n = 200$

Hình 2.10: *Thời gian thực hiện của của* HR *và* MCS *với* $n \in \{100, 200\}$

2.3.4.1. So sánh với thuật toán MCS

Phần này trình bày các kết quả thực nghiệm để so sánh chất lượng nghiệm và thời gian thực hiện của thuật toán HR với chất lượng nghiệm và thời gian thực hiện của thuật toán MCS được đề xuất trong mục 2.2.

Thực nghiệm 2.4. Trong thực nghiệm này các giá trị của tham số của các thể hiện SMTI được chọn gồm: $n \in \{100, 200\}$, $p_1 \in \{0.1, 0.2, \cdots, 0.9\}$ và $p_2 \in \{0.0, 0.1, \cdots, 1.0\}$. Với mỗi sự kết hợp của các giá trị của các tham số (n, p_1, p_2), luận án tạo 100 thể hiện SMTI ngẫu nhiên, thực hiện HR và MCS trên các thể hiện SMTI đã tạo.

Hình 2.9 chỉ ra phần trăm phép ghép hoàn chỉnh tìm được của HR và MCS. Chú ý rằng khi $p_1 \in \{0.1, 0.2, \cdots, 0.5\}$ và $p_2 \in \{0.0, 0.1, \cdots, 1.0\}$, cả HR và MCS tìm được 100% phép ghép hoàn chỉnh, do đó kết quả không được đưa ra trong hình này. Chúng ta dễ thấy

rằng HR vượt trội MCS về số lượng các phép ghép hoàn chỉnh tìm được cho mọi giá trị n, p_1 và p_2.

Hình 2.10 chỉ ra thời gian thực hiện trung bình của HR và MCS. Khi p_1 tăng, thời gian thực hiện của HR và MCS tăng, tuy nhiên khi p_2 tăng, thời gian thực hiện của HR và MCS giảm không đáng kể. Khi $n = 100$, thời gian thực hiện của HR tăng từ $10^{-2.7887}$ giây đến $10^{-2.1384}$ giây, trong khi thời gian thực hiện của MCS tăng từ $10^{-0.9463}$ giây đến $10^{0.3287}$ giây, tức là HR chạy nhanh gấp khoảng từ 70 đến 290 lần so với MCS khi p_1 tăng từ 0.6 đến 0.9. Khi $n = 200$, thời gian thực hiện của HR tăng từ $10^{-2.2903}$ giây đến $10^{-1.6394}$ giây, trong khi thời gian thực hiện của MCS tăng từ $10^{-0.1155}$ giây đến $10^{0.7802}$ giây, tức là HR chạy nhanh gấp khoảng từ 150 lần đến 260 lần so với MCS khi p_1 tăng từ 0.6 đến 0.9.

Từ kết quả thực nghiệm này, chúng ta thấy rằng HR không chỉ vượt trội MCS về chất lượng nghiệm tìm được mà còn hiệu quả hơn nhiều MCS về thời gian thực hiện.

2.3.4.2. So sánh với thuật toán GSA2

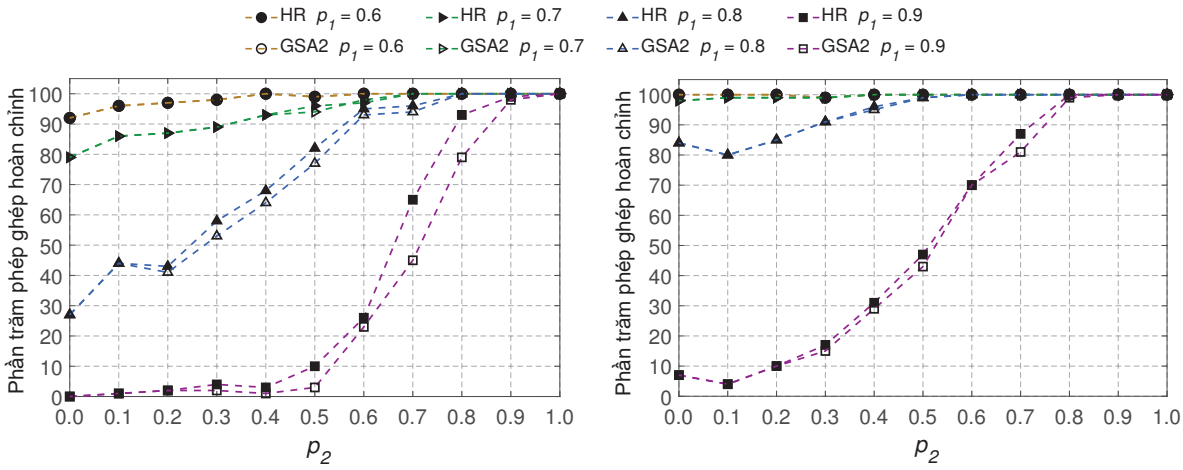
Phần này trình bày các kết quả thực nghiệm để so sánh chất lượng nghiệm và thời gian thực hiện của thuật toán HR với chất lượng nghiệm và thời gian thực hiện của thuật toán GSA2 [62]. GSA2 được chọn để so sánh với HR vì cả HR và GSA2 được cải tiến dựa trên GS [62].

Thực nghiệm 2.5. Trong thực nghiệm này, bộ dữ liệu đã tạo trong Thực nghiệm 2.4 được chọn để so sánh HR với GSA2. Hình 2.11 chỉ ra phần trăm phép ghép hoàn chỉnh tìm được của HR với GSA2. Giống như Thực nghiệm 2.4, khi $p_1 \in \{0.1, 0.2, \cdots, 0.5\}$ và $p_2 \in \{0.0, 0.1, \cdots, 1.0\}$, cả HR và GSA2 tìm được 100% phép ghép hoàn chỉnh, do đó kết quả không được chỉ ra trong hình này. Từ các kết quả của thực nghiệm này, chúng ta có thể thấy:

1) Khi n tăng, phần trăm phép ghép hoàn chỉnh tìm được bởi HR và GSA2 tăng vì mỗi người xếp hạng ưu tiên những người khác giới nhiều hơn. Hơn nữa, phần trăm phép ghép hoàn chỉnh tìm được của HR cao hơn của GSA2.

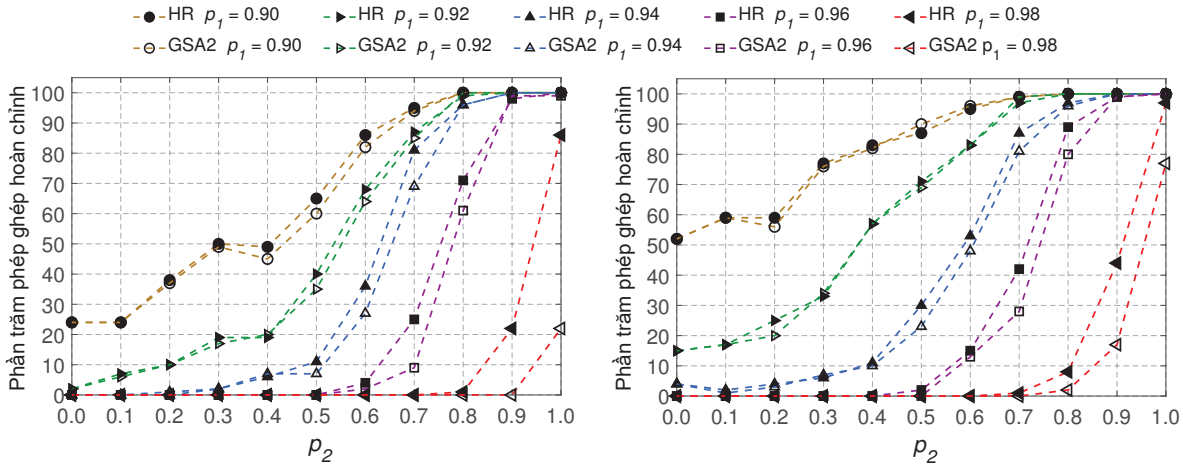
2) Khi p_1 tăng, tỷ lệ phần trăm phép ghép hoàn chỉnh tìm được bởi HR và GSA2 giảm vì số lượng các cặp được chấp nhận trong danh sách ưu tiên của mỗi $m_i \in \mathcal{M}$ và mỗi $w_j \in \mathcal{W}$ giảm, gây khó hơn cho việc tìm kiếm các phép ghép hoàn chỉnh.

3) Khi p_2 tăng, tỷ lệ phần trăm phép ghép hoàn chỉnh tìm được bởi HR và GSA2 tăng vì cùng một giá trị p_1, số lượng ưu tiên ngang bằng trong danh sách xếp hạng của mỗi $m_i \in \mathcal{M}$ và mỗi $w_j \in \mathcal{W}$ tăng, giúp dễ hơn cho tìm kiếm các phép ghép hoàn chỉnh.



(a) $n = 100$ (b) $n = 200$

Hình 2.11: *Phần trăm phép ghép hoàn chỉnh của* HR *và* GSA2

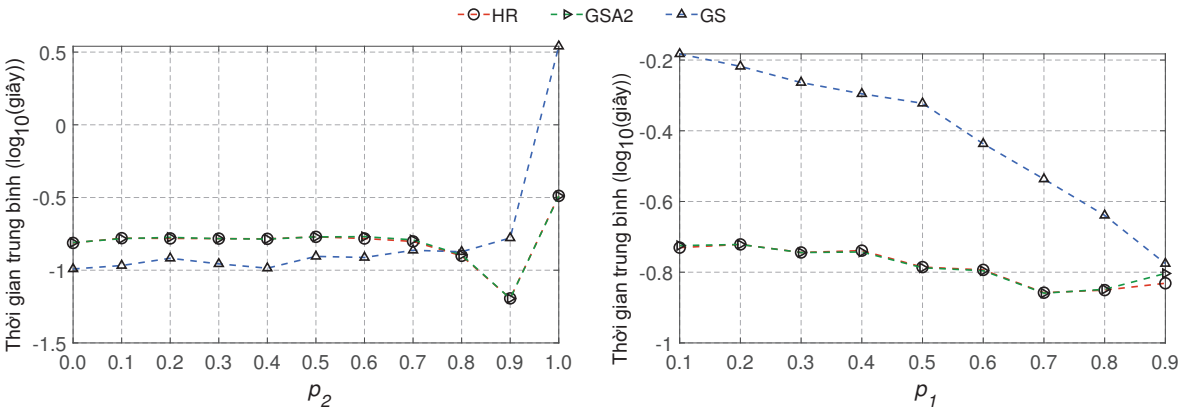


(a) $n = 300$ (b) $n = 400$

Hình 2.12: *Phần trăm phép ghép hoàn chỉnh của* HR *và* GSA2

Thực nghiệm 2.6. Trong thực nghiệm 2.5, với $p_1 \in \{0.1, 0.2, \cdots, 0.8\}$ và khi n tăng, cả HR và GSA2 đều dễ tìm được các phép ghép hoàn chỉnh vì số lượng các cặp chấp nhận trong danh sách xếp hạng của mỗi $m_i \in \mathcal{M}$ và mỗi $w_j \in \mathcal{W}$ tăng. Do đó việc so sánh tỷ lệ phần trăm phép ghép hoàn chỉnh tìm được bởi HR và GSA2 không có nhiều ý nghĩa. Trong thực nghiệm này, các tham số để tạo các thể hiện SMTI được chọn là $n \in \{300, 400\}$, $p_1 \in \{0.90, 0.92, \cdots, 0.98\}$ và $p_2 \in \{0.0, 0.1, \cdots, 1.0\}$. Hình 2.12 chỉ ra kết quả của thực nghiệm này. Chúng ta thấy rằng khi p_1 tăng, phần trăm phép ghép hoàn chỉnh tìm được bởi HR và GSA2 giảm và khi p_2 tăng, phần trăm phép ghép hoàn chỉnh tìm được bởi HR và GSA2 tăng. Tuy nhiên, HR vượt trội GSA2 về việc tìm kiếm các phép ghép hoàn chỉnh cho các thể hiện SMTI.

Thực nghiệm 2.7. Trong các thực nghiệm trên, khi n nhỏ, thời gian thực hiện trung bình của HR và GSA2 là rất nhỏ và xấp xỉ nhau và do đó việc so sánh thời gian thực hiện của HR và GSA2 không có nhiều ý nghĩa. Để so sánh thời gian thực hiện của HR và GSA2 rõ ràng



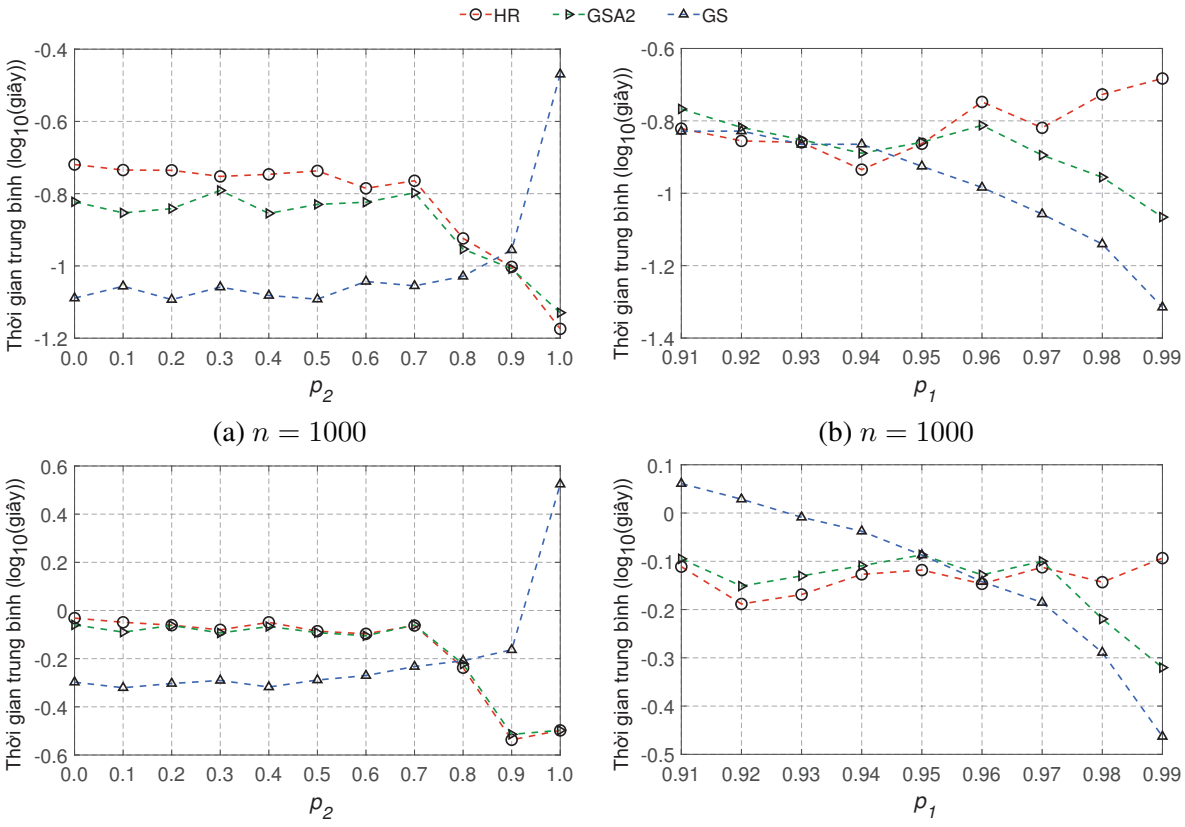
(a) $n = 1000$ (b) $n = 1000$

(c) $n = 2000$ (d) $n = 2000$

Hình 2.13: *Thời gian thực hiện trung bình của* HR, GSA2 *và* GS

hơn, luận án chọn $n \in \{1000, 2000\}$, $p_1 \in \{0.1, 0.2, \cdots, 0.9\}$ và $p_2 \in \{0.0, 0.1, \cdots, 1.0\}$. Vì HR và GSA2 đều dựa trên GS [45, 62], luận án so sánh thời gian thực hiện của HR với GSA2 và GS. Hình 2.13 chỉ ra thời gian thực hiện trung bình của HR, GSA2 và GS để tìm các phép ghép hoàn chỉnh. Chúng ta thấy rằng thời gian thực hiện của HR xấp xỉ thời gian thực hiện của GSA2. Khi p_2 tăng từ 0.0 lên 0.8, thời gian thực hiện của cả HR và GSA2 hầu như không thay đổi, nhưng lớn hơn thời gian thực hiện của GS. Khi $p_2 = 0.9$, thời gian thực hiện của cả HR và GSA2 đều giảm đáng kể, nhưng thời gian thực hiện của GS tăng. Khi $p_2 = 1.0$, thời gian thực hiện của HR, GSA2 và GS tăng. Khi p_1 tăng từ 0.1 lên 0.9, thời gian thực hiện của cả HR và GSA2 hầu như không thay đổi, trong khi thời gian của GS giảm đáng kể. Cần nhấn mạnh rằng khi $n = 2000$, các thể hiện SMTI có không gian tìm kiếm rất lớn ($2000! \backsimeq 10^{5735}$ phép ghép), nhưng HR chạy khoảng $10^0 = 1$ giây cho $p_2 \leq 0.9$ và khoảng $10^{0.3} \backsimeq 2$ giây cho $p_2 = 1.0$.

Thực nghiệm 2.8. Trong Thực nghiệm 2.7, khi $p_1 \in \{0.1, 0.2, \cdots, 0.9\}$, cả HR và GSA2 đều tìm được 100% phép ghép hoàn chỉnh. Điều này có thể dẫn đến thời gian thực hiện của HR xấp xỉ với thời gian thực hiện của GSA2. Trong thực nghiệm này, luận án chọn n và p_2 như trong Thực nghiệm 2.7, nhưng chọn $p_1 \in \{0.91, 0.92, \cdots, 0.99\}$. Hình 2.14 chỉ ra thời gian thực hiện trung bình của HR, GSA2 và GS để tìm phép ghép hoàn chỉnh. Chúng ta



(a) $n = 1000$ (b) $n = 1000$

(c) $n = 2000$ (d) $n = 2000$

Hình 2.14: *Thời gian thực hiện trung bình của* HR, GSA2 *và* GS

thấy rằng thời gian thực hiện của HR xấp xỉ với thời gian thực hiện của GSA2. Khi p_2 tăng từ 0.0 lên 1.0, thời gian thực hiện của cả HR và GSA2 đều giảm, trong khi thời gian của GS tăng. Khi p_1 tăng từ 0.91 lên 0.99, thời gian thực hiện của HR hầu như không thay đổi, trong khi thời gian thực hiện của GSA2 và GS giảm. Cần lưu ý rằng (*i*) khi $n = 1000$, HR và GSA2 tìm được tương ứng 72% và 67% phép ghép hoàn chỉnh; và (*ii*) khi $n = 2000$, HR và GSA2 tìm được tương ứng 90% và 87% phép ghép hoàn chỉnh.

Như đã đề cập ở trên, HR bao gồm GS để tìm một phép ghép ổn định và một hàm heuristic để cải thiện kích thước của phép ghép ổn định, tuy nhiên khi $p_2 \in \{0.9, 1.0\}$ hoặc p_1 nhỏ, thời gian thực hiện của HR nhỏ hơn GS. Điều này là do ở mỗi bước lặp của GS, mỗi $m_i \in \mathcal{M}$ đề xuất một $w_j \in \mathcal{W}$ mà m_i thích nhất. Nếu w_j có một bạn ghép m_k và $rank(w_j, m_k) < rank(w_j, m_i)$ thì m_i bị w_j từ chối. Khi $p_2 = 0.9$, mỗi $m_i \in \mathcal{M}$ xếp hạng những $w_j \in \mathcal{W}$ gần như ngang nhau (tức là ít khi $rank(w_j, m_i) < rank(w_j, m_k)$) và khi $p_2 = 1.0$, mỗi $m_i \in \mathcal{M}$ xếp hạng những $w_j \in \mathcal{W}$ như nhau (tức là $rank(w_j, m_i) = rank(w_j, m_k)$) và ngược lại. Điều này có nghĩa là m_i phải đề xuất một w_j tiếp theo mà m_i thích nhất ở bước lặp tiếp theo. Nếu mọi w_j trong danh sách ưu tiên của m_i đều có bạn ghép, thì m_i phải đề xuất tới mọi w_j nhưng bị w_j từ chối và do đó m_i trở thành người độc thân. Ngược lại, ở mỗi bước lặp HR, mỗi $m_i \in \mathcal{M}$ đề xuất một $w_j \in \mathcal{W}$ mà m_i thích nhất. Chúng

ta xem xét hai trường hợp như sau: (*i*) nếu tồn tại một $w_j \in \mathcal{W}$ độc thân trong tập $\mathcal{W}$ mà m_i thích ngang nhau, thì w_j được ghép cho m_i; (*ii*) nếu w_j đang ghép với m_k và nếu có một $w_t \in \mathcal{W}$ độc thân mà $rank(m_k, w_t) = rank(m_k, w_j)$ thì w_j được ghép với m_i và w_t có cơ hội được ghép với m_k khi m_k đề xuất w_t ở một bước lặp tiếp theo nào đó. Do vậy, m_i không phải tìm $w_j \in \mathcal{W}$ thích nhất ở các bước lặp tiếp theo. Rõ ràng, khi p_1 tăng lên, mỗi $m_i \in \mathcal{M}$ xếp hạng ít $w_j \in \mathcal{W}$ hơn trong danh sách xếp hạng của m_i và do đó HR chạy nhanh hơn nhiều so với GS khi $p_2 \in \{0.9, 1.0\}$ hoặc p_1 nhỏ.

2.4. Kết luận Chương 2

Chương này đã trình bày hai thuật toán đề xuất bao gồm MCS và HR để giải quyết bài toán MAX-SMTI, tức là tìm các phép ghép ổn định với kích thước tối đa cho các thể hiện của bài toán SMTI.

Thuật toán MCS tìm một nghiệm của bài toán từ một phép ghép ngẫu nhiên. Ở mỗi bước lặp, MCS tìm một tập hợp các cặp chặn vượt trội, định nghĩa một hàm heuristic để chọn cặp chặn vượt trội tốt nhất và loại bỏ cặp chặn được chọn khỏi phép ghép hiện tại. MCS lặp cho đến khi tìm được phép ghép hoàn chỉnh hoặc đạt đến số bước lặp tối đa. Kết quả thực nghiệm với các bộ dữ liệu được tạo ngẫu nhiên chỉ ra rằng MCS hiệu quả vượt trội so với thuật toán LTIU [11] và AS [14] về thời gian và chất lượng nghiệm cho bài toán MAX-SMTI kích thước lớn.

Thuật toán HR tìm một nghiệm của bài toán từ một phép ghép rỗng. Ở mỗi bước lặp, HR áp dụng thuật toán GS [45, 62] để tìm một phép ghép ổn định. Nếu phép ghép tìm được không hoàn chỉnh, HR định nghĩa một hàm heuristic để cải tiến kích thước của phép ghép. HR lặp cho đến khi tìm được một phép ghép hoàn chỉnh hoặc đạt đến số bước lặp tối đa. Kết quả thực nghiệm với các bộ dữ liệu được tạo ngẫu nhiên chỉ ra rằng HR hiệu quả vượt trội so với thuật toán MCS [111] và GSA2 [62] về thời gian và chất lượng nghiệm cho bài toán MAX-SMTI kích thước lớn.

CHƯƠNG 3.
ĐỀ XUẤT THUẬT TOÁN GIẢI BÀI TOÁN MAX-HRT

Chương này trình bày hai thuật toán tìm kiếm heuristic để giải quyết bài toán MAX-HRT, tức là tìm một phép ghép ổn định với kích thước tối đa cho các thể hiện của bài toán HRT. Các thuật toán này được công bố tại công trình [A.3] và [B.2] trong phần "*Các công bố sử dụng trong Luận án*".

3.1. Giới thiệu

Bài toán HRT [2] là một bài toán mở rộng của bài toán SMTI [1, 53]. Một thể hiện HRT kích thước $n \times m$ gồm một tập $\mathcal{R} = \{r_1, r_2, \cdots, r_n\}$ các sinh viên và một tập $\mathcal{H} = \{h_1, h_2, \cdots, h_m\}$ 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 không nghiêm ngặt, mỗi $h_i \in \mathcal{H}$ xếp hạng một tập con của $\mathcal{R}$ theo một thứ tự ưu tiên không nghiêm ngặt và mỗi $h_j \in \mathcal{H}$ có một số lượng tối đa $c_j \in \mathbb{Z}^+$ sinh viên có thể nhận thực tập. Mục tiêu của bài toán HRT là tìm một phép ghép ổn định $M = \{(r_i, h_j) \in \mathcal{R} \times \mathcal{H}\}$ với kích thước lớn nhất, tức là bài toán MAX-HRT. Với việc xuất hiện thêm các ràng buộc c_j trong các doanh nghiệp $h_j \in \mathcal{H}$ của các thể hiện HRT thì các thuật toán đã đề xuất cho bài toán MAX-SMTI chưa thực sự hiệu quả khi áp dụng giải quyết bài toán MAX-HRT. Do vậy, gần đây nhiều thuật toán đã được đề xuất [9, 28, 62, 20, 79, 14, 53] để giải quyết bài toán MAX-HRT. Tuy nhiên, các thuật toán đã đề xuất thường giải quyết với bài toán MAX-HRT có kích thước nhỏ. Irving và cộng sự [55] đã chứng minh rằng bài toán MAX-HRT là một bài toán NP-khó, do đó thách thức của các nhà nghiên cứu là tìm ra các phương pháp giải bài toán MAX-HRT với kích thước lớn hiệu quả hơn các phương pháp đã công bố.

3.2. Đề xuất thuật toán MCA

3.2.1. Ý tưởng

Phần này đề xuất một thuật toán xung đột tối thiểu (Min-Conflicts algorithm, viết tắt MCA) để giải quyết bài toán MAX-HRT. Ý tưởng chính của đề xuất MCA là xem bài toán

Algorithm 3.1: Thuật toán MCA

Input: - Thể hiện I của HRT.
- Xác suất nhỏ, p.
- Bước lặp tối đa, max_iters.

Output: Một phép ghép M.

```
1.  function MCA(I)
2.      Khởi tạo ngẫu nhiên M;
3.      M_best := M;
4.      f_best := n;
5.      iter := 0;
6.      while (iter ≤ max_iters) do
7.          iter := iter + 1;
8.          [f(M), X] := Find_Cost_And_UBPs(M);
9.          if (X = ∅) then
10.             if (f_best > f(M)) then
11.                 M_best := M;
12.                 f_best := f(M);
13.             if (f_best > 0) then
14.                 M := phép ghép được tạo ngẫu nhiên;
15.                 continue;
16.             else
17.                 break;
18.         if (xác suất nhỏ p) then
19.             r_j := một r_i ∈ X ngẫu nhiên;
20.         else
21.             r_j := argmin(rank(h_k, r_i)), ∀(r_i, h_k) ∈ X;
22.         Xóa cặp chặn (r_j, X(r_j));
23.     return M_best;
24. end function
```

HRT như một bài toán thỏa mãn ràng buộc trong đó tập $r_i \in \mathcal{R}$ là các biến, các h_j trong danh sách xếp hạng của mỗi r_i là miền giá trị của các biến và các ràng buộc là các điều kiện tạo ra các cặp chặn của một phép ghép. Theo đó, một phép ghép ổn định M là một phép ghép các doanh nghiệp cho các sinh viên mà không vi phạm các ràng buộc. MCA bắt đầu từ một phép ghép ngẫu nhiên $M = \{(r_i, h_j) \in \mathcal{R} \times \mathcal{H}\}$, trong đó (r_i, h_j) là các cặp chấp nhận. Tại mỗi bước lặp, MCA là tìm một tập cặp trội nhất theo xếp hạng của $r_i \in \mathcal{R}$ cho phép ghép M và xóa một cặp UBP tốt nhất theo nghĩa không chỉ xóa tất cả các cặp trội nhất tạo bởi r_i mà còn xóa nhiều nhất các cặp trội nhất theo xếp hạng của h_j. Nếu MCA tìm thấy một phép ghép không hoàn chỉnh, nó khởi tạo lại một phép ghép mới và tiếp tục lặp đến một số bước lặp tối đa. Khi kết thúc, MCA trả về một phép ghép hoàn chỉnh hoặc một phép ghép ổn định với kích thước tối đa đã tìm được trong các bước lặp trước đó.

Algorithm 3.2: Tìm giá trị hàm $f(M)$ và cặp chặn vượt trội X

Input: Phép ghép M.
Output: Giá trị hàm $f(M)$ tập hợp các cặp chặn cặp trội nhất X.

```
1.  function Find_Cost_And_UBPs(M)
2.      X := ∅;
3.      #nur := 0;
4.      #nbp := 0;
5.      for (mỗi r_i ∈ R) do
6.          ubp := false;
7.          while (∃h_j ∈ H|rank(r_i, h_j) > 0) do
8.              h_j := argmin(rank(r_i, h_j) > 0);
9.              if (rank(r_i, h_j) = rank(r_i, M(r_i))) then
10.                 break;
11.             if ((r_i, h_j) là một cặp chặn) then
12.                 X := X ∪ (r_i, h_j);
13.                 #nbp := #nbp + 1;
14.                 ubp := true;
15.                 break;
16.             else
17.                 Xóa h_j trong danh sách xếp hạng của r_i;
18.         if ((ubp = false) và (r_i là chưa được ghép)) then
19.             #nur := #nur + 1;
20.     f(M) := #nbp + #nur;
21.     return (f(M), X);
22. end function
```

3.2.2. Mô tả thuật toán

Thuật toán MCA được mô tả chi tiết trong Thuật toán 3.1. MCA bắt đầu tìm một phép ghép ổn định với kích thước lớn nhất M_{best} từ một phép ghép được tạo ngẫu nhiên $M = \{(r_i, h_j) \in \mathcal{R} \times \mathcal{H}\}$, trong đó (r_i, h_j) là các cặp chấp nhận (dòng 2). Tại mỗi lần lặp, MCA gọi hàm được mô tả trong Thuật toán 3.2 để tính giá trị $f(M)$ của M và tập cặp trội nhất, ký hiệu là $X = \{(r_i, h_j) \in \mathcal{R} \times \mathcal{H}\}$, cho M. Nếu X rỗng, tức là M ổn định, thì MCA kiểm tra nếu: (*i*) $f_{best} > f(M)$, thì MCA gán M cho M_{best} (dòng 10-12); và (*ii*) $f_{best} > 0$, tức là M_{best} là phép ghép không hoàn chỉnh, MCA sẽ khởi tạo lại một phép ghép M mới và tiếp tục lặp (dòng 13-15). Ngược lại, MCA kiểm tra nếu thỏa mãn một xác suất p nhỏ, MCA sẽ chọn ngẫu nhiên một $r_j \in X$ (dòng 18-19). Nếu không, MCA sẽ chọn một $r_j \in X$ (dòng 21), sao cho $rank(h_k, r_j)$ nhỏ nhất với mọi $(r_i, h_k) \in X$. Tiếp theo, MCA loại bỏ cặp chặn $(r_j, X(r_j))$ cho phép phép M (dòng 22). MCA lặp cho đến khi M_{best} là một phép ghép hoàn chỉnh hoặc đạt đến số lần lặp tối đa. Trong trường hợp thứ hai, MCA trả về một phép ghép ổn định với kích thước lớn nhất đã tìm thấy trong các bước lặp. Chú ý rằng việc xóa

Bảng 3.1: *Ví dụ một thể hiện* HRT

Danh sách xếp hạng của $r_i \in \mathcal{R}$	Danh sách xếp hạng của $h_j \in \mathcal{H}$
r_1: h_1 h_3 h_2	h_1: 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: r_5 r_6 (r_3 r_4) r_1
r_3: h_1 h_5 h_2	h_3: (r_5 r_2) r_6 r_1 r_7
r_4: h_1 (h_2 h_4)	h_4: r_8 r_2 r_4 r_7
r_5: h_3 h_1 h_2	h_5: 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	
Số sinh viên tối đa của $h_j \in \mathcal{H}$: $c_1 = 2$, $c_2 = 3$, $c_3 = c_4 = c_5 = 1$	

cặp chặn (r_i, h_j) cho M sẽ tạo ra một phép ghép M', trong đó h_j được ghép cho r_i và nếu $|M(h_j)| = c_j$ thì r_i được xếp hạng ưu tiên thấp nhất bởi h_j trong $M(h_j)$ bị loại khỏi phép ghép M và các cặp khác trong M không đổi.

Hàm tìm giá trị $f(M)$ và tập hợp cặp trội nhất cho M được chỉ ra trong Thuật toán 3.2. Hàm định nghĩa giá trị $f(M) = \#nbp(M) + \#nur(M)$, trong đó $\#nbp(M)$ là số các cặp chặn trội nhất cho M và $\#nur(M)$ là số $r_i \in \mathcal{R}$ chưa được ghép trong M. Với mỗi $r_i \in \mathcal{R}$, hàm sẽ xét mỗi h_j trong danh sách xếp hạng của r_i sao cho r_i thích h_j hơn $M(r_i)$, tức là $rank(r_i, h_j) < rank(r_i, M(r_i))$ (dòng 8-10). Nếu (r_i, h_j) là một cặp chặn thì (r_i, h_j) là một cặp chặn trội nhất và do vậy số lượng $\#nbp$ sẽ tăng và cặp (r_i, h_j) được thêm vào tập X của các cặp chặn trội nhất (dòng 11-15). Ngược lại, nếu (r_i, h_j) không tạo thành một cặp chặn thì r_i xóa h_j trong danh sách xếp hạng của r_i (dòng 17). Nếu r_i duyệt hết các h_j mà $rank(r_i) = rank(r_i, M(r_i))$ và r_i chưa được ghép với h_j thì hàm sẽ tăng giá trị của $\#nur$ (dòng 18-19) và thực hiện lặp cho $r_i \in \mathcal{R}$ tiếp theo.

3.2.3. Ví dụ

Cho một thể hiện HRT được chỉ ra trong Bảng 3.1. Bắt đầu từ một phép ghép được tạo ngẫu nhiên M = $\{(r_1, \varnothing), (r_2, h_5), (r_3, h_1), (r_4, h_4), (r_5, \varnothing), (r_6, h_1), (r_7, h_3), (r_8, \varnothing)\}$, trong đó $f(M) = 6$, MCA gán M cho M_{best} và chạy các lần lặp được chỉ ra trong Bảng 3.2. Ở bước lặp đầu tiên, MCA tìm một tập X của các cặp chặn trội nhất cho phép ghép M. Vì $rank(h_3, r_5) = 1$ có giá trị nhỏ nhất, MCA xóa cặp (r_5, h_3) trong M để tìm được một phép ghép ổn định mới M = $\{(r_1, \varnothing), (r_2, h_5), (r_3, h_1), (r_4, h_4), (r_5, h_3), (r_6, h_1), (r_7, h_3), (r_8, \varnothing)\}$. MCA lặp lại cho đến bước lặp 8, khi đó X rỗng và $f(M) = 0$. Vì $f_{best} > f(M)$, M được gán cho M_{best}, tức là $f_{best} = 0$, và do đó MCA trả về phép ghép hoàn chỉnh M_{best} = $\{(r_1, h_2), (r_2, h_1), (r_3, h_1), (r_4, h_2), (r_5, h_3), (r_6, h_2), (r_7, h_5), (r_8, h_4)\}$ với $|M| = 8$.

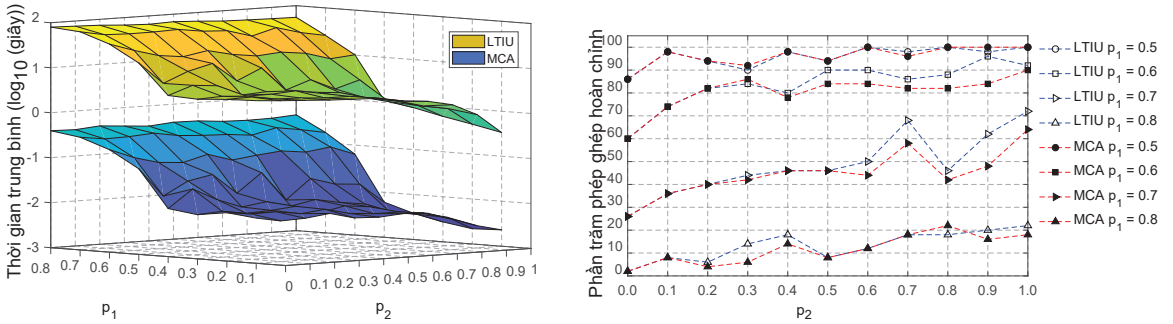
Bảng 3.2: *Ví dụ thực hiện của thuật toán* MCA

Lặp	M	$f(M)$	X	$rank(h_k, r_i)$
1	$(r_1, \varnothing)$, (r_2, h_5), (r_3, h_1), (r_4, h_4), $(r_5, \varnothing)$, (r_6, h_1), (r_7, h_3), $(r_8, \varnothing)$	6	(r_1, h_2)	4
			(r_2, h_1)	2
			(r_4, h_1)	3
			(r_5, h_3)	1
			(r_6, h_2)	2
			(r_8, h_5)	2
2	$(r_1, \varnothing)$, (r_2, h_5), (r_3, h_1), (r_4, h_4), (r_5, h_3), (r_6, h_1), $(r_7, \varnothing)$, $(r_8, \varnothing)$	6	(r_1, h_2)	4
			(r_2, h_1)	2
			(r_4, h_1)	3
			(r_6, h_2)	2
			(r_7, h_5)	2
			(r_8, h_5)	2
3	$(r_1, \varnothing)$, (r_2, h_1), (r_3, h_1), (r_4, h_4), (r_5, h_3), $(r_6, \varnothing)$, $(r_7, \varnothing)$, $(r_8, \varnothing)$	4	(r_1, h_2)	4
			(r_6, h_2)	2
			(r_7, h_5)	2
			(r_8, h_5)	2
4	$(r_1, \varnothing)$, (r_2, h_1), (r_3, h_1), (r_4, h_4), (r_5, h_3), (r_6, h_2), $(r_7, \varnothing)$, $(r_8, \varnothing)$	3	(r_1, h_2)	4
			(r_7, h_5)	2
			(r_8, h_5)	2
5	$(r_1, \varnothing)$, (r_2, h_1), (r_3, h_1), (r_4, h_4), (r_5, h_3), (r_6, h_2), (r_7, h_5), $(r_8, \varnothing)$	2	(r_1, h_2)	4
			(r_8, h_4)	1
6	$(r_1, \varnothing)$, (r_2, h_1), (r_3, h_1), $(r_4, \varnothing)$, (r_5, h_3), (r_6, h_2), (r_7, h_5), (r_8, h_4)	2	(r_1, h_2)	4
			(r_4, h_2)	3
7	$(r_1, \varnothing)$, (r_2, h_1),(r_3, h_1), (r_4, h_2), (r_5, h_3), (r_6, h_2), (r_7, h_5), (r_8, h_4)	1	(r_1, h_2)	4
8	(r_1, h_2), (r_2, h_1), (r_3, h_1), (r_4, h_2), (r_5, h_3), (r_6, h_2), (r_7, h_5), (r_8, h_4)	0	$\{\}$	
9	Sau 8 bước lặp, MCA trả về M = (r_1, h_2), (r_2, h_1), (r_3, h_1), (r_4, h_2), (r_5, h_3), (r_6, h_2), (r_7, h_5), (r_8, h_4) với $\|M\| = 8$			

3.2.4. Các kết quả thực nghiệm

3.2.4.1. So sánh với thuật toán LTIU

Thực nghiệm 3.1. Đầu tiên luận án so sánh thời gian thực hiện và chất lượng nghiệm được tìm được của MCA với thời gian thực hiện và chất lượng nghiệm tìm được của thuật toán LTIU [45]. Dữ liệu thực nghiệm là các thể hiện HRT được tạo ngẫu nhiên với các tham số gồm $n = 50$, $m = 10$, $p_1 \in [0.1, 0.8]$ và $p_2 \in [0.0, 1.0]$ với bước tăng 0.1. Với mỗi bộ tham số (n, m, p_1, p_2), luận án tạo ngẫu nhiên 50 thể hiện HRT với c_j của mỗi $h_j \in \mathcal{H}$ là một số ngẫu nhiên và $c_j \in [1, C_j]$, trong đó C_j là số $r_i \in \mathcal{R}$ được xếp hạng bởi $h_j \in \mathcal{H}$. Vì MCA và LTIU đều tìm phép ghép ổn định với kích thước tối đa từ một phép ghép ngẫu nhiên, với mỗi thể hiện của HRT, một phép ghép ngẫu nhiên được tạo ra để làm đầu vào cho cả hai



(a)Trung bình thời gian thực hiện (b) Phần trăm phép ghép hoàn chỉnh.

Hình 3.1: *Thời gian thực nghiệm và chất lượng nghiệm của* MCA *và* LTIU

thuật toán MCA và LTIU. Xác suất cho MCA để chọn ngẫu nhiên một cặp chặn trội nhất là p = 0.03. Số lần lặp tối đa trong cả hai thuật toán MCA và LTIU là 2000.

Hình 3.1(a) chỉ ra thời gian thực hiện trung bình của MCA và LTIU để tìm các phép ghép ổn định tối đa của các thể hiện HRT. Thời gian thực hiện trung bình của MCA nhỏ hơn nhiều so với LTIU (do đó luận án sử dụng thang đo log10 trên trục Y). Khi p_1 thay đổi từ 0.1 đến 0.8, thời gian thực hiện của cả MCA và LTIU đều tăng đáng kể với mọi giá trị của p_2. Thời gian thực hiện trung bình của MCA tăng từ khoảng 0.007 giây lên 0.37 giây, trong khi thời gian thực hiện trung bình của LTIU tăng từ khoảng 2.55 giây lên 65.80 giây với mọi giá trị của p_2. Ngược lại, khi p_2 thay đổi từ 0.0 đến 1.0, thời gian thực hiện trung bình của cả MCA và LTIU giảm với mọi giá trị của p_1. Kết quả thực nghiệm cũng chỉ ra rằng MCA chạy nhanh hơn khoảng 190 lần so với LTIU với mọi giá trị p_1 và p_2. Điều này có thể được giải thích như sau: LTIU là một thuật toán tìm kiếm cục bộ, do đó tại mỗi lần lặp LTIU phải tìm một tập hợp các phép ghép lân cận của phép ghép hiện tại, tính chi phí của tất cả các phép ghép lân cận và di chuyển phép ghép hiện tại đến phép ghép tốt nhất trong tập hợp phép ghép lân cận. Với thể hiện HRT có kích thước $n \times m$, tại mỗi bước lặp LTIU cần $O(n \times m)$ thời gian để xác định tập cặp trội nhất cho một phép ghép M. Bằng cách xóa từng cặp chặn trong tập hợp cặp trội nhất để tạo các phép ghép lân cận, LTIU sẽ đạt được tập hợp các phép ghép lân cận của phép ghép hiện tại. Vì chi phí $f(M)$ của một phép ghép M được tính cần $O(n^2)$ thời gian, LTIU cần $O(n^3)$ để xác định chi phí của các phép lân cận để tìm ra một phép ghép lân cận tốt nhất trong tập hợp các phép lân cận của phép ghép hiện tại. Như vậy tại mỗi bước lặp, LTIU cần $O(n \times m) + O(n^3)$. Rõ ràng, kích thước của các thể hiện HRT càng lớn thì LTIU càng chạy chậm. Tuy nhiên ở mỗi bước lặp, MCA chỉ cần $O(n \times m)$ thời gian để xác định tập hợp các cặp trội nhất và chỉ xóa một cặp chặn trong tập hợp cặp trội nhất để tạo một phép ghép mới cho bước lặp tiếp theo. Điều này cho phép MCA chạy nhanh hơn nhiều so với LTIU.

Hình 3.1(b) chỉ ra phần trăm phép ghép hoàn chỉnh tìm được bởi thuật toán MCA và LTIU. Chú ý rằng khi p_1 thay đổi từ 0.1 đến 0.5, cả MCA và LTIU luôn tìm được 100% phép ghép hoàn chỉnh và do vậy kết quả không được chỉ ra trong hình này. Khi p_1 thay đổi từ 0.6 đến 0.8, tỷ lệ phần trăm phép ghép hoàn chỉnh mà LTIU tìm được cao hơn không đáng kể so với MCA. Kết quả thực nghiệm cũng chỉ ra rằng khi $p_1 = 0.7$ và $p_1 = 0.8$, tức là xác suất tạo danh sách xếp hạng không đầy đủ tăng, thì tỷ lệ phép ghép hoàn chỉnh tìm được bởi hai thuật toán MCA và LTIU giảm.

3.2.4.2. Thực nghiệm với HRT *kích thước lớn*

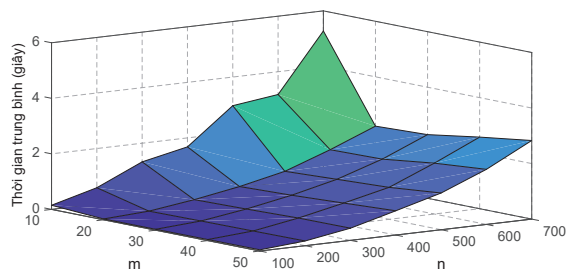
Với thể hiện HRT kích thước $n \times m$, tại mỗi bước lặp LTIU cần $O(n \times m) + O(n^3)$ thời gian tính toán và do vậy nếu kích thước của các thể hiện HRT tăng thì thời gian thực hiện của LTIU tăng theo hàm mũ. Với $n = 50$ và $n = 10$, MCA đã vượt trội LTIU về thời gian và vì vậy phần này trình bày các kết quả thực nghiệm của MCA với các thể hiện HRT kích thước lớn để xem xét hiệu quả của MCA mà không so sánh với LTIU.

Thực nghiệm 3.2. Trong thực nghiệm này, dữ liệu thực nghiệm là các thể hiện HRT được tạo ngẫu nhiên với các tham số gồm $n = \{100, 200, \cdots, 700\}$, $m = \{10, 20, \cdots, 50\}$, $p_1 = 0.5$ và $p_2 = 0.5$. Với mỗi bộ tham số (n, m, p_1, p_2), luận án tạo ngẫu nhiên 50 thể hiện HRT với c_j của mỗi $h_j \in \mathcal{H}$ là một số ngẫu nhiên và $c_j \in [1, C_j]$, trong đó C_j là số $r_i \in \mathcal{R}$ được xếp hạng bởi $h_j \in \mathcal{H}$.

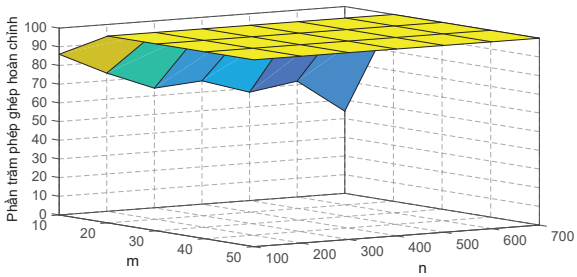
Hình 3.2(a) chỉ ra thời gian thực hiện trung bình của MCA. Khi $m = 10$, thời gian thực hiện của MCA tăng từ khoảng 0.15 giây lên 5.25 giây với giá trị n thay đổi từ 100 đến 700. Với các giá trị khác của m, thời gian thực hiện tăng từ khoảng 0.03 giây lên 2.25 giây.

Hình 3.2(b) chỉ ra phần trăm phép ghép hoàn chỉnh. MCA tìm được 100% phép ghép hoàn chỉnh với mọi giá trị của n và $m \neq 10$. Tuy nhiên, khi $m = 10$ MCA chỉ tìm được 86%, 74%, 64%, 66%, 58%, 62% và 44% phép ghép hoàn chỉnh cho $n = \{100, 200, \cdots, 700\}$, tương ứng. Điều này có nghĩa là phần trăm phép ghép hoàn chỉnh giảm khi $m = 10$ và n tăng. Khi MCA không thể tìm thấy 100% các phép ghép hoàn chỉnh, có nghĩa là tồn tại một số phép ghép ổn định nhưng không hoàn chỉnh và do vậy MCA phải lặp đến số bước lặp tối đa. Do vậy, MCA tăng thời gian thực hiện khi $m = 10$, tuy nhiên, kết quả thực nghiệm chỉ ra rằng MCA hiệu quả đối với các thể hiện HRT có kích thước lớn.

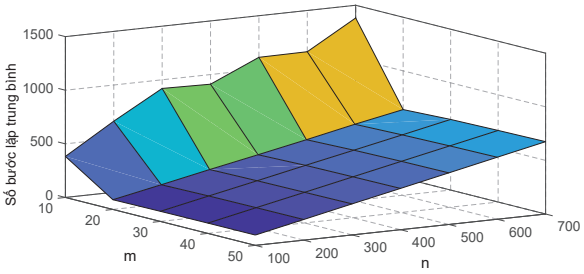
Hình 3.2(c) chỉ ra số bước lặp trung bình của MCA. Khi $m = 10$, MCA phải lặp đến đến số bước lặp tối đa để tìm phép ghép hoàn chỉnh và do đó, số bước lặp trung bình lớn hơn so với các giá trị khác của m. Tuy nhiên, ngay cả khi $n = 700$, số bước lặp tối đa không vượt



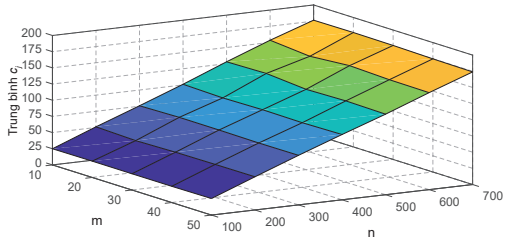
(a)Trung bình thời gian thực hiện



(b) Phần trăm phép ghép hoàn chỉnh



(c)Trung bình số bước lặp



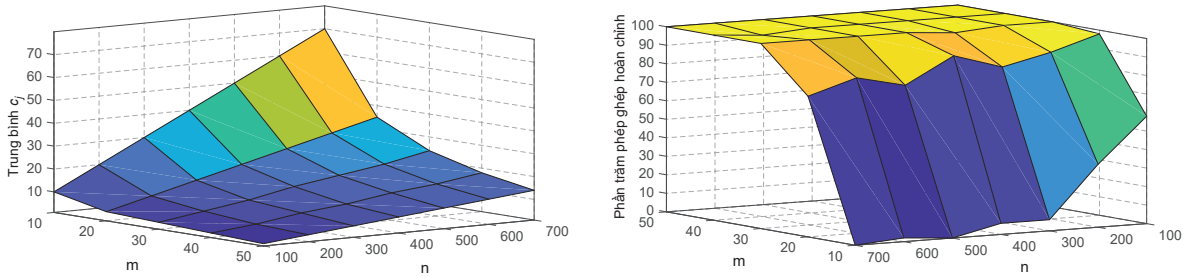
(d) c_j ngẫu nhiên

Hình 3.2: MCA *với các tham số* $n = \{100, 200, \cdots, 700\}$

quá 1500. Điều này có nghĩa là MCA hiệu quả về mặt thời gian thực hiện.

Hình 3.2(d) chỉ ra giá trị trung bình của các c_j của mỗi $h_j \in \mathcal{H}$ trong các thể hiện HRT đã tạo. Chúng ta thấy rằng giá trị trung bình của $c_j \approx n/4$, tức là 25, 50, 75, 100, 125, 150 và 175 với $n = \{100, 200, \cdots, 700\}$, tương ứng. Điều này bởi vì trong phương pháp tạo các thể hiện HRT, luận án chọn $p_1 = 0.5$, nghĩa là có khoảng 50% sinh viên được xếp hạng bởi mỗi h_j. Hơn nữa, c_j của mỗi h_j được sinh ngẫu nhiên với phân phối xác suất đều (uniform distribution) là $n/2$ sinh viên trong danh sách xếp hạng của mỗi h_j và vì vậy giá trị trung bình của $c_j \approx n/4$ cho mỗi h_j.

Thực nghiệm 3.3. Trong thực nghiệm này, luận án xem xét dữ liệu thực nghiệm là các thể hiện HRT được tạo ngẫu nhiên với các tham số như trong Thực nghiệm 3.2. Với mỗi bộ tham số (n, m, p_1, p_2), luận án tạo ngẫu nhiên 50 thể hiện HRT với $c_j = n/m$. Hình 3.3(a) chỉ ra rằng c_j nhỏ hơn 2.5 lần so với trường hợp c_j được tạo ngẫu nhiên như trong Hình 3.2(d). Hình 3.3(b) chỉ ra phần trăm phép ghép hoàn chỉnh tìm được bởi MCA. Rõ ràng, MCA không tìm được 100% phép ghép hoàn chỉnh khi $m = 10$ và $m = 20$ và một lần nữa chúng ta thấy rằng MCA rất khó tìm được 100% các phép ghép hoàn chỉnh khi m nhỏ và n lớn.



(a) $c_j = n/m$ (b) Phần trăm phép ghép hoàn chỉnh

Hình 3.3: *Phần trăm phép ghép hoàn chỉnh trong trường hợp* $c_j = n/m$

3.3. Đề xuất thuật toán HS

3.3.1. Ý tưởng

Phần này trình bày một thuật toán tìm kiếm heuristic (Heuristic Search algorithm, viết tắt HS) để giải quyết bài toán MAX-HRT. Ý tưởng chính của HS bắt đầu từ một phép ghép ngẫu nhiên, HS định nghĩa một hàm heuristic để định hướng tìm các cặp chặn trội nhất. Sau đó, HS loại các cặp chặn trội nhất ra khỏi phép ghép hiện tại để tìm một phép ghép ổn định với kích thước tối đa.

3.3.2. Mô tả thuật toán

Thuật toán HS được chỉ ra trong Thuật toán 3.3. Bắt đầu, HS tạo một phép ghép ngẫu nhiên M. Sau đó, HS gán M cho M_{best} và xem xét mỗi $r_i \in \mathcal{R}$ là đang hoạt động, có nghĩa là r_i có khả năng tạo ra cặp chặn, ký hiệu $a(r_i) = 1$, ngược lại, ký hiệu $a(r_i) = 0$. Ngoài ra, nếu một $r_i \in \mathcal{R}$ thay đổi một $h_j \in \mathcal{H}$ đang ghép với r_i, thì r_i sẽ tăng số lần thay đổi h_j, ký hiệu là $y(r_i)$. Tương tự, nếu một $h_j \in \mathcal{H}$ thay đổi $r_i \in \mathcal{R}$ đang ghép với h_j, h_j sẽ tăng số lần thay đổi r_i, ký hiệu là $z(h_j)$.

Bắt đầu, HS tạo một phép ghép ngẫu nhiên M, gán M cho M_{best} và đặt các $r_i \in \mathcal{R}$ trong trạng thái hoạt động (dòng 2-4). Bên cạnh đó, HS gán $y(r_i) = 0$ với mọi $r_i \in \mathcal{R}$, $z(h_j) = 0$ với mọi $h_j \in \mathcal{H}$, và lưu danh sách xếp hạng ban đầu của mọi $r_i \in \mathcal{R}$, ký hiệu $rank^{\dagger}(r_i, h_j)$ (dòng 5-7). Chú ý rằng cả $y(r_i)$ và $z(h_j)$ được sử dụng trong Thuật toán 3.5, trong khi $rank^{\dagger}(r_i, h_j)$ được sử dụng để khôi phục danh sách xếp hạng của r_i khi tìm một cặp chặn trội nhất, tức là r_i đã xóa h_j trong danh sách xếp hạng trong Thuật toán 3.4. Tại mỗi bước lặp, nếu tồn tại một $r_i \in \mathcal{R}$ đang hoạt động, thuật toán tìm một $h_j \in \mathcal{H}$ và một danh sách chờ $W(r_i)$ được chỉ ra trong Thuật toán 3.4 (dòng 20) sao cho (r_i, h_j) là một cặp chặn trội nhất và $W(r_i)$ là tập hợp các $h_t \in \mathcal{H}$ sao cho $rank(r_i, h_t) \leq rank(r_i, h_j)$. Nếu

Algorithm 3.3: Thuật toán HS

Input: - Thể hiện I của HRT.
- Bước lặp tối đa max_iter.

Output: Phép ghép ổn định M

1. **function** `HS (`I`)`
2. Khởi tạo phép ghép ngẫu nhiên M;
3. $M_{best} := M$;
4. $a(r_i) := 1, \forall r_i \in \mathcal{R}$;
5. $y(r_i) := 0, \forall r_i \in \mathcal{R}$;
6. $z(h_j) := 0, \forall h_j \in \mathcal{H}$;
7. $rank^{\dagger}(r_i, h_j) := rank(r_i, h_j), \forall r_i \in \mathcal{R}, h_j \in \mathcal{H}$;
8. $iter := 0$;
9. **while** $iter \leq max\_iter$ **do**
10. $iter := iter + 1$;
11. r_i := một sinh viên đang hoạt động;
12. **if** *($\nexists r_i$ mà $a(r_i) = 1$)* **then**
13. **if** $|M_{best}| < |M|$ **then**
14. $M_{best} := M$;
15. **if** $(|M_{best}| = n)$ **then** break;
16. M' := `Improve(`M`)`;
17. **if** $M' = M$ **then** break;
18. $M := M'$;
19. continue;
20. $[h_j, W(r_i)]$:= `Find_UBP_and_Wait_List(`r_i, M`)`;
21. **if** $h_j \neq \varnothing$ **then**
22. $M := M \setminus \{(r_i, h_k)\} \cup \{(r_i, h_j)\}$, với $h_k := M(r_i)$;
23. $a(r_i) := 0$;
24. **for** *mỗi* $r_t \in \mathcal{R}$ *để mà* $W(r_t) = h_k$ **do**
25. $rank(r_t, h_k) := rank^{\dagger}(r_t, h_k)$;
26. $a(r_t) := 1$;
27. **if** $|M(h_j)| > c_j$ **then**
28. r_w := một sinh viên có thứ tự xếp hạng ưu tiên thấp nhất;
29. $M := M \setminus \{(r_w, h_j)\}$;
30. $a(r_w) := 1$;
31. **else**
32. $a(r_i) := 0$;
33. **return** M_{best};
34. **end function**

không tồn tại h_j, nghĩa là r_i không có cặp chặn trong M và r_i trở thành không hoạt động (dòng 32). Ngược lại, thuật toán tìm $h_k = M(r_i)$, xóa cặp chặn trội nhất (r_i, h_j) cho M bằng cách ghép h_j cho r_i và r_i trở thành không hoạt động (dòng 22-23). Sau đó, thuật toán khôi phục h_k trong danh sách xếp hạng của mọi r_t mà có $h_k \in W(r_t)$ và r_t hoạt động trở lại vì vậy r_t có thể tìm một doanh nghiệp bất kỳ cho bước lặp tiếp theo (dòng 27-30). Sau đó, HS

Algorithm 3.4: Tìm cặp chặn trội nhất và danh sách chờ $W(r_i)$

Input: r_i và M.
Output: - h_j trong đó (r_i, h_j) là một UBP.
- Danh sách chờ $W(r_i)$.

1. **function** `Find_UBP_and_Wait_List` (r_i, M)
2. **for** $h_k \in$ *danh sách xếp hạng của* r_i **do**
3. $f(h_k) := rank(r_i, h_k) + |M(h_k)|/(c_k + 1)$;
4. $h_j := \varnothing$;
5. **while** $(\exists h_k \in \mathcal{H} | rank(r_i, h_k) > 0)$ **do**
6. $h_k :=$ argmin$(f(h_k) \geq 1), \forall h_k \in \mathcal{H}$;
7. **if** $rank(r_i, h_k) = rank(r_i, M(r_i))$ **then**
8. break;
9. **if** (r_i, h_k) *là cặp chặn* **then**
10. $h_j := h_k$;
11. break;
12. **else**
13. $W(r_i) := W(r_i) \cup h_k$;
14. $rank(r_i, h_k) := 0$;
15. $f(h_k) := 0$;
16. **return** h_j và $W(r_i)$;
17. **end function**

kiểm tra nếu $|M(h_j)| > c_j$, thì HS loại bỏ cặp (r_w, h_j) sao cho $|M(h_j)| = c_j$, trong đó r_w có thứ tự xếp hạng ưu tiên thấp nhất được ghép cho h_j và gán $a(r_w) = 1$. Nếu không tồn tại r_i nào đang hoạt động, có nghĩa là HS tìm được phép ghép ổn định M, HS kiểm tra nếu $|M_{best}| < |M|$, thì M gán cho M_{best}. Nếu $|M_{best}| = n$, nghĩa là tất cả $r_i \in \mathcal{R}$ được ghép với $h_j \in \mathcal{H}$, HS trả về phép ghép hoàn chỉnh M_{best}, ngược lại HS tìm được một phép ghép không hoàn chỉnh. Do đó, HS sử dụng Thuật toán 3.5 để thoát khỏi điểm tối thiểu cục bộ và tiếp tục lặp cho các bước tiếp theo (dòng 12-19). Thuật toán HS kết thúc khi tìm được một phép ghép hoàn chỉnh hoặc đạt đến số lần lặp tối đa. Trong trường hợp thứ hai, HS trả về một phép ghép ổn định với kích thước tối đa.

Thuật toán 3.4 được sử dụng để tìm cặp chặn trội nhất (r_i, h_j) cho phép ghép M và danh sách chờ $W(r_i)$. Ban đầu, thuật toán tính giá trị $f(h_k)$ cho mỗi h_k trong danh sách xếp hạng của r_i. Vì $1 \leq rank(r_i, h_k) \leq n$ và $0 \leq |M(h_k)| \leq c_k$, chúng ta có $1 \leq f(h_k) \leq n + c_k/(c_k + 1)$, trong đó c_k là số sinh viên tối đa có thể nhận được của h_k. Tại mỗi bước lặp, thuật toán chạy các bước như sau: Đầu tiên, thuật toán tìm một h_k sao cho $f(h_k)$ có giá trị nhỏ nhất (dòng 6). Tiếp theo, thuật toán kiểm tra nếu h_k có cùng thứ hạng với $M(r_i)$ trong danh sách xếp hạng của r_i, tức là r_i đang ghép với với $M(r_i)$ cái mà có thứ hạng nhỏ nhất nhất trong danh sách xếp hạng của r_i, sau đó nó kết thúc vòng lặp (dòng 7-8). Tuy nhiên, nếu

Algorithm 3.5: Cải thiện kích thước phép ghép ổn định

Input: Phép ghép ổn định, M.
Output: Phép ghép, M.

1. **function** `Improve(`M`)`
2. **for** *mỗi* $r_u \in \mathcal{R}$, *trong đó* $M(r_u) = \varnothing$ **do**
3. Tìm $(r_i, h_k) \in M$ sao cho $rank(h_k, r_i) = rank(h_k, r_u)$
4. **if** $y(r_u) \geq y(r_i)$ **then**
5. $M := M \setminus \{(r_i, h_k)\} \cup \{(r_u, h_k)\}$;
6. $y(r_u) := y(r_u) + 1$;
7. $a(r_u) := 0$;
8. $a(r_i) := 1$;
9. $rank(r_u, h_k) := rank^{\dagger}(r_u, h_k)$;
10. **for** *mỗi* $h_t \in \mathcal{H}$, *trong đó* $|M(h_t)| < c_j$ **do**
11. Tìm $(r_i, h_k) \in M$ sao cho $rank^{\dagger}(r_i, h_k) = rank^{\dagger}(r_i, h_t)$
12. **if** $z(h_t) \geq z(h_k)$ **then**
13. $M := M \setminus \{(r_i, h_k)\} \cup \{(r_i, h_t)\}$;
14. $z(h_t) := z(h_t) + 1$;
15. $rank(r_i, h_t) := rank^{\dagger}(r_i, h_t)$;
16. **for** *mỗi* r_w *sao cho* $W(r_w) = h_k$ **do**
17. $rank(r_w, h_k) := rank^{\dagger}(r_w, h_k)$;
18. $a(r_w) := 1$;
19. **return** M;
20. **end function**

(r_i, h_k) là một cặp chặn, nghĩa là (r_i, h_k) là cặp chặn đầu tiên và cũng là một cặp chặn UBP từ phía r_i, thuật toán trả về $h_j = h_k$ (dòng 9-11). Ngược lại thuật toán thêm h_k vào danh sách chờ $W(r_i)$ của r_i, xóa h_k trong danh sách xếp hạng của r_i (dòng 13-15) và chạy vòng lặp tiếp theo. Bằng cách sử dụng $W(r_i)$, nếu h_k bị từ chối bởi một sinh viên r_i nào đó, r_i khôi phục lại thứ hạng của h_k trong danh sách xếp hạng của r_i và r_i hoạt động trở lại. Điều này cho phép, h_k ghép với r_i tại một số bước lặp mà không tạo một cặp chặn trội nhất (r_i, h_k) cho phép ghép M.

Thuật toán 3.5 được sử dụng để vượt qua điểm tối thiểu cục bộ. Khi tìm thấy phép ghép M không hoàn chỉnh, thuật toán thay thế các cặp trong M để thu được phép ghép M' và HS tiếp tục tìm phép ghép hoàn chỉnh từ M'. Thuật toán xem xét hai trường hợp như sau:

Trường hợp 1: Đối với mỗi $r_u \in \mathcal{R}$ chưa được ghép, thuật toán tìm một cặp $(r_i, h_k) \in M$ mà r_i cùng thứ hạng với r_u trong danh sách xếp hạng của h_k (dòng 3). Nếu tồn tại cặp (r_i, h_k) và $y(r_u) \geq y(r_i)$, thuật toán sẽ hoán đổi r_i bằng r_u để tạo thành một cặp ổn định $(r_u, h_k) \in M$. Thuật toán tăng giá trị biến $y(r_u)$, đồng thời thiết lập $a(r_u) = 0$ và $a(r_i) = 1$ (dòng 5-8). Thuật toán khôi phục thứ hạng của h_k trong danh sách xếp hạng của r_u (dòng 9).

Cần lưu ý rằng điều kiện $y(r_u) \geq y(r_i)$, có nghĩa là một r_u với $M(r_u) = \varnothing$ có số lần được ghép trong hàm cải thiện trong Thuật toán 3.5 là nhiều hơn hoặc bằng r_i, thì thuật toán sẽ ưu tiên ghép cho h_k.

Trường hợp 2: Đối với mỗi $h_t \in \mathcal{H}$ chưa đủ số lượng sinh viên tối đa có thể nhận thực tập, thuật toán tìm một cặp $(r_i, h_k) \in M$ mà h_k cùng thứ hạng với h_t trong danh sách xếp hạng ban đầu của r_i (dòng 11). Nếu tồn tại một cặp (r_i, h_k) và $z(h_t) \geq z(h_k)$, thuật toán sẽ hoán đổi h_k cho h_t để tạo thành một cặp $(r_i, h_t) \in M$, đồng thời tăng $z(h_t)$ và phục hồi thứ hạng của h_t trong danh sách xếp hạng của r_i (dòng 13-15). Ngoài ra, thuật toán khôi phục thứ tự xếp hạng của h_k trong danh sách xếp hạng của r_w mà $W(r_w) = h_k$ và thiết lập giá trị $a(r_w) = 1$ (dòng 16-18).

Bảng 3.3: *Ví dụ một thể hiện* HRT

Danh sách xếp hạng của $r_i \in \mathcal{R}$	Danh sách xếp hạng của $h_j \in \mathcal{H}$
r_1: h_1 h_3	h_1: r_3 (r_2 r_5) (r_1 r_7 r_8)
r_2: (h_1 h_4)	h_2: r_3 r_8 (r_5 r_6) r_7 r_4
r_3: h_1 h_2	h_3: (r_1 r_4 r_6) r_5
r_4: (h_2 h_3 h_4)	h_4: r_2 r_6 r_4
r_5: (h_1 h_2) h_3	
r_6: h_4 h_2 h_3	
r_7: h_1 h_2	
r_8: (h_1 h_2)	
Số sinh viên tối đa của $h_j \in \mathcal{H}$: $c_1 = c_2 = c_3 = c_4 = 2$	

3.3.3. Ví dụ

Cho một thể hiện HRT được mô tả trong Bảng 3.3. Bắt đầu từ một phép ghép ngẫu nhiên M_0 = $\{(r_1, h_1), (r_2, h_1), (r_3, h_2), (r_4, h_3), (r_5, \varnothing), (r_6, h_4), (r_7, h_2), (r_8, \varnothing)\}$, thuật toán HS thực hiện các bước lặp được chỉ ra trong Bảng 3.4. Thuật toán xem xét các sinh viên với trạng thái hoạt động tức là $a(r_i) = 1$ và gán M_0 cho M_{best}. Sau 17 bước lặp, thuật toán HS trả về một phép ghép hoàn chỉnh M_{best} = $\{(r_1, h_3), (r_2, h_4), (r_3, h_1), (r_4, h_3), (r_5, h_2), (r_6, h_4), (r_7, h_1), (r_8, h_2)\}$ với kích thước 8.

3.3.4. Các kết quả thực nghiệm

Phần này trình bày các kết quả thực nghiệm để so sánh thời gian thực hiện và chất lượng nghiệm tìm được bởi thuật toán HS so với thuật toán AS [67] và thuật toán HP [62].

Bảng 3.4: *Ví dụ thực hiện của thuật toán* HS

Khởi tạo: $M_0 = \{(r_1,h_1),(r_2,h_1),(r_3,h_2),(r_4,h_3),(r_5,\varnothing),(r_6,h_4),(r_7,h_2),(r_8,\varnothing)\}$ $a(r_i)=1, \forall r_i \in \mathcal{R}$ và $M_{best}=M_0$.			
Lặp	r_i	h_j	M
1	r_1	$\varnothing$	$M_1 = \{(r_1,h_1),(r_2,h_1),(r_3,h_2),(r_4,h_3),(r_5,\varnothing),(r_6,h_4),(r_7,h_2),(r_8,\varnothing)\}$
2	r_2	$\varnothing$	$M_2 = \{(r_1,h_1),(r_2,h_1),(r_3,h_2),(r_4,h_3),(r_5,\varnothing),(r_6,h_4),(r_7,h_2),(r_8,\varnothing)\}$
3	r_3	h_1	$M_3 = \{(r_1,\varnothing),(r_2,h_1),(r_3,h_1),(r_4,h_3),(r_5,\varnothing),(r_6,h_4),(r_7,h_2),(r_8,\varnothing)\}$
4	r_4	$\varnothing$	$M_4 = \{(r_1,\varnothing),(r_2,h_1),(r_3,h_1),(r_4,h_3),(r_5,\varnothing),(r_6,h_4),(r_7,h_2),(r_8,\varnothing)\}$
5	r_5	h_2	$M_5 = \{(r_1,\varnothing),(r_2,h_1),(r_3,h_1),(r_4,h_3),(r_5,h_2),(r_6,h_4),(r_7,h_2),(r_8,\varnothing)\}$
6	r_6	$\varnothing$	$M_6 = \{(r_1,\varnothing),(r_2,h_1),(r_3,h_1),(r_4,h_3),(r_5,h_2),(r_6,h_4),(r_7,h_2),(r_8,\varnothing)\}$
7	r_7	$\varnothing$	$M_7 = \{(r_1,\varnothing),(r_2,h_1),(r_3,h_1),(r_4,h_3),(r_5,h_2),(r_6,h_4),(r_7,h_2),(r_8,\varnothing)\}$
8	r_8	h_2	$M_8 = \{(r_1,\varnothing),(r_2,h_1),(r_3,h_1),(r_4,h_3),(r_5,h_2),(r_6,h_4),(r_7,\varnothing),(r_8,h_2)\}$
9	r_1		$M_9 = \{(r_1,h_3),(r_2,h_1),(r_3,h_1),(r_4,h_3),(r_5,h_2),(r_6,h_4),(r_7,\varnothing),(r_8,h_2)\}$
10	r_7	$\varnothing$	$M_{10} = \{(r_1,h_3),(r_2,h_1),(r_3,h_1),(r_4,h_3),(r_5,h_2),(r_6,h_4),(r_7,\varnothing),(r_8,h_2)\}$
11	$\varnothing$	$\varnothing$	M_{10} là phép ổn định và $\lvert M_{10}\rvert = 7$, vì vậy $M_{best} = M_{10}$ HS gọi Thuật toán 3.5 để tạo một phép ghép mới M_{11} và khởi tạo r_1, r_7 và r_8 $M_{11} = \{(r_1,h_3),(r_2,h_4),(r_3,h_1),(r_4,h_3),(r_5,h_2),(r_6,h_4),(r_7,\varnothing),(r_8,h_2)\}$ $a(r_1)=1, a(r_7)=1, a(r_8)=1$.
12	r_1	h_1	$M_{12} = \{(r_1,h_1),(r_2,h_4),(r_3,h_1),(r_4,h_3),(r_5,h_2),(r_6,h_4),(r_7,\varnothing),(r_8,h_2)\}$
13	r_7	$\varnothing$	$M_{13} = \{(r_1,h_1),(r_2,h_4),(r_3,h_1),(r_4,h_3),(r_5,h_2),(r_6,h_4),(r_7,\varnothing),(r_8,h_2)\}$
14	r_8	$\varnothing$	$M_{14} = \{(r_1,h_1),(r_2,h_4),(r_3,h_1),(r_4,h_3),(r_5,h_2),(r_6,h_4),(r_7,\varnothing),(r_8,h_2)\}$
15	$\varnothing$	$\varnothing$	M_{14} là phép ghép ổn định và $\lvert M_{14}\rvert = 7$ HS gọi Thuật toán 3.5 để tạo một phép mới M_{15} và khởi tạo lại r_1 $M_{15} = \{(r_1,\varnothing),(r_2,h_4),(r_3,h_1),(r_4,h_3),(r_5,h_2),(r_6,h_4),(r_7,h_1),(r_8,h_2)\}$ $a(r_1)=1$.
16	r_1	h_3	$M_{16} = \{(r_1,h_3),(r_2,h_4),(r_3,h_1),(r_4,h_3),(r_5,h_2),(r_6,h_4),(r_7,h_1),(r_8,h_2)\}$
17	$\varnothing$	$\varnothing$	M_{16} là phép ghép ổn định và $\lvert M_{16}\rvert = 8$, gán $M_{best} = M_{16}$ Vì $\lvert M_{best}\rvert = n$, HS trả về $M_{best} = M_{16}$

3.3.4.1. So sánh với thuật toán AS

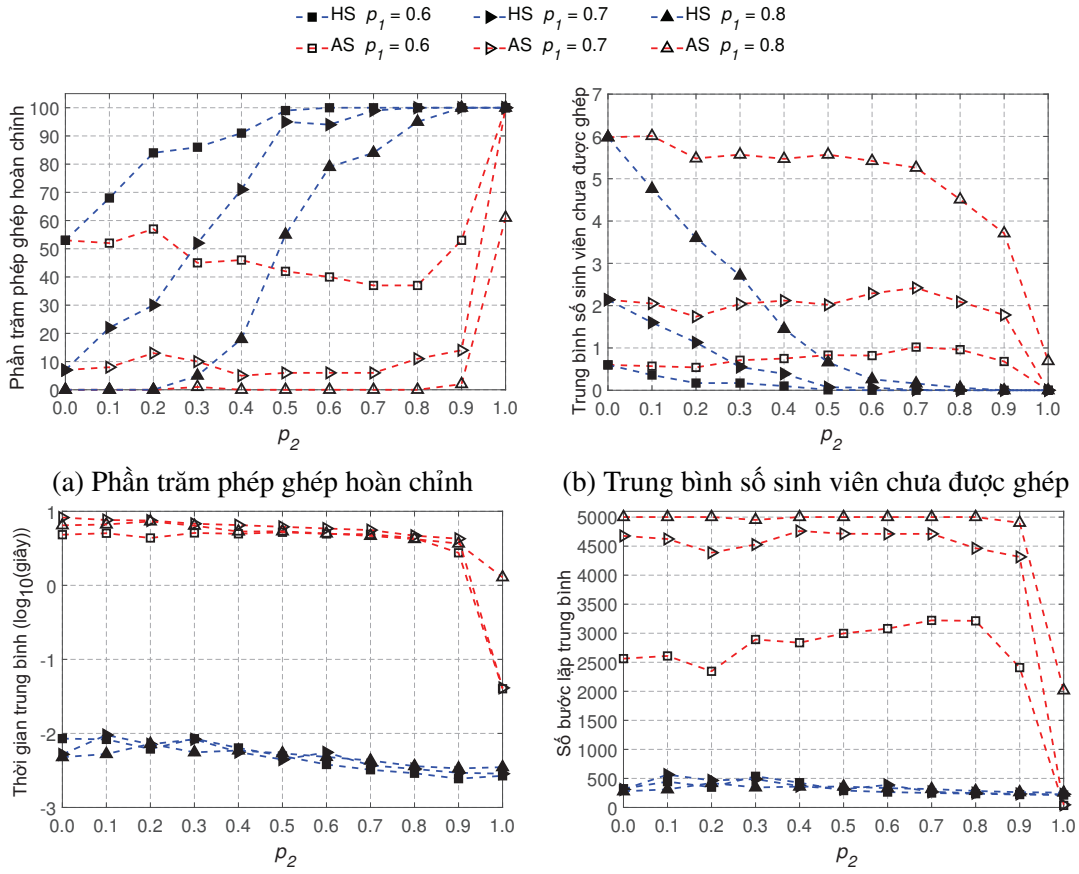
Dữ liệu thực nghiệm là các thể hiện HRT được tạo ngẫu nhiên với các tham số gồm $p_1 \in [0.1, 0.8]$ và $p_2 \in [0.0, 1.0]$ với bước tăng 0.1. Bước lặp tối đa cho cả hai thuật toán HS và AS là 5000.

Thực nghiệm 3.5. Đầu tiên, luận án tạo ngẫu nhiên 100 thể hiện HRT gồm các tham số (p_1, p_2) với kích thước $n = 200$ và $m = 20$. Đối với mỗi thể hiện đã tạo, luận án chọn $c_j = n/m$ cho tất cả $h_j \in \mathcal{H}$ $(j = 1, 2, \cdots, m)$, theo đó, tổng $\sum_{j=1}^{m} c_j = n$. Rõ ràng, nếu $\sum_{j=1}^{m} c_j < n$, thì tồn tại ít nhất một r_i không được ghép cho h_j, có nghĩa là không thể tìm được bất kỳ phép ghép hoàn chỉnh nào cho các thể hiện HRT. Tuy nhiên, nếu $\sum_{j=1}^{m} c_j > n$, thì có thể tìm thấy các phép ghép hoàn chỉnh dễ dàng hơn cho các thể hiện HRT.

Hình 3.4(a) chỉ ra phần trăm phép ghép hoàn chỉnh mà HS và AS tìm thấy khi p_1 thay đổi từ 0.6 đến 0.8 và p_2 thay đổi từ 0.0 đến 1.0. Cần lưu ý rằng khi p_1 thay đổi từ 0.1 đến

0.5, cả HS và AS đều tìm thấy 100% phép ghép hoàn chỉnh và do đó, luận án không chỉ ra kết quả cho các giá trị p_1 trong hình này. Chúng ta thấy rằng khi p_1 tăng từ 0.6 lên 0.8, tỷ lệ phần trăm phép ghép hoàn chỉnh mà cả HS và AS tìm được giảm. Khi p_2 = 0, nghĩa là không xuất hiện thứ tự ưu tiên ngang bằng trong danh sách xếp hạng, thì cả hai thuật toán HS và AS tìm được cùng một tỷ lệ phần trăm phép ghép hoàn chỉnh. Khi p_2 thay đổi từ 0.1 đến 1.0, HS tìm được tỷ lệ phần trăm phép ghép hoàn chỉnh cao hơn so với AS tìm được. Hình 3.4(b) chỉ ra số lượng trung bình số người chưa được ghép trong các phép ghép ổn định. Khi $p_2 = 0$, HS và AS tìm được cùng một số lượng người chưa được ghép trong các phép ghép ổn định vì tất cả các phép ghép ổn định đều có cùng kích thước. Khi p_2 thay đổi từ 0.1 đến 1.0, HS tìm được số lượng người chưa được ghép nhỏ hơn so với AS tìm được trong các phép ghép không hoàn chỉnh. Điều này có nghĩa là HS tìm được các phép ghép ổn định có kích thước tối đa tốt hơn so với AS. Điều này là bởi vì trong quá trình tìm kiếm cặp chặn trội nhất, HS xem xét h_j trong danh sách xếp hạng của r_i nếu có thứ tự ưu tiên xếp hạng ngang bằng, HS sẽ ưu tiên xét những h_j đang còn thiếu nhiều vị trí trước. Ngoài ra, HS sử dụng hàm "*Improve*" để cải tiến kích thước phép ghép ổn định khi chưa hoàn chỉnh bằng cách xem xét định hướng chọn h_j hoặc r_i để thay thế dựa vào xếp hạng ngang bằng thay vì chọn một cặp ngẫu nhiên như thuật toán AS.

Hình 3.4(c) chỉ ra thời gian thực hiện trung bình của HS và AS. Khi p_1 tăng từ 0.6 lên 0.8, thời gian thực hiện trung bình của HS và AS tăng không đáng kể. Khi p_2 tăng từ 0.0 lên 0.9, thời gian thực hiện trung bình của HS và AS giảm. Khi p_2 = 1.0, thời gian thực hiện trung bình của HS xấp xỉ như trường hợp $p_2 = 0.9$, trong khi thời gian thực hiện trung bình của AS giảm đáng kể. Chúng ta có thể thấy rằng thời gian thực hiện trung bình của HS tăng từ khoảng $10^{-2.5}$ giây lên $10^{-2.1}$ giây, trong khi thời gian của AS tăng từ khoảng $10^{-1.4}$ giây đến $10^{0.9}$ giây cho mọi giá trị của p_1 và p_2. Điều này có nghĩa là HS chạy nhanh hơn AS khoảng 12 đến 1000 lần. Bởi vì AS tìm một tập các cặp chặn cặp trội nhất và lựa chọn một cặp chặn trội nhất để loại bỏ trong tập các cặp chặn cặp trội nhất dựa vào hàm đánh giá lỗi. Nếu một phép ghép ổn định tìm được chưa đạt được kích thước tối đa, thuật toán AS sẽ cải tiến lại kích thước phép ghép bằng hàm Reset mà không cần khởi tạo lại một ghép ngẫu nhiên. Bằng cách làm như vậy, AS có thể tìm thấy phép ghép ổn định với kích thước tối đa hiệu quả, tuy nhiên thuật toán này cần $O(n^2)$ tìm cặp chặn và thêm $O(n^2)$ để tính toán giá trị của các phép ghép mới trong quá trình tìm cặp chặn trội nhất. Trong khi đó, thuật toán HS chỉ mất $O(n)$ thời gian để tìm một cặp chặn trội nhất thay vì tìm tập các cặp chặn cặp trội nhất như thuật toán AS. Hình 3.4(d) chỉ ra số lần lặp trung bình được sử dụng bởi HS và AS. Khi p_2 thay đổi từ 0.1 đến 1.0, số lần lặp lại trung bình của HS chỉ thay đổi ít, trong khi của AS giảm đáng kể



(a) Phần trăm phép ghép hoàn chỉnh

(b) Trung bình số sinh viên chưa được ghép

(c) Trung bình thời gian thực hiện

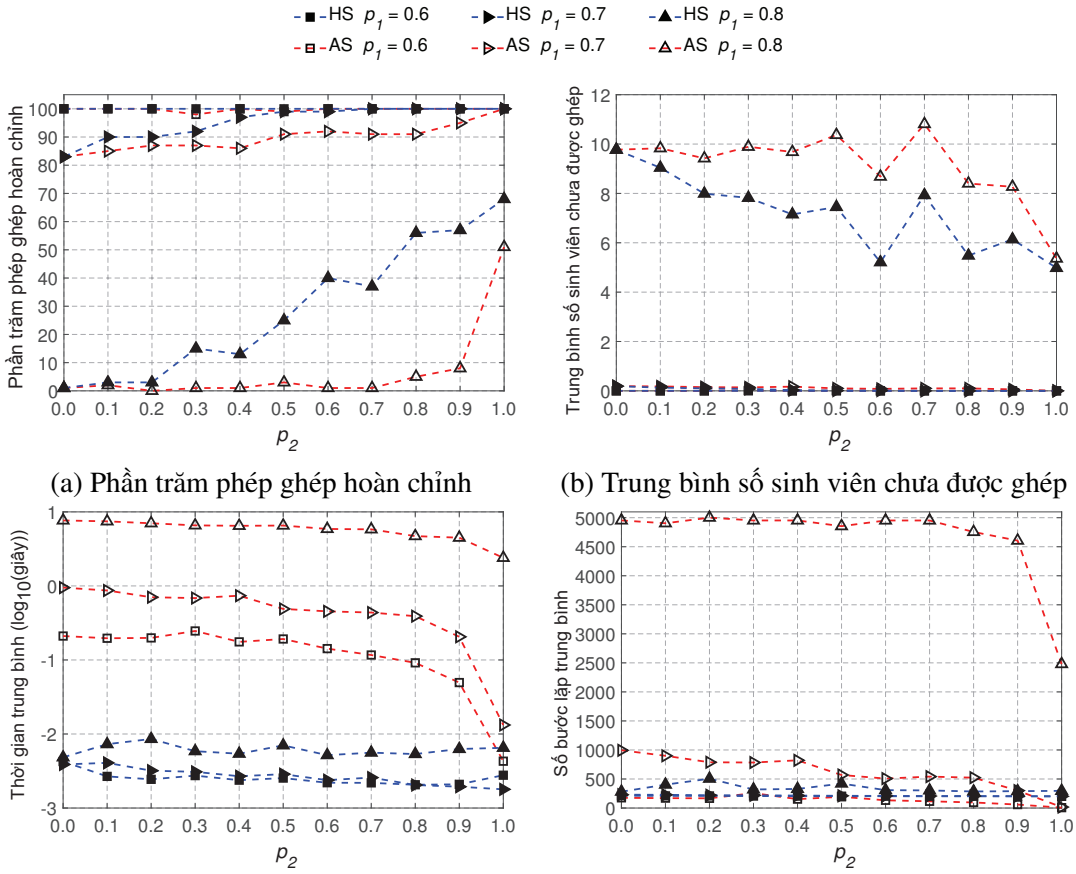
(d) Trung bình số bước lặp

Hình 3.4: HS *và* AS *với* $n = 200$, $m = 20$ *và* $c_j = n/m$

cho $p_2 \in [0.9, 1.0]$. Chúng ta thấy rằng HS sử dụng số lần lặp ít hơn nhiều so với số lần lặp của HS để tìm các phép ghép ổn định. Điều này giải thích rằng HS chạy nhanh hơn so với AS trong Hình 3.4(a).

Thực nghiệm 3.6. Trong thực nghiệm này, luận án tạo ngẫu nhiên 100 thể hiện HRT cho các giá trị tham số gồm n, m, p_1 và p_2 được tạo ra trong *Thực nghiệm 3.5*, nhưng c_j của h_j trong mỗi thể hiện được tạo thành bởi một số nguyên ngẫu nhiên trong đoạn $[0.1q, 0.4q]$, trong đó q là tổng số sinh viên được xếp hạng cho tất cả $h_j \in \mathcal{H}$ $(j = 1, 2, \cdots, m)$. Điều này có thể hiểu rằng h_j xếp hạng q sinh viên nhưng chọn từ 10% đến 40% sinh viên trong danh sách xếp hạng. Chúng ta thấy rằng khi $p_1 = 0.6$ hoặc $p_1 = 0.7$, cả HS và AS đều dễ dàng tìm được các phép ghép ổn định hơn so với khi $c_j = n/m$ và do đó, cả HS và AS sử dụng ít thời gian và số lần lặp hơn để tìm các phép ghép ổn định như được chỉ ra trong Hình 3.5. Chúng ta thấy rằng HS không chỉ tìm được phần trăm phép hoàn chỉnh cao hơn so với phần trăm phép ghép của AS tìm được, mà còn có số lượng sinh viên chưa được ghép nhỏ hơn của AS trong các phép ghép ổn định. Bên cạnh đó, HS chạy nhanh hơn và sử dụng ít lần lặp hơn so với AS.

Thực nghiệm 3.7. Trong thực nghiệm này, luận án thay đổi số sinh viên và doanh



(a) Phần trăm phép ghép hoàn chỉnh

(b) Trung bình số sinh viên chưa được ghép

(c) Trung bình thời gian thực hiện

(d)Trung bình số bước lặp

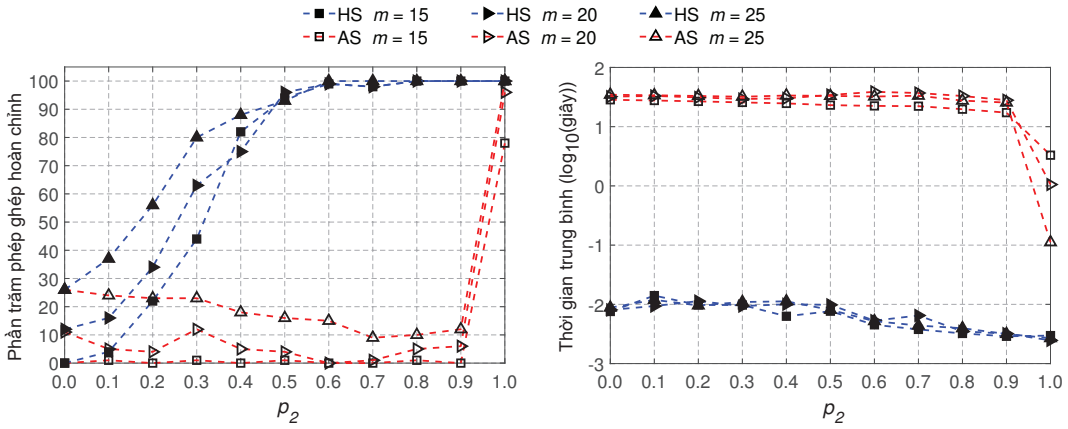
Hình 3.5: HS *và* AS *với* $n = 200$, $m = 20$ *và* $c_j = [0.1q, 0.4q]$

nghiệp của các thể hiện HRT để xem xét hiệu quả của HS và AS. Cụ thể, luận án tạo ngẫu nhiên 100 thể hiện HRT cho mỗi bộ tham số $n = 300$, $m = \{15, 20, 25\}$, $p_1 = 0.7$, $p_2 = \{0.0, 0.1, \cdots, 1.0\}$. Điều này có nghĩa là mỗi sinh viên xếp hạng doanh nghiệp khoảng 50, 67 và 83 khi $m = \{15, 20, 25\}$ và mỗi doanh nghiệp xếp hạng khoảng 90 sinh viên. Chúng tôi đặt $c_j = n/m$ cho tất cả $h_j \in \mathcal{H}$ $(j = 1, 2, \cdots, m)$. Hình 3.6 chỉ ra kết quả của thực nghiệm này. Khi m tăng từ 15 lên 25, cả HS và AS đều tìm được các phép ghép ổn định dễ dàng hơn nhiều vì tồn tại nhiều cặp chấp nhận trong các thể hiện đã tạo. Từ đó, chúng ta thấy rằng HS vượt trội hơn AS trong cả việc tìm kiếm các phép ghép hoàn chỉnh và thời gian thực hiện.

3.3.4.2. So sánh với thuật toán HP

Phần này trình bày các thực nghiệm để so sánh tỷ lệ phần trăm phép ghép hoàn chỉnh và thời gian thực hiện trung bình của thuật toán HS so với HP [62] cho các thể hiện HRT.

Thực nghiệm 3.8. Đầu tiên, luận án chạy cả HS và HP cho các thể hiện được tạo ra trong *Thực nghiệm 3.5* và *3.6*. Các kết quả của thực nghiệm này được chỉ ra trong Hình 3.7. Khi $p_2 = 0$, cả HS và HP đều tìm được cùng một tỷ lệ phần phép ghép hoàn



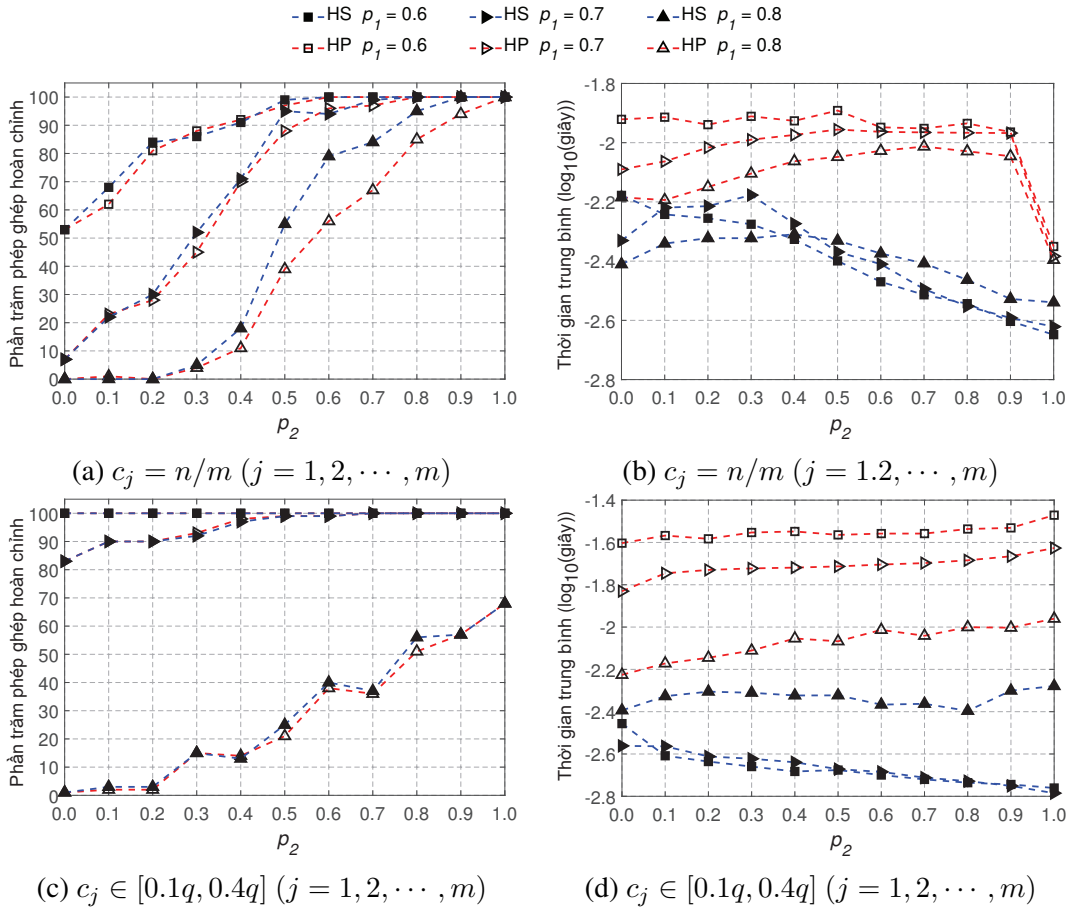
(a) Phần trăm phép ghép hoàn chỉnh (b) Trung bình thời gian thực hiện

Hình 3.6: HS *và* AS *với* $n = 300$, $n = \{15, 20, 25\}$ *và* $c_j = n/m$

chỉnh, tất cả các phép ghép ổn định đều có cùng kích thước. Khi p_2 thay đổi từ 0.1 đến 1.0, chúng ta có 60 điểm để so sánh tỷ lệ phần phép ghép hoàn chỉnh tìm được bởi HS và HP, nhưng HP chỉ tìm được tỷ lệ phần phép ghép hoàn chỉnh cao hơn so với của HS ở 5 điểm ($\approx 8.3\%$) bao gồm $\{(200, 20, 0.6, 0.3), (200, 20, 0.7, 0.6)\}$ khi $c_j = n/m$ và $\{(200, 20, 0.7, 0.3), (200, 20, 0.7, 0.4), (200, 20, 0.8, 0.4)\}$ khi $c_j = [0.1q, 0.4q]$. Bên cạnh đó, thực nghiệm này chỉ ra rằng thời gian thực hiện trung bình được sử dụng bởi HS nhỏ hơn thời gian thực hiện được sử dụng bởi HP.

Thực nghiệm 3.9. Kết quả Thực nghiệm 3.8 chỉ ra HS hiệu quả hơn HP về thời gian thực hiện và chất lượng nghiệm khi n nhỏ. Do đó, trong thực nghiệm này, chúng tôi chọn $n = 1000$, $m = 25$, $p_1 = \{0.7, 0.8, 0.9\}$ và $p_2 = \{0.0, 0.1, \cdots, 1.0\}$. Đối với bộ các tham số n, m, p_1 và p_2, luận án tạo ngẫu nhiên 100 thể hiện HRT với $c_j = n/m$ và 100 thể hiện HRT với c_j $(j = 1, 2, \cdots, m)$ là một số nguyên ngẫu nhiên trong đoạn $[30, 55]$. Số lần lặp tối đa của HS trong thực nghiệm này là 10000.

Hình 3.8(a) và 3.8(c) chỉ ra phần trăm phép ghép hoàn chỉnh tìm được bởi HS và HP. Chúng ta thấy rằng khi p_1 tăng từ 0.7 lên 0.9, tỷ lệ phần trăm phép ghép hoàn chỉnh của HS và HP tìm được giảm vì mỗi sinh viên xếp hạng một vài doanh nghiệp và mỗi doanh nghiệp xếp hạng số lượng sinh viên ít hơn. Khi p_2 tăng từ 0.0 lên 1.0, tỷ lệ phần trăm phép ghép hoàn chỉnh mà HS và HP tìm được tăng lên vì mỗi sinh viên có thể được ghép vào một số doanh nghiệp có mức xếp hạng ưu tiên ngang bằng trong danh sách xếp hạng mà không tạo thành các cặp chặn. Hình 3.8(b) và 3.8(d) chỉ ra thời gian thực hiện trung bình của HS và HP. Khi p_1 tăng từ 0.7 lên 0.9, thời gian thực hiện trung bình của HS chỉ tăng nhẹ, trong khi thời gian thực hiện của HP giảm xuống. Khi p_2 tăng từ 0.0 lên 1.0, thời gian thực hiện trung bình của HS giảm xuống, trong khi thời gian thực hiện trung bình của HP hầu như không thay đổi,



(a) $c_j = n/m$ $(j = 1, 2, \cdots, m)$

(b) $c_j = n/m$ $(j = 1.2, \cdots, m)$

(c) $c_j \in [0.1q, 0.4q]$ $(j = 1, 2, \cdots, m)$

(d) $c_j \in [0.1q, 0.4q]$ $(j = 1, 2, \cdots, m)$

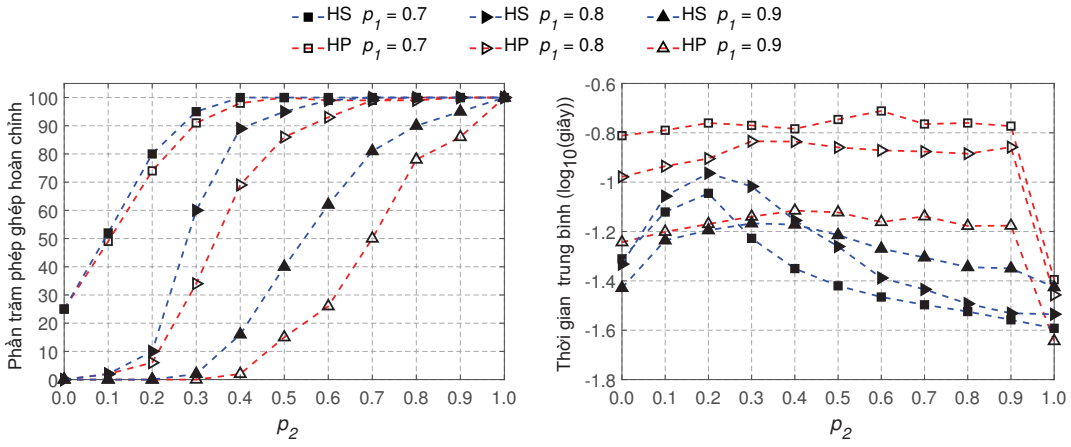
Hình 3.7: HS *và* HP *với* n = *200,* m = *20*

ngoại trừ $p_2 = 1.0$, thời gian thực hiện trung bình thời gian của HP giảm đáng kể.

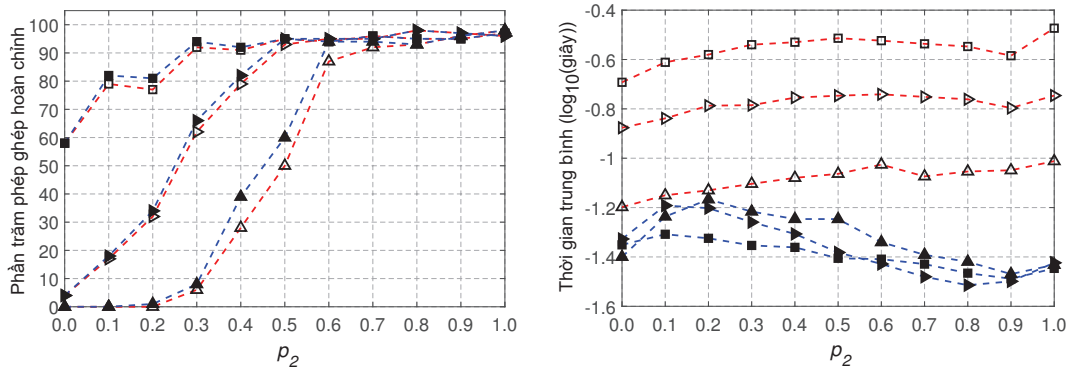
Thực nghiệm này chỉ ra rằng HS không chỉ tìm được tỷ lệ phần trăm phép ghép hoàn chỉnh cao hơn so với HP mà còn chạy nhanh hơn nhiều so với HP cho dù $c_j = n/m$ hoặc c_j là một số nguyên ngẫu nhiên trong đoạn $[30, 55]$.

Thực nghiệm 3.10. Trong thực nghiệm này, luận án thay đổi số lượng doanh nghiệp để so sánh hiệu quả của HS và HP. Cụ thể, luận án tạo ngẫu nhiên 100 thể hiện HRT cho mỗi kết hợp $n = 1000$, $m = 50, 75, 100$, $p_1 = 0.9$ và $p_2 = \{0.0, 0.1, \cdots, 1.0\}$. Đối với mỗi thể hiện đã tạo, luận án chọn $c_j = n/m$ cho tất cả $h_j \in \mathcal{H}$ $(j = 1, 2, \cdots, m)$. Điều này có nghĩa là $\sum_{j=1}^{m} c_j = n$, tức là chỉ tồn tại vị trí cho mỗi sinh viên trong doanh nghiệp. Hình 3.9 chỉ ra kết quả của thực nghiệm này. Một lần nữa, chúng ta thấy rằng HS vượt trội hơn HP về thời gian thực hiện và chất lượng nghiệm.

Thực nghiệm 3.11. Cuối cùng, luận án chọn $n = 5000$, $m = 50, 100, 150$, $p_1 = 0.95$, $p_2 = \{0.0, 0.1, \cdots, 1.0\}$ và $c_j = n/m$ $(j = 1, 2, \cdots, m)$ cho thực nghiệm này. Đối với các tham số n, m, p_1 và p_2, luận án tạo ngẫu nhiên 100 thể hiện HRT và tính trung bình các kết quả. Luận án chọn số lần lặp tối đa trong HRT là 25000. Hình 3.10 chỉ ra kết quả của thực nghiệm này. Chúng ta thấy rằng khi $m = 50$ hoặc $m = 100$, HS tìm được tỷ lệ phần

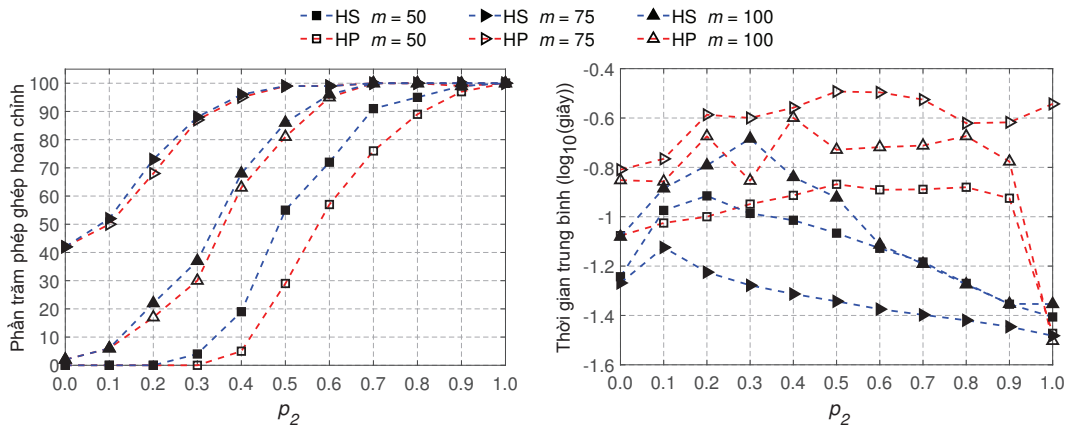


(a) $c_j = n/m$ $(j = 1, 2, \cdots, m)$ (b) $c_j = n/m$ $(j = 1, 2, \cdots, m)$



(c) $c_j = [30, 55]$ $(j = 1, 2, \cdots, m)$ (d) $c_j = [30, 55]$ $(j = 1, 2, \cdots, m)$

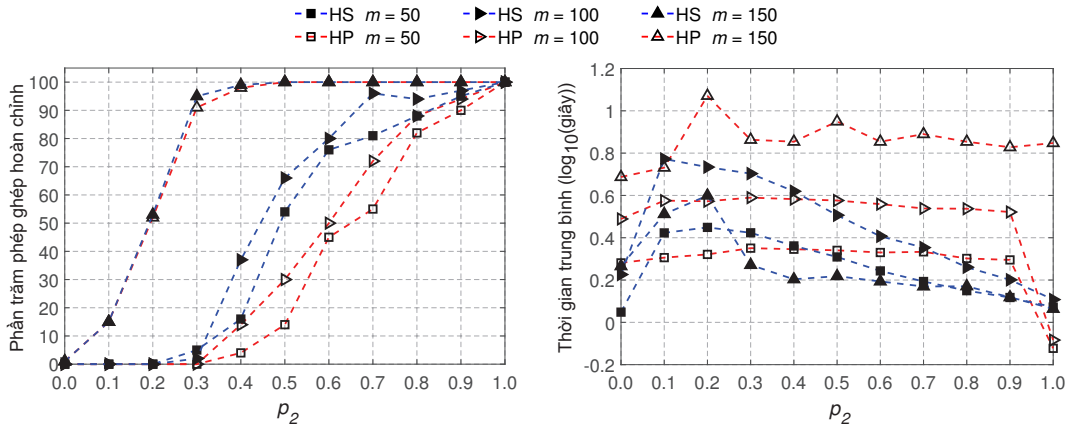
Hình 3.8: HS *và* HP *với* $n = 1000$, $m = 25$



(a) Phần trăm phép ghép hoàn chỉnh (b) Trung bình thời gian thực hiện

Hình 3.9: HS *và* HP *với* $n = 1000$, $m = 50, 75$ *và* 100

trăm phép ghép hoàn chỉnh cao hơn nhiều và sử dụng ít thời gian hơn nhiều so với HP. Khi $m = 150$, thuật toán HS tìm được tỷ lệ phần trăm phép ghép hoàn chỉnh xấp xỉ bằng với thuật toán HP tìm được, nhưng HS chạy nhanh hơn nhiều so với HP. Điều này có nghĩa là HS vượt trội so với HP về thời gian thực hiện và chất lượng nghiệm để giải quyết bài toán MAX-HRT kích thước lớn.



(a) Phần trăm phép ghép hoàn chỉnh (b) Thời gian thực hiện trung bình

Hình 3.10: HS *và* HP *với* $n = 5000$

3.4. Kết luận Chương 3

Chương này đã trình bày hai thuật toán gồm MCA và HS để giải quyết bài toán MAX-HRT với kích thước lớn. Mục đích của bài toán là tìm một phép ghép ổn định giữa sinh viên và doanh nghiệp với kích thước tối đa.

MCA dựa trên ý tưởng thuật toán xung đột tối thiểu [112], để tìm phép ghép ổn định với kích thước tối đa cho HRT. Ý tưởng chính thuật toán là xuất phát từ một phép ghép ngẫu nhiên, tại mỗi bước tìm kiếm, thuật toán chọn một doanh nghiệp để ghép cho một sinh viên sao cho số cặp chặn là nhỏ nhất. Bằng cách như vậy, thuật toán MCA nhanh chóng tìm được nghiệm là phép ghép hoàn chỉnh với thời gian thực hiện vượt trội so với thuật toán LTIU [45] cho bài toán MAX-HRT với kích thước lớn. Các kết quả thực nghiệm với các bộ dữ liệu tạo ngẫu nhiên chỉ ra rằng, MCA vượt trội hơn thuật toán LTIU về cả chất lượng phần trăm phép ghép hoàn chỉnh và thời gian thực hiện cho bài toán MAX-HRT kích thước lớn.

HS bắt đầu từ một phép ghép ngẫu nhiên, thuật toán xem xét lần lượt các sinh viên và định nghĩa một hàm heuristic để định hướng tìm cặp chặn trội nhất nhanh nhất. Sau đó, thuật toán loại các cặp chặn trội nhất ra khỏi phép ghép ban đầu. Nếu tìm được phép ghép ổn định mà chưa hoàn chỉnh thì thuật toán tiếp tục cải thiện kích thước của phép ghép ổn định để đạt được một phép ổn định với kích thước tối đa. Các kết quả thực nghiệm chỉ ra rằng thuật toán HS vượt trội hơn thuật toán AS và HP về thời gian thực hiện và chất lượng nghiệm cho bài toán MAX-HRT kích thước lớn.

CHƯƠNG 4.
ĐỀ XUẤT THUẬT TOÁN GIẢI BÀI TOÁN MAX-SPA

Chương này đề xuất hai thuật toán tìm kiếm heuristic để giải quyết bài toán MAX-SPA, tức là tìm một phép ghép ổn định với kích thước tối đa cho các thể hiện của bài toán SPA-P và SPA-ST. Các thuật toán này đã được công bố tại công trình [A.6], [B.1] và [A.4] trong phần "*Các công bố sử dụng trong Luận án*".

4.1. Giới thiệu

Bài toán SPA là một ứng dụng mở rộng quan trọng của bài toán SMTI [1, 53]. Một số biến thể của bài toán SPA được đề xuất bao gồm SPA-S [16], SPA-P [60, 19], SPA-ST [3, 60, 90, 91], trong đó, bài toán SPA-P và SPA-ST đã được cộng đồng nghiên cứu quan tâm nhiều vì những ứng dụng trong thực tế [94, 113, 114, 115, 116, 93]. Mục đích của bài toán là tìm một phép ghép ổn định yếu với kích thước tối đa, gọi là bài toán MAX-SPA [3, 17, 16]. Tuy nhiên, bài toán MAX-SPA là NP-khó [55, 27] và do vậy việc tìm ra một thuật toán hiệu quả để giải quyết bài toán MAX-SPA là một thách thức đối với các nhà nghiên cứu. Với việc xuất hiện thêm các ràng buộc từ phía giảng viên và đề tài, các thuật toán đã giải quyết cho các bài toán MAX-SMTI và MAX-HRT là không hiệu quả cho bài toán MAX-SPA. Gần đây, có nhiều thuật toán xấp xỉ [3, 17, 16, 116, 117, 94, 92, 104] và thuật toán cục bộ [11, 118] đã được đề xuất để giải quyết bài toán MAX-SPA, tuy nhiên các thuật toán trước đây đề xuất thường giải quyết với bài toán MAX-SPA có kích thước nhỏ. Dựa trên tình hình nghiên cứu của bài toán MAX-SPA, Chương này đề xuất hai thuật toán tìm kiếm heuristic để giải quyết hiệu quả bài toán MAX-SPA-P và MAX-SPA-ST.

4.2. Đề xuất thuật toán SPA-P-heuristic giải bài toán MAX-SPA-P

4.2.1. Ý tưởng

Phần này trình bày một thuật toán heuristic (viết tắt SPA-P-heuristic) để giải quyết bài toán MAX-SPA-P. Ý tưởng chính của SPA-P-heuristic là khởi tạo một phép ghép rỗng và thiết lập các sinh viên ở trạng thái hoạt động. Ở mỗi bước lặp, SPA-P-heuristic ghép một sinh viên

với một đề tài có thứ tự tiên cao nhất trong danh sách xếp hạng của sinh viên và thiết lập sinh viên sang trạng thái không hoạt động. Nếu một đề tài hoặc giảng viên được ghép quá số lượng sinh viên tối đa, SPA-P-heuristic xác định một sinh viên tồi nhất dựa trên một hàm heuristic để loại bỏ khỏi phép ghép hiện tại và sinh viên được thiết lập trở lại trạng thái hoạt động. SPA-P-heuristic kết thúc khi tất cả các sinh viên ở trạng thái không hoạt động và thuật toán trả về một phép ghép ổn định với kích thước tối đa.

4.2.2. Hàm heuristic

Xét các thuật toán SPA-P-approx [60] và SPA-P-promotion [102]. Bắt đầu một phép ghép được khởi tạo là $M = \varnothing$. Ở mỗi bước lặp, khi một sinh viên $s_i \in \mathcal{S}$ được ghép cho một đề tài $p_j \in \mathcal{P}$ có ưu tiên cao nhất trong danh sách xếp hạng của s_i để tạo thành một cặp $(s_i, p_j) \in M$, nếu đề tài p_j hoặc giảng viên $l_k \in \mathcal{L}$ đề xuất p_j đã được ghép quá số lượng sinh viên tối đa thì một sinh viên tùy ý $s_r \in M(p_z)$, trong đó p_z là đề tài mà l_k xếp hạng ưu tiên thấp nhất bị loại khỏi M. Như vậy, nếu ba điều kiện sau được thỏa mãn: (*i*) $M(p_z)$ có ít nhất hai sinh viên $s_r \in \mathcal{S}$ và $s_t \in \mathcal{S}$; (*ii*) s_r chỉ xếp hạng một đề tài; và (*iii*) s_t xếp hạng nhiều hơn một đề tài, và nếu chúng ta xóa s_r khỏi M, thì s_r sẽ không được ghép trong M, trong khi nếu xóa s_t khỏi M thì s_t có thể được ghép cho một đề tài nào đó trong danh sách xếp hạng của s_t ở các lần lặp tiếp theo. Để giải quyết vấn đề này, luận án đề xuất một hàm heuristic như sau:

$$h_{l_k}(s_t) = rank(l_k, p_z) + y(s_t)/(q+1), \tag{4.1}$$

trong đó l_k là giảng viên đề xuất đề tài p_z và $y(s_t)$ là số lượng đề tài được xếp hạng bởi s_t. Theo đó, thay vì loại bỏ một sinh viên tùy ý $s_r \in M(p_z)$, nếu đề tài p_j được ghép quá số lượng sinh viên tối đa thì chúng ta loại bỏ một sinh viên $s_t \in M(p_z)$ tương ứng với giá trị lớn nhất của $h_{l_k}(s_t)$ và nếu l_k được ghép vượt quá số lượng sinh viên tối đa thì loại bỏ một sinh viên $s_t \in M(l_k)$ tương ứng với giá trị lớn nhất của $h_{l_k}(s_t)$. Luận án gọi một sinh viên s_t tương ứng với giá trị lớn nhất của $h_{l_k}(s_t)$ là sinh viên có ưu tiên thấp nhất được ghép cho đề tài $p_z \in M(p_z)$ hoặc $p_z \in M(l_k)$. Rõ ràng, s_t không chỉ được ghép cho đề tài p_z mà l_k đề xuất xếp hạng ưu tiên thấp nhất (tức là $max(rank(l_k, p_z))$, mà s_t còn xếp hạng nhiều đề tài nhất (tức là $max(y(s_t))$). Bằng cách loại bỏ một sinh viên s_t, thuật toán không chỉ giữ cho M ổn định mà còn giữ lại những sinh viên trong M có ít cơ hội được ghép lại cho một đề tài nào đó trong danh sách của họ ở các lần lặp tiếp theo.

Cho một ví dụ thể hiện của SPA-P trong Bảng 4.1 gồm 5 sinh viên, 2 giảng viên và 5 đề tài. Giả sử tại một bước lặp nào đó, các thuật toán SPA-P-approx và SPA-P-promotion tìm

Bảng 4.1: *Một thể hiện của* SPA-P

Danh sách xếp hạng của $s_i \in \mathcal{S}$	Danh sách xếp hạng của $l_k \in \mathcal{L}$
s_1: p_1 p_3 p_4	l_1: p_1 p_2 p_3
s_2: p_5 p_1	l_2: p_4 p_5
s_3: p_2 p_5	
s_4: p_4 p_2	
s_5: p_5	
Số sinh viên tối đa của $p_j \in \mathcal{P}$: $c_1 = c_2 = c_3 = c_4 = 1$, $c_5 = 2$	
Số sinh viên tối đa của $l_k \in \mathcal{L}$: $d_1 = 3$, $d_2 = 2$	

được phép ghép $M = \{(s_1, p_1), (s_2, p_5), (s_3, p_2), (s_4, p_4), (s_5, p_5)\}$. Vì $M(l_2) = \{s_2, s_4, s_5\}$ và $|M(l_2)| > d_2$, các thuật toán này sẽ loại ngẫu nhiên một cặp ghép là $(s_2, p_5) \in M$ hoặc $(s_5, p_5) \in M$ vì l_2 xếp hạng ưu tiên p_5 thấp nhất và $M(p_5) = \{s_2, s_5\}$. Nếu loại cặp ghép $(s_5, p_5) \in M$ thì s_5 sẽ không được ghép với đề tài nào khác. Tuy nhiên với hàm heuristic đã đề xuất, ta có: $h_{l_2}(s_2)$ = 2.3, $h_{l_2}(s_4)$ = 1.3, và $h_{l_2}(s_5)$ = 2.2. Vì $h_{l_2}(s_2)$ có giá trị lớn nhất nên cặp ghép $(s_2, p_5) \in M$ sẽ bị loại và khi đó M = {(s_1, p_1), (s_2, $\varnothing$), (s_3, p_2), (s_4, p_4), (s_5, p_5)}, tức là s_2 trở thành chưa được ghép, do đó s_2 có thể đề xuất đề tài p_1 trong danh sách xếp hạng của s_2.

4.2.3. Mô tả thuật toán

Thuật toán SPA-P-heuristic được mô tả trong Thuật toán 4.1. Trong khi thực hiện thuật toán, mỗi sinh viên được thiết lập một trong hai trạng thái là đang hoạt động (tức là $a(s_i) = 1$) hoặc không hoạt động (tức là $a(s_i) = 0$). Khi bắt đầu, tất cả sinh viên $s_i \in \mathcal{S}$ được thiết lập trạng thái đang hoạt động và chưa được ghép trong M (dòng 2-3). Để tăng tốc độ tính toán các giá trị heuristic của $h_{l_k}(s_i)$ cho $\forall l_k \in \mathcal{L}$ và $\forall s_i \in \mathcal{S}$, thuật toán định nghĩa một bảng $h_{l_k}(s_i)$ gồm m giảng viên và n sinh viên và khởi tạo $h_{l_k}(s_i) = 0$ cho $\forall l_k \in \mathcal{L}$ và $\forall s_i \in \mathcal{S}$ (dòng 4). Tại mỗi bước lặp lại, nếu s_i đang hoạt động và không có đề tài nào trong danh sách xếp hạng thì s_i trở nên vĩnh viễn không hoạt động (tức là $a(s_i) = 0$), do đó s_i không được ghép trong M (dòng 6-8). Ngược lại, s_i được ghép cho đề tài thích nhất p_j và s_i trở thành không hoạt động (dòng 9-12). Khi s_i được ghép cho một đề tài p_j, giá trị $h_{l_k}(s_i)$, trong đó l_k đề xuất p_j, được cập nhật để sử dụng ở các lần lặp tiếp theo (dòng 13-14). Nếu p_j được ghép quá số lượng sinh viên tối đa ($|M(p_j)| > c_j$) thì $s_t \in M(p_j)$ có xếp hạng ưu tiên thấp nhất bởi p_j bị loại khỏi M, s_t xóa p_j trong danh sách xếp hạng và hoạt động trở lại (dòng 15-20). Nếu l_k được ghép quá số lượng sinh viên tối đa ($|M(l_k)| > d_k$) thì $s_t \in M(l_k)$ có xếp hạng ưu tiên thấp nhất bởi l_k bị loại khỏi M, s_t xóa p_z trong danh xếp hạng của s_t, trong

Algorithm 4.1: Thuật toán SPA-P-heuristic

Input: Thể hiện I của SPA-P.
Output: Phép ghép ổn định M.

1. **function** `SPA-P-heuristic(I)`
2. $M := \emptyset$;
3. $a(s_i) := 1, \forall s_i \in \mathcal{S}$;
4. $h_{l_k}(s_i) := 0, \forall l_k \in \mathcal{L}, \forall s_i \in \mathcal{S}$;
5. **while** $\exists\ s_i$ *với* $a(s_i) = 1$ **do**
6. **if** $(\nexists p_j \in \mathcal{P} | rank(s_i, p_j) > 0)$ **then**
7. $a(s_i) := 0$;
8. continue;
9. $p_j := argmin(rank(s_i, p_j) > 0)$;
10. $l_k :=$ giảng viên đề xuất đề tài p_j;
11. $M := M \cup \{(s_i, p_j)\}$;
12. $a(s_i) := 0$;
13. $y(s_i) :=$ số lượng đề tài được xếp hạng bởi s_i;
14. $h_{l_k}(s_i) := rank(l_k, p_j) + y(s_i)/(q+1)$;
15. **if** $|M(p_j)| > c_j$ **then**
16. $s_t := \text{argmax}(h_{l_k}(s_t)), \forall s_t \in M(p_j)$;
17. $M := M \setminus \{(s_t, p_j)\}$;
18. $rank(s_t, p_j) := 0$;
19. $a(s_t) := 1$;
20. $h_{l_k}(s_t) := 0$;
21. **if** $|M(l_k)| > d_k$ **then**
22. $s_t := \text{argmax}(h_{l_k}(s_t)), \forall s_t \in M(l_k)$;
23. $p_z := M(s_t)$;
24. $M := M \setminus \{(s_t, p_z)\}$;
25. $rank(s_t, p_z) := 0$;
26. $a(s_t) := 1$;
27. $h_{l_k}(s_t) := 0$;
28. **return** M;
29. **end function**

đó p_z được ghép cho s_t, và s_t sẽ hoạt động trở lại. Khi s_t bị xóa khỏi M, giá trị $h_{l_k}(s_t) = 0$ (dòng 21-27). Thuật toán lại cho đến khi tất cả sinh viên ở trạng thái không hoạt động và trả về phép ghép ổn định với kích thước tối đa tìm được trong các bước lặp.

Chú ý rằng trong trường hợp tốt nhất, nếu mỗi sinh viên $s_i \in \mathcal{S}$ đề xuất đề tài $p_j \in \mathcal{P}$ có thứ tự ưu tiên cao nhất trong danh sách xếp hạng và ghép với đề tài đó để tạo thành một phép ghép ổn định, thì SPA-P-heuristic cần $O(n)$ thời gian, trong đó n là số lượng sinh viên. Trong trường hợp xấu nhất, nếu mỗi sinh viên $s_i \in \mathcal{S}$ đề xuất tất cả các đề tài $p_j \in \mathcal{P}$ trong danh sách xếp hạng, thì SPA-P-heuristic cần $O(T)$ thời gian, trong đó T là tổng độ dài của tất cả danh sách của xếp hạng sinh viên.

Bảng 4.2: *Ví dụ thực hiện của thuật toán* SPA-P-*heuristic*

Lặp	s_i	p_j	l_k	*Phép ghép*
1	s_1	p_1	l_1	$M = \{(s_1, p_1)\}$
2	s_2	p_5	l_2	$M = \{(s_1, p_1), (s_2, p_5)\}$
3	s_3	p_2	l_1	$M = \{(s_1, p_1), (s_2, p_5), (s_3, p_2)\}$
4	s_4	p_4	l_2	$M = \{(s_1, p_1), (s_2, p_5), (s_3, p_2), (s_4, p_4)\}$
5	s_5	p_5	l_2	$M = \{(s_1, p_1), (s_3, p_2), (s_4, p_4), (s_5, p_5)\}$
6	s_2	p_1	l_1	$M = \{(s_2, p_1), (s_3, p_2), (s_4, p_4), (s_5, p_5)\}$
7	s_1	p_3	l_1	$M = \{(s_1, p_3), (s_2, p_1), (s_3, p_2), (s_4, p_4), (s_5, p_5)\}$

4.2.4. Ví dụ

Xét ví dụ thực hiện của thuật toán SPA-P-heuristic cho thể hiện SPA-P trong Bảng 4.1. Ban đầu, SPA-P-heuristic khởi tạo một phép ghép $M = \emptyset$ và thiết lập tất cả sinh viên ở trạng thái hoạt động. Tiếp theo, SPA-P-heuristic thực hiện các bước lặp chỉ ra trong Bảng 4.2. Cụ thể, từ bước lặp 1 đến bước lặp 4, SPA-P-heuristic ghép s_1, s_2, s_3 và s_4 với các đề tài có ưu tiên cao nhất trong danh sách xếp hạng của s_1, s_2, s_3 và s_4, tức là $M = \{(s_1, p_1), (s_2, p_5), (s_3, p_2), (s_4, p_4)\}$, gán $a(s_1) = 0$, $a(s_2) = 0$, $a(s_3) = 0$, và $a(s_4) = 0$. Ở bước lặp 5, SPA-P-heuristic ghép s_5 với p_5 vì p_5 có ưu tiên cao nhất trong danh sách xếp hạng của s_5, tức là $M = \{(s_1, p_1), (s_2, p_5), (s_3, p_2), (s_4, p_4), (s_5, p_5)\}$, và gán $a(s_5) = 0$. Vì l_2 đã vượt quá số sinh viên tối đa và $h_{l_2}(s_2)$ có giá trị lớn nhất, cặp ghép (s_2, p_5) bị xóa khỏi M, tức là $M = \{(s_1, p_1), (s_3, p_2), (s_4, p_4), (s_5, p_5)\}$, s_2 xóa p_5 trong danh sách xếp hạng và $a(s_2) = 1$. Ở bước lặp 6, SPA-P-heuristic ghép s_2 với p_1 vì p_1 có ưu tiên cao nhất trong danh sách xếp hạng của s_2, tức là $M = \{(s_1, p_1), (s_2, p_1), (s_3, p_2), (s_4, p_4), (s_5, p_5)\}$, và gán $a(s_2) = 0$. Vì p_1 vượt quá số sinh viên tối đa và $h_{l_1}(s_1)$ có giá trị lớn nhất, cặp ghép (s_1, p_1) bị xóa khỏi M, tức là $M = \{(s_2, p_1), (s_3, p_2), (s_4, p_4), (s_5, p_5)\}$, s_1 xóa p_1 trong danh sách xếp hạng và $a(s_1) = 1$. Ở bước lặp 7, SPA-P-heuristic ghép s_1 với p_3 vì p_3 có ưu tiên cao nhất trong danh sách xếp hạng của s_1, tức là $M = \{(s_1, p_3), (s_2, p_1), (s_3, p_2), (s_4, p_4), (s_5, p_5)\}$, và gán $a(s_1) = 0$. Ở bước lặp 8, vì tất cả $s_i \in \mathcal{S}$ đều ở trạng thái không hoạt động, SPA-P-heuristic trả về phép ghép hoàn chỉnh $M = \{(s_1, p_3), (s_2, p_1), (s_3, p_2), (s_4, p_4), (s_5, p_5)\}$.

4.2.5. Các kết quả thực nghiệm

4.2.5.1. So sánh với các thuật toán SPA-P-*approx và* SPA-P-*promotion*

Phần này trình bày các kết quả thực nghiệm để so sánh thời gian thực hiện và chất lượng nghiệm tìm được của thuật toán SPA-P-heuristic với các thuật toán SPA-P-approx [60] và SPA-P-promotion [102].

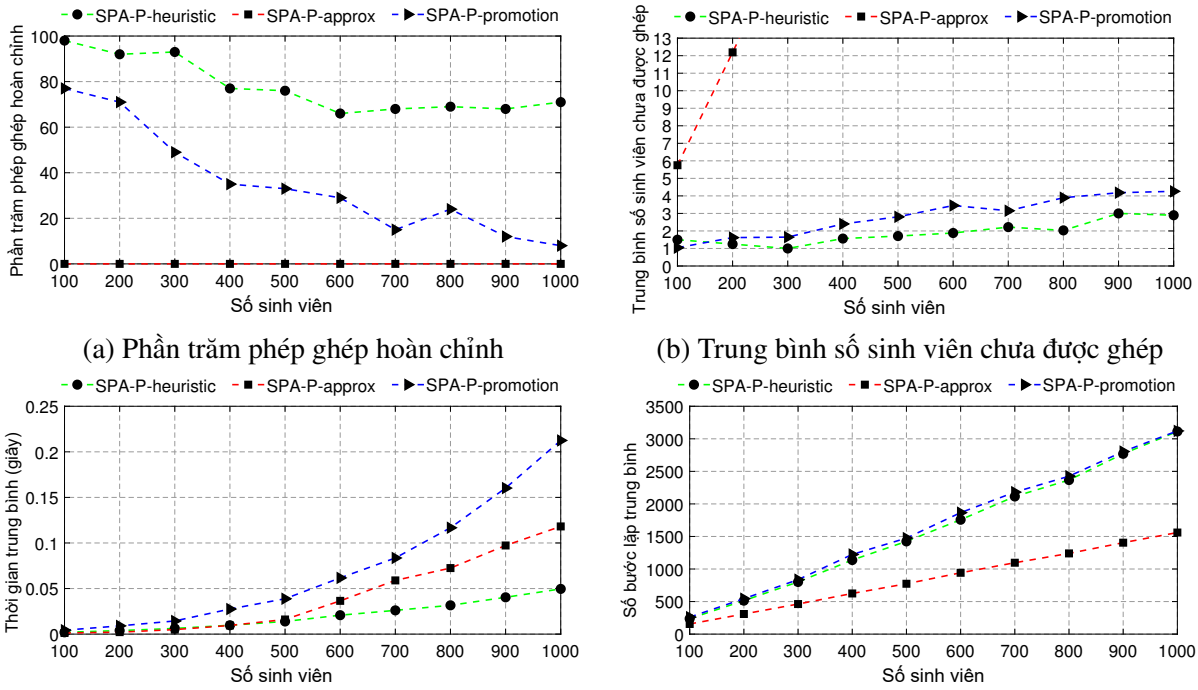
Bảng 4.3: *Các giá trị của tham số*

ID	n	Số giảng viên $(0.02n \leq m \leq 0.1n)$		Số đề tài $(0.1n \leq q \leq 0.5n)$	
		Min	*Max*	*Min*	*Max*
1	100	2	10	10	50
2	200	4	20	20	100
3	300	6	30	30	150
4	400	8	40	40	200
5	500	10	50	50	250
6	600	12	60	60	300
7	700	14	70	70	350
8	800	16	80	80	400
9	900	18	90	90	450
10	1000	20	100	100	500

Thực nghiệm 4.1. Thực nghiệm này chọn $n = \{100, 200, \cdots, 1000\}$. Với mỗi giá trị n, thực nghiệm tạo ngẫu nhiên 100 thể hiện SPA-P với các tham số (n, m, q), trong đó m và q là các số nguyên ngẫu nhiên thỏa mãn $0.02n \leq m \leq 0.1n$ và $0.1n \leq q \leq 0.5n$ như chỉ ra trong Bảng 4.3. Các ràng buộc của m và q có nghĩa là tỷ lệ sinh viên trên giảng viên là từ 10 đến 50, tỷ lệ sinh viên trên đề tài là từ 2 đến 10 và mỗi giảng viên đề xuất từ 1 đến 25 đề tài. Trong mỗi thể hiện SPA-P được tạo, mỗi sinh viên được xếp hạng ngẫu nhiên từ 1 đến 15 đề tài trong danh sách các đề tài được đề xuất bởi tất cả các giảng viên. Ngoài ra, số sinh viên tối đa c_j của các đề tài $p_j \in \mathcal{P}$ được sinh ngẫu nhiên thỏa mãn điều kiện $2 < c_j < 11$ và $\sum_{j=1}^{q} c_j = n$, tức là tổng số sinh viên tối đa của các đề tài bằng số sinh viên. Số sinh viên tối đa d_k của mỗi giảng viên $l_k \in \mathcal{L}$ được chọn là $d_k = \sum_{j=1}^{|P_k|} c_j$, trong đó c_j là số sinh viên tối đa của các đề tài $p_j \in P_k$.

Hình 4.1(a) chỉ ra phần trăm phép ghép hoàn chỉnh tìm được của SPA-P-heuristic, SPA-P-approx và SPA-P-promotion. Khi n tăng từ 100 đến 1000, SPA-P-heuristic tìm được từ 98% giảm xuống 66% phép ghép hoàn chỉnh, SPA-P-promotion tìm được từ 77% giảm xuống 8% phép ghép hoàn chỉnh, trong khi SPA-P-approx không tìm được phép ghép hoàn chỉnh nào. Hình 4.1(b) chỉ ra trung bình số sinh viên chưa được ghép của SPA-P-heuristic, SPA-P-approx và SPA-P-promotion. Khi n tăng từ 100 đến 1000, SPA-P-approx tìm thấy nhiều hơn 6 sinh viên chưa được ghép với các đề tài trong các phép ghép ổn định. Trong khi đó, SPA-P-heuristic tìm được số sinh viên chưa được ghép với các đề tài ít hơn so với SPA-P-promotion, tức là các phép ghép ổn định tìm được bởi SPA-P-heuristic tốt hơn các phép ghép ổn định tìm được bởi SPA-P-promotion về kích thước.

Hình 4.1(c) chỉ ra thời gian thực hiện trung bình của SPA-P-heuristic, SPA-P-approx và SPA-P-promotion. Khi n tăng từ 100 đến 1000, thời gian thực hiện trung bình của SPA-P-



(a) Phần trăm phép ghép hoàn chỉnh

(b) Trung bình số sinh viên chưa được ghép

(c) Thời gian thực hiện trung bình

(d) Trung bình số bước lặp

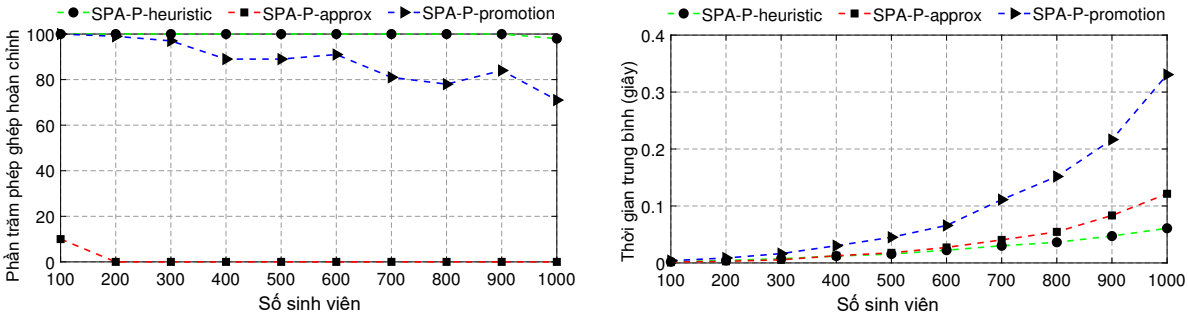
Hình 4.1: *So sánh về chất lượng nghiệm*

heuristic tăng từ 0.002 giây đến 0.049 giây, thời gian thực hiện trung bình của SPA-P-approx tăng từ 0.001 giây đến 0.118 giây, trong khi SPA-P-promotion tăng từ 0.004 giây đến 0.212 giây. Hình 4.1(d) chỉ ra số lần lặp lại trung bình của SPA-P-heuristic, SPA-P-approx và SPA-P-promotion. Số lần lặp lại trung bình của SPA-P-heuristic nhỏ hơn SPA-P-promotion, nhưng lớn hơn SPA-P-approx.

Thực nghiệm 4.2. Thực nghiệm này chọn các tham số (n, m, q) như trong Thực nghiệm 4.1, nhưng trong mỗi thể hiện SPA-P mỗi sinh viên được xếp hạng ngẫu nhiên từ 1 đến 25 đề tài trong danh sách các đề tài được đề xuất bởi tất cả các giảng viên.

Hình 4.2(a) chỉ ra phần trăm phép ghép hoàn chỉnh tìm của SPA-P-heuristic, SPA-P-approx và SPA-P-promotion. SPA-P-heuristic tìm được 100% phép ghép hoàn chỉnh khi n tăng từ 100 đến 900 và 98% phép ghép hoàn chỉnh khi $n = 1000$. SPA-P-promotion tìm được từ 100% giảm xuống 71% phép ghép hoàn chỉnh khi n tăng từ 100 đến 1000, trong khi SPA-P-approx chỉ tìm được 10% phép ghép hoàn chỉnh với $n = 100$. So với Thực nghiệm 4.1, trong đó mỗi sinh viên được phép xếp hạng ngẫu nhiên từ 1 đến 15 đề tài, chúng ta thấy rằng khi mỗi sinh viên xếp hạng nhiều đề tài hơn, các thuật toán này dễ tìm được các phép ghép hoàn chỉnh trong các thể hiện của SPA-P.

Hình 4.2(b) chỉ ra thời gian thực hiện trung bình của SPA-P-heuristic, SPA-P-approx và SPA-P-promotion. Kết quả thực nghiệm cho thấy rằng, thuật toán SPA-P-heuristic chạy nhanh hơn SPA-P-approx và SPA-P-promotion.



(a) Phần trăm phép ghép hoàn chỉnh

(b) Thời gian thực hiện trung bình

Hình 4.2: *Phần trăm phép ghép và thời gian thực hiện trung bình*

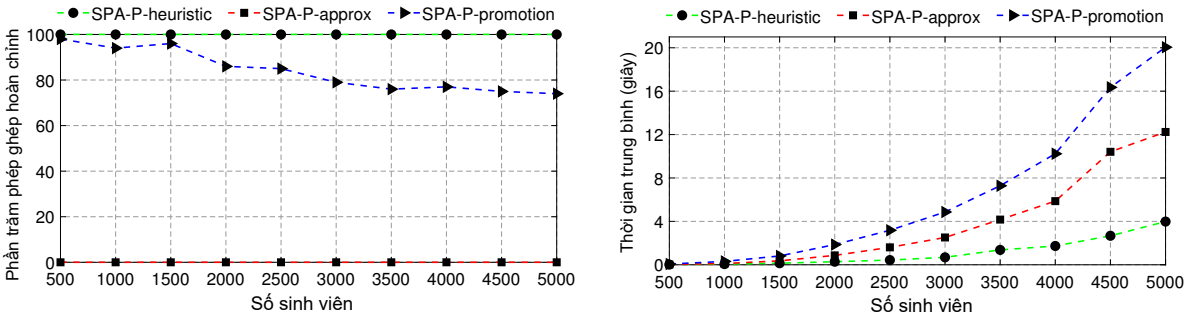
Thực nghiệm 4.3. Thực nghiệm này tăng số lượng sinh viên, số lượng giảng viên và số lượng đề tài để xem xét hiệu quả của SPA-P-heuristic cho các thể hiện SPA-P kích thước lớn. Cụ thể, thực nghiệm chọn các giá trị của tham số (n, m, q) như trong Bảng 4.4. Với mỗi giá trị của n tăng từ 500 đến 5000 với tăng 500, thực nghiệm tạo 100 thể hiện SPA-P của các tham số (n, m, q), trong đó m và q là các số ngẫu nhiên bị ràng buộc bởi $0.02n \leq m \leq 0.1n$ và $0.1n \leq q \leq 0.5n$. Điều này có nghĩa là tỷ lệ sinh viên trên giảng viên là từ 10 đến 50, tỷ lệ sinh viên trên đề tài là từ 2 đến 10 và mỗi giảng viên đề xuất từ 1 đến 25 đề tài. Trong mỗi thể hiện, mỗi sinh viên xếp hạng ngẫu nhiên từ 1 đến 30 đề tài trong danh sách các đề tài được đề xuất bởi tất cả các giảng viên. Số lượng sinh viên tối đa của giảng viên và đề tài được chọn như trong Thực nghiệm 4.1.

Hình 4.3(a) chỉ ra phần trăm phép ghép hoàn chỉnh tìm được của SPA-P-heuristic, SPA-P-approx và SPA-P-promotion. Khi n tăng từ 100 đến 1000, SPA-P-heuristic tìm được từ 98% giảm xuống 66% phép ghép hoàn chỉnh, SPA-P-promotion tìm được từ 77% giảm xuống 8% phép ghép hoàn chỉnh, trong khi SPA-P-promotion không tìm được bất kỳ phép ghép hoàn chỉnh nào. Hình 4.3(b) chỉ ra thời gian thực hiện trung bình của SPA-P-heuristic, SPA-P-approx và SPA-P-promotion. Khi n tăng từ 500 đến 5000, thời gian thực hiện trung bình của SPA-P-heuristic tăng từ 0.018 giây đến 3.980 giây, SPA-P-approx tăng từ 0.022 giây đến 12.240 giây và SPA-P-promotion tăng từ 0.069 giây đến 20.053 giây. Chúng ta có thể thấy rằng khi n tăng, SPA-P-heuristic vượt trội về thời gian thực hiện so với SPA-P-approx và SPA-P-promotion.

Từ kết quả của 3 thực nghiệm trên, chúng ta thấy rằng SPA-P-heuristic vượt trội SPA-P-approx và SPA-P-promotion về chất lượng nghiệm và thời gian thực hiện. Hơn nữa, khi $n = 5000$, $100 \leq m \leq 500$ và $500 \leq q \leq 2500$, SPA-P-heuristic chỉ cần khoảng 4.0 giây để tìm được phép ghép ổn định với kích thước tối đa, nghĩa là SPA-P-heuristic hiệu quả đối với bài toán SPA-P kích thước lớn.

Bảng 4.4: *Các giá trị tham số*

ID	n	Số giảng viên $(0.02n \leq m \leq 0.1n)$		Số đề tài $(0.1n \leq q \leq 0.5n)$	
		Min	*Max*	*Min*	*Max*
1	500	10	50	50	250
2	1000	20	100	100	500
3	1500	30	150	150	750
4	2000	40	200	200	1000
5	2500	50	250	250	1250
6	3000	60	300	300	1500
7	3500	70	350	350	1750
8	4000	80	400	400	2000
9	4500	90	450	450	2250
10	5000	100	500	500	2500



(a) Phần trăm phép ghép hoàn chỉnh

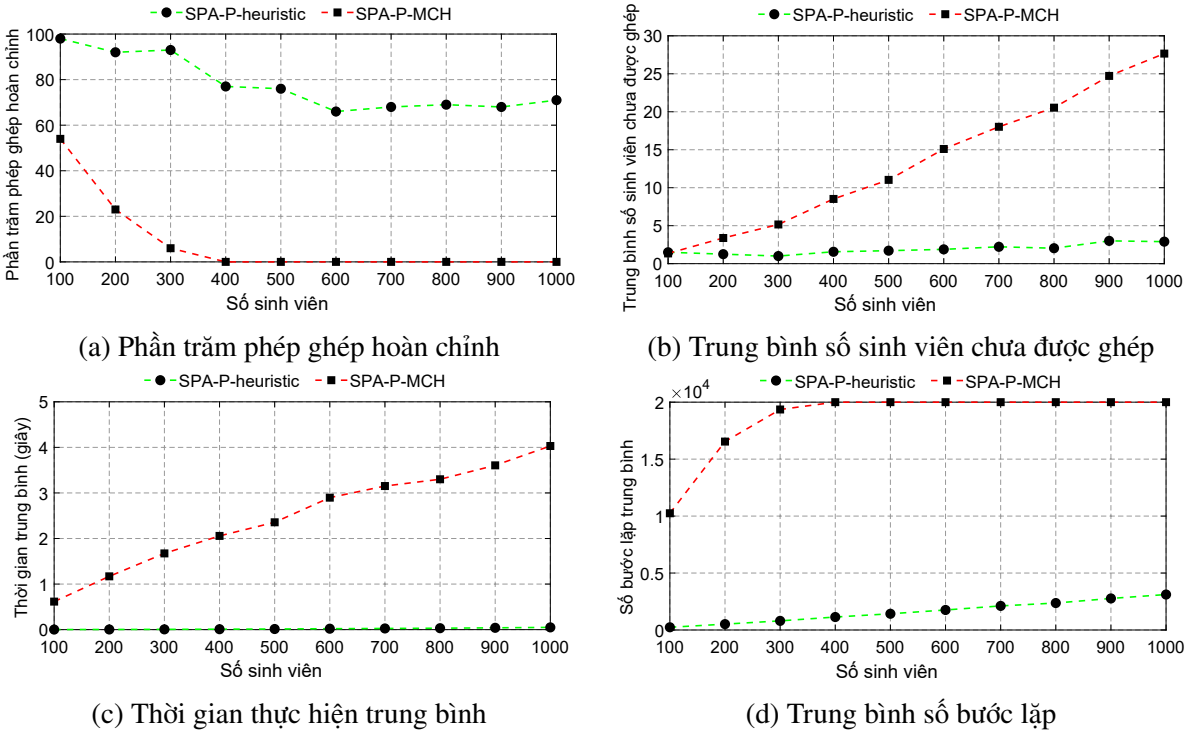
(b) Thời gian thực hiện trung bình

Hình 4.3: *Phần trăm phép ghép và thời gian thực hiện trung bình*

4.2.5.2. So sánh với thuật toán SPA-P-MCH

Phần này trình bày các kết quả thực nghiệm để so sánh thời gian thực hiện và chất lượng nghiệm tìm được của thuật toán SPA-P-heuristic với thời gian thực hiện và chất lượng nghiệm của thuật toán SPA-P-MCH [118] được công bố gần đây. Số bước lặp tối đa được thiết lập trong SPA-P-MCH là 20000.

Thực nghiệm 4.4. Thực nghiệm này sử dụng bộ dữ liệu được tạo trong Thực nghiệm 4.1. Hình 4.4(a) chỉ ra phần trăm phép ghép hoàn chỉnh tìm được của thuật toán SPA-P-heuristic và thuật toán SPA-P-MCH. Khi n tăng từ 100 đến 1000, SPA-P-heuristic tìm được từ 98% giảm xuống 66% phép ghép hoàn chỉnh, trong khi n tăng từ 100 đến 300 SPA-P-MCH tìm được từ 54% giảm xuống 6% phép ghép hoàn chỉnh và với $n > 300$, SPA-P-MCH không tìm được bất kỳ phép ghép hoàn chỉnh nào. Hình 4.4(b) chỉ ra trung bình số sinh viên chưa được ghép của SPA-P-heuristic và SPA-P-MCH. Khi n tăng từ 100 đến 1000, SPA-P-heuristic tìm được số sinh viên chưa được ghép với các đề tài ít hơn nhiều so với SPA-P-MCH.



(a) Phần trăm phép ghép hoàn chỉnh

(b) Trung bình số sinh viên chưa được ghép

(c) Thời gian thực hiện trung bình

(d) Trung bình số bước lặp

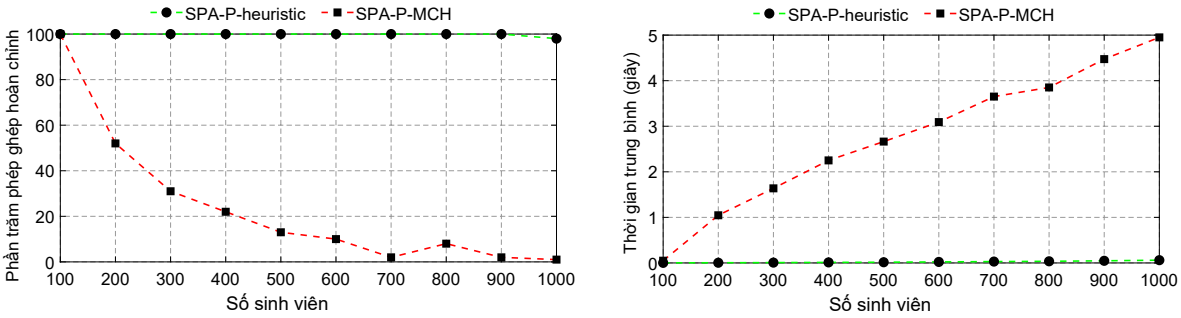
Hình 4.4: *So sánh về chất lượng nghiệm*

Điều này có nghĩa rằng SPA-P-heuristic vượt trội SPA-P-MCH về chất lượng nghiệm tìm được.

Hình 4.4(c) chỉ ra thời gian thực hiện trung bình của SPA-P-heuristic và SPA-P-MCH. Khi n tăng từ 100 đến 1000, thời gian thực hiện trung bình của SPA-P-heuristic tăng từ 0.002 giây đến 0.049 giây, trong khi thời gian thực hiện trung bình của SPA-P-MCH tăng từ 0.615 giây đến 4.030 giây, tức là SPA-P-heuristic chạy nhanh gấp khoảng từ 80 lần đến 300 lần SPA-P-MCH. Hình 4.4(d) chỉ ra số bước lặp trung bình của SPA-P-heuristic và SPA-P-MCH. Khi n tăng từ 100 đến 1000, số bước lặp trung bình của SPA-P-heuristic tăng từ 1000 đến 3113, trong khi số bước lặp trung bình của SPA-P-MCH tăng từ 10247 đến 20000, tức bằng số bước lặp tối đa được thiết lập trong SPA-P-MCH.

Thực nghiệm 4.5.Thực nghiệm này sử dụng bộ dữ liệu được tạo trong Thực nghiệm 4.2. Hình 4.5(a) chỉ ra phần trăm phép ghép hoàn chỉnh tìm được của SPA-P-heuristic và SPA-P-MCH. SPA-P-heuristic tìm được 100% phép ghép hoàn chỉnh khi n tăng từ 100 đến 900 và 98% phép ghép hoàn chỉnh khi $n = 100$, trong khi SPA-P-MCH tìm được từ 100% phép ghép hoàn chỉnh giảm xuống 1% phép ghép hoàn chỉnh khi n tăng từ 100 đến 1000. Hình 4.5(b) chỉ ra thời gian thực hiện trung bình của SPA-P-heuristic và SPA-P-MCH. Tương tự như Thực nghiệm 4.4, SPA-P-heuristic chạy nhanh gấp khoảng từ 80 lần đến 300 lần SPA-P-MCH.

Từ các kết quả thực nghiệm trên, chúng ta thấy rằng, SPA-P-heuristic vượt trội về thời gian thực hiện và chất lượng nghiệm so với SPA-P-MCH.



(a) Phần trăm phép ghép hoàn chỉnh (b) Thời gian thực hiện trung bình

Hình 4.5: *Phần trăm phép ghép hoàn chỉnh và thời gian thực hiện trung bình*

4.3. Đề xuất thuật toán HAG giải quyết bài toán MAX-SPA-ST

4.3.1. Ý tưởng

Phần này trình bày một thuật toán tìm kiếm heuristic (Heuristic Algorithm, viết tắt HAG) để giải quyết bài toán MAX-SPA-ST. Ý tưởng chính của HAG là bắt đầu từ một phép ghép rỗng, thuật toán HAG tìm một phép ghép ổn định với kích thước tối đa bằng cách định nghĩa hai *hàm heuristic* để cải thiện hiệu suất tìm kiếm và chất lượng nghiệm của bài toán so với các kết quả của thuật toán APX. Thực nghiệm chỉ ra rằng, thuật toán HAG vượt trội về thời gian và chất lượng phép ghép hoàn chỉnh so với thuật toán xấp xỉ APX [94].

Thuật toán sử dụng khái niệm đề tài tiềm năng (*potential*) là đề tài chưa đủ số lượng sinh viên tối đa và được đề xuất bởi một giảng viên cũng chưa đủ số lượng sinh viên tối đa. Cho một ví dụ của SPA-ST trong Bảng 4.5, một cặp (s_1, p_1) được gọi là chấp nhận bởi vì s_1 xếp hạng p_1 trong danh sách xếp hạng của s_1 và s_1 cũng được xếp hạng bởi l_1 người mà đề xuất đề tài p_1. Theo đó, chúng ta có $rank(s_1, p_1) = 1$ và $rank(l_1, s_1) = 2$. Giả sử cho một phép ghép M = $\{(s_1, p_7), (s_2, \varnothing), (s_3, p_4), (s_4, p_2), (s_5, p_1), (s_6, p_6), (s_7, p_8)\}$, trong đó s_2 là chưa được ghép vì $M(s_2) = \varnothing$, $|M(p_1)| = 1 < c_1 = 2$, và $|M(p_2)| = c_2 = 1$. Tương tự, ta có $|M(l_1)| = 2 < d_1 = 3$ và $|M(l_2)| = d_2 = 2$. Vì vậy, p_1 được gọi là một đề tài *tiềm năng* bởi vì cả p_1 và l_1 đều chưa đủ số lượng sinh viên tối đa.

4.3.2. Hàm heuristic

Luận án định nghĩa hai hàm heuristic để định hướng quá trình tìm kiếm như sau: *Thứ nhất* là hàm $h(p_j)$ giúp định hướng sinh viên chọn đề tài tốt nhất để ghép; *Thứ hai* là hàm $g(s_t)$ giúp giảng viên lựa chọn sinh viên để loại khi sinh viên đề xuất đề tài do giảng viên đó hướng dẫn, mà giảng viên hoặc đề tài đã đủ số lượng sinh viên tối đa.

Bảng 4.5: *Một thể hiện của* SPA-ST

Danh sách xếp hạng của $s_i \in \mathcal{S}$	Danh sách xếp hạng của $l_k \in \mathcal{L}$
s_1: $(p_1\ p_7)$	l_1: $(s_7\ s_4)\ s_1\ s_3\ (s_2\ s_5)\ s_6$
s_2: $p_1\ p_3\ p_5$	l_2: $s_3\ s_2\ s_7\ s_5$
s_3: $(p_2\ p_1)$	l_3: $(s_1\ s_7)\ s_6$
s_4: p_2	
s_5: $p_1\ p_4$	l_1 đề xuất p_1, p_2, p_3
s_6: $p_2\ p_8$	l_2 đề xuất p_4, p_5, p_6
s_7: $(p_5\ p_3)\ p_8$	l_3 đề xuất p_7, p_8
Số sinh viên tối đa của $p_j \in \mathcal{P}$: c_1 = 2, c_j = 1, $(2 \leq j \leq 8)$	
Số sinh viên tối đa của $l_k \in \mathcal{L}$: d_1 = 3, d_2 = 2, d_3 = 2	

Định nghĩa 4.1 (Hàm $h(p_j)$)**.** *Với mỗi sinh viên $s_i \in \mathcal{S}$, thuật toán xác định hàm heuristic $h(p_j)$ cho mỗi đề tài p_j trong danh sách xếp hạng s_i, trong đó p_j được đề xuất bởi l_k để chọn đề tài tốt nhất dựa vào giá trị hàm $h(p_j)$ nhỏ nhất:*

$$h(p_j) = rank(s_i, p_j) - min(d_k - |M(l_k)|, 1)/2 - (c_j - |M(p_j)|)/(2 \times c_j + 1). \quad (4.2)$$

Nếu (s_i, p_j) là cặp chấp nhận, thì $1 \leq rank(s_i, p_j) \leq q$. Nếu l_k đã đủ số lượng sinh viên tối đa, tức là $d_k - |M(l_k)| = 0$, thì $min(d_k - |M(l_k)|, 1)/2 = 0$. Nếu l_k chưa đủ số lượng sinh viên tối đa, tức là $|M(l_k)| < d_k$, thì $min(d_k - |M(l_k)|, 1)/2 = 0.5$ với mọi $l_k \in \mathcal{L}$. Ngoài ra, chúng ta có $0 \leq c_j - |M(p_j)| \leq c_j$, thì $0 \leq (c_j - |M(p_j)|)/(2 \times c_j + 1) < 0.5$ với mọi $p_j \in \mathcal{P}$, tức là $0 < h(p_j) \leq q$.

Mệnh đề 4.1. *Nếu một đề tài p_j được ghép cho một sinh viên s_i, thì không tồn tại bất kỳ một đề tài p_z sao cho $rank(s_i, p_z) < rank(s_i, p_j)$ từ danh sách xếp hạng của s_i mà có thể tạo thành cặp chặn với s_i (dòng 13).*

Chứng minh. Chúng ta có $min(d_k - |M(l_k)|, 1)/2 = 0$ hoặc 0.5 và $0 \leq (c_j - |M(p_j)|)/(2 \times c_j + 1) < 0.5$, vì vậy chúng ta có $rank(s_i, p_j) - 1 < h(p_j) \leq rank(s_i, p_j)$. Chú ý rằng, nếu có một đề tài p_t mà $h(p_t) > h(p_j)$, thì $rank(s_i, p_t) \geq rank(s_i, p_j)$ trong danh sách xếp hạng của s_i với mọi $s_i \in \mathcal{S}$. Do vậy, nếu một đề tài $p_j \in \mathcal{P}$ được chọn sao cho $h(p_j)$ là nhỏ nhất, nghĩa là thứ tự ưu tiên của p_j là cao nhất trong danh sách xếp hạng của s_i. □

Mệnh đề 4.2. *Nếu tồn tại một số đề tài có cùng thứ tự ưu tiên trong danh sách xếp hạng của s_i, thì thuật toán sẽ chọn một đề tài tiềm năng.*

Chứng minh. Nếu p_j hoặc l_k đủ số lượng sinh viên tối đa, tức là $c_j - |M(p_j)| = 0$ hoặc $d_k - |M(l_k)| = 0$, có nghĩa là $(c_j - |M(p_j)|)/(2 \times c_j + 1) = 0$ hoặc $min(d_k - |M(l_k)|, 1)/2 = 0$. Bởi vì $rank(s_i, p_j) - 0.5 \leq h(p_j) \leq rank(s_i, p_j)$. Nếu p_j là một đề tài tiềm năng, tức là

$c_j - |M(p_j)| > 0$ và $d_k - |M(l_k)| > 0$, thì tương ứng với $(c_j - |M(p_j)|)/(2 \times c_j + 1) > 0$ và $min(d_k - |M(l_k)|, 1)/2 = 0.5$. Vì vậy, chúng ta có $rank(s_i, p_j) - 1 < h(p_j) < rank(s_i, p_j) - 0.5$. Giả sử rằng có một đề tài p_k có cùng thứ tự ưu tiên với đề tài p_j trong danh sách xếp hạng của s_i, tức là $rank(s_i, p_k) = rank(s_i, p_j)$, trong đó p_k đủ số lượng sinh viên tối đa hoặc giảng viên người mà đề xuất p_k đủ số lượng sinh viên tối đa và p_j là một đề tài *tiềm năng*, thì chúng ta có $h(p_j) < rank(s_i, p_j) - 0.5 \leq h(p_k)$, vì thế thuật toán sẽ chọn p_j để ghép trong quá trình thực hiện. □

Mệnh đề 4.3. *Nếu tồn tại một số đề tài tiềm năng có thứ tự ưu tiên xếp hạng ưu tiên cao nhất trong danh sách xếp hạng của s_i, thì thuật toán sẽ chọn một đề tài có số sinh viên còn có thể ghép là lớn nhất.*

Chứng minh. Giả sử rằng, có hai đề tài *tiềm năng* là p_j và p_k có cùng thứ tự ưu tiên nhỏ nhất trong danh sách xếp hạng của s_i, điều đó có nghĩa là $rank(s_i, p_j) = rank(s_j, p_k) = y$, trong đó y là giá trị nhỏ nhất trong danh sách xếp hạng, p_j và p_k được đề xuất tương ứng với l_z và l_u, vì vậy $min(d_z - M(l_z), 1)/2 = min(d_u - M(l_u), 1)/2 = 0.5$, có nghĩa là $h(p_j) = y - 0.5 - (c_j - M(p_j)/(2 \times c_j + 1)$ và $h(p_k) = y - 0.5 - (c_k - M(p_k)/(2 \times c_k + 1)$. Nếu p_j có số sinh viên còn có thể ghép nhiều hơn p_k, thì $(c_j - M(p_j)/(2 \times c_j + 1) > (c_k - M(p_k)/(2 \times c_k + 1)$, vì vậy $h(p_j) < h(p_k)$, do đó thuật toán sẽ chọn p_j để ghép. □

Định nghĩa 4.2 (Hàm $g(s_t)$)**.** *Với mỗi sinh viên $s_i \in \mathcal{S}$, s_i đề xuất p_j nếu p_j hoặc l_k đã đủ số lượng sinh viên tối đa, thì thuật toán sẽ chọn sinh viên để xóa từ phép ghép hiện tại dựa vào giá trị lớn nhất của hàm heuristic $g(s_t)$:*

$$g(s_t) = rank(l_k, s_t) + t(s_t) + r(s_t)/(q + 1). \tag{4.3}$$

Với mỗi sinh viên $s_t \in M(l_k)$, thì ta có $1 \leq rank(l_k, s_t) \leq n$. Hơn thế nữa, $t(s_t) = sum(min(d_z - |M(l_z)|, 1) \times min(c_u - |M(p_u)|, 1) \times n)$, trong đó p_u có cùng thứ tự ưu tiên với $M(s_t)$ trong danh sách xếp hạng của s_t và p_u được đề xuất bởi l_z. Nếu p_u chưa đủ số lượng sinh viên tối đa, tức là $|M(p_u)| < c_u$, thì $1 \leq c_u - |M(p_u)| \leq c_u$, vì vậy ta có $min(c_u - |M(p_u)|, 1) = 1$. Ngược lại, nếu p_u đã đủ số lượng sinh viên tối đa, tức là $|M(p_u)| = c_u$, thì $c_u - |M(p_u)| = 0$, chúng ta sẽ có $min(c_u - |M(p_u)|, 1) = 0$. Tương tự, chúng ta có lần lượt là $min(d_z - |M(l_z)|, 1) = 1$ hoặc 0, nếu l_z là thiếu hoặc đủ số lượng sinh viên tối đa. Nếu p_u là một đề tài *tiềm năng*, chúng ta có $min(d_z - |M(l_z)|, 1) \times min(c_u - |M(p_u)|, 1) \times n = n$, ngược lại, $min(d_z - |M(l_z)|, 1) \times min(c_u - |M(p_u)|, 1) \times n = 0$. Vì vậy, chúng ta có $0 \leq t(s_t) \leq (q - 1) \times n$. Chúng ta gọi $r(s_t)$ là số đề tài được xếp hạng bởi sinh viên s_t, thì

$1 \leq r(s_t) \leq q$, vì vậy $0 < r(s_t)/(q+1) < 1$. Điều đó có nghĩa là $1 < g(s_t) < (q \times n + 1)$ với mọi $s_t \in M(l_k)$.

Mệnh đề 4.4. *Nếu một sinh viên s_t xếp hạng đề tài tiềm năng mà có cùng thứ tự ưu tiên với $M(s_t)$ trong danh sách xếp hạng của s_t, thì thuật toán sẽ chọn s_t để xóa.*

Chứng minh. Với mỗi $s_t \in M(l_k)$, nếu tồn tại một đề tài *tiềm năng* có cùng thứ tự ưu tiên với $M(s_t)$ trong danh sách xếp hạng của s_t, nghĩa là $t(s_t) > 0$, do đó chúng ta có $g(s_t) > n + 1$. Ngược lại, chúng ta có $t(s_t) = 0$, vì vậy $g(s_t) < n + 1$. Giả sử rằng s_t xếp hạng một đề tài *tiềm năng* có cùng thứ tự ưu tiên với $M(s_t)$ trong danh sách xếp hạng của s_t và s_k không xếp hạng đề tài *tiềm năng* nào có cùng thứ tự ưu tiên với $M(s_k)$ trong danh sách xếp hạng của s_k. Do đó, chúng ta có $g(s_k) < n + 1 < g(s_t)$, nghĩa là s_t sẽ được chọn để xóa khỏi phép ghép hiện tại. □

Mệnh đề 4.5. *Nếu có nhiều sinh viên xếp hạng đề tài tiềm năng mà có thứ tự ưu tiên bằng nhau với đề tài trong phép ghép hiện tại, thì thuật toán sẽ chọn một sinh viên có số lượng đề tài tiềm năng lớn nhất.*

Chứng minh. Giả sử rằng, có hai sinh viên s_t và s_k xếp hạng các đề tài *tiềm năng* có cùng thứ tự ưu tiên lần lượt với $M(s_t)$ và $M(s_k)$. Nếu s_t có số lượng đề tài *tiềm năng* lớn hơn s_k, thì chúng ta có $t(s_t) > t(s_k)$, nghĩa là $g(s_t) > g(s_k)$. Do đó, thuật toán sẽ chọn s_t. □

Mệnh đề 4.6. *Nếu một sinh viên s_t được chọn và $g(s_t) < n + 1$, thì không tồn tại sinh viên $s_w \in M(l_k)$ hoặc $M(p_j)$ sao cho $rank(l_k, s_w) > rank(l_k, s_t)$.*

Chứng minh. Giả sử rằng, nếu l_k hoặc p_j là đủ số lượng sinh viên tối đa, thì chúng ta có $t(s_t) = 0$, nghĩa là $rank(l_k, s_t) < g(s_t) < rank(l_k, s_t) + 1$. Nếu một sinh viên s_k thỏa mãn $g(s_k) < g(s_t)$ thì $rank(l_k, s_t) \geq rank(l_k, s_k)$ trong danh sách xếp hạng của l_k cho tất cả $l_k \in \mathcal{L}$. Hơn nữa, nếu một sinh viên $s_t \in \mathcal{S}$ được chọn sao cho $g(s_t)$ là lớn nhất, nghĩa là thứ tự ưu tiên của s_t là lớn nhất trong $M(l_k)$ hoặc $M(p_j)$. Trong trường hợp này, không tồn tại sinh viên $s_w \in M(l_k)$ hoặc $M(p_j)$ sao cho $rank(l_k, s_w) > rank(l_k, s_t)$. □

Mệnh đề 4.7. *Nếu tồn tại một số sinh viên có cùng thứ tự ưu tiên lớn nhất với $M(l_k)$ hoặc $M(p_j)$, thì sinh viên có số lượng đề tài lớn nhất sẽ được chọn.*

Chứng minh. Giả sử rằng, tồn tại một sinh viên s_t có cùng thứ tự ưu tiên với s_z trong $M(l_k)$ hoặc $M(p_j)$, thì thứ tự ưu tiên của s_t và s_z là lớn nhất trong danh sách xếp hạng của l_k, tức là $rank(l_k, s_t) = rank(l_k, s_z)$. Ngược lại, nếu không có sinh viên $s_w \in M(l_k)$ hoặc $M(p_j)$ sao

cho $rank(l_k, s_w) > rank(l_k, s_t)$ hoặc $rank(l_k, s_z)$. Nếu $r(s_t) > r(s_z)$, thì chúng ta chọn s_t để loại bỏ khỏi phép ghép hiện tại. Điều này có nghĩa là s_t có nhiều cơ hội nhất để ghép cho các đề tài khác hơn là s_z. □

4.3.3. Mô tả thuật toán

Thuật toán HAG được mô tả trong Thuật toán 4.2. Bắt đầu từ một phép ghép rỗng, $M = \emptyset$. Tại mỗi lần lặp, HAG xem xét một sinh viên $s_i \in \mathcal{S}$ chưa được ghép mà danh sách xếp hạng của s_i không rỗng. Sau đó, HAG tính giá trị hàm $h(p_j)$ cho mỗi đề tài $p_j \in \mathcal{P}$ trong danh sách xếp hạng của s_i (dòng 13-15). Tiếp theo, sinh viên s_i sẽ đề xuất đề tài p_j sao cho giá trị hàm $h(p_j)$ là nhỏ nhất, trong đó đề tài p_j được hướng dẫn bởi giảng viên giảng viên l_k. Khi đó, thuật toán xảy ra ba trường hợp như sau:

Trường hợp 1. Nếu cả p_j và l_k chưa đủ số lượng sinh viên tối đa, thì HAG sẽ thêm (s_i, p_j) vào M (dòng 18-19).

Trường hợp 2. Nếu p_j đủ số lượng sinh viên tối đa, HAG xác định giá trị hàm $g(s_t)$ cho mỗi sinh viên $s_t \in M(p_j)$ để chọn sinh viên s_t tương ứng với giá trị lớn nhất trong tổng số $g(s_t)$ được chỉ ra trong Thuật toán 4.3 (dòng 21). Nếu $g(s_t) > n + 1$ hoặc $rank(l_k, s_t) > rank(l_k, s_i)$ thì HAG xóa (s_t, p_j) và thêm (s_i, p_j) thành M (dòng 22-23), sau đó nó sẽ xóa p_j trong danh sách xếp hạng của s_t nếu $g(s_t) < n + 1$ (dòng 24-25). Nếu không, HAG sẽ xóa p_j trong danh sách xếp hạng của s_i (dòng 27). Lưu ý rằng nếu $g(s_t) > n + 1$, thì s_t chứa một đề tài *tiềm năng* có cùng thứ tự ưu tiên với p_j trong danh sách xếp hạng của s_t.

Trường hợp 3. Nếu l_k đủ số lượng sinh viên tối đa, HAG xác định giá trị hàm heuristic $g(s_w)$ cho mỗi sinh viên $s_w \in M(l_k)$ để chọn sinh viên s_w tương ứng với giá trị lớn nhất trong tổng số $g(s_w)$, như được chỉ ra trong Thuật toán 4.3 (dòng 29). Nếu $g(s_w) > n + 1$ hoặc $rank(l_k, s_w) > rank(l_k, s_i)$ thì HAG sẽ xóa (s_w, p_u), trong đó $p_u = M(s_w)$ và thêm (s_i, p_j) vào M (dòng 30-31), sau đó nó sẽ xóa p_u trong danh sách xếp hạng của s_w nếu $g(s_w) < n+1$ (dòng 33-34). Nếu không, HAG sẽ xóa p_j trong danh sách xếp hạng của s_i (dòng 36). Khi một đề tài p_u bị xóa, thì có thể tạo các cặp chặn với các sinh viên trong $M(l_k)$, sau đó HAG gọi Thuật toán 4.4 (dòng 32) để phá vỡ các cặp chặn loại *(3bi)* (Xem Định nghĩa 1.27 trong Chương 1).

Các trường hợp trên được lặp lại cho đến khi tìm được M là phép ghép ổn định. Nếu $|M| = n$, thì HAG trả về một kết quả phép ghép hoàn chỉnh. Ngược lại, HAG gọi Thuật toán 4.5 để ghép sinh viên *chưa ghép* trong phép ghép ổn định hiện tại. Thuật toán HAG dừng lại khi tìm được kết quả phép ghép *hoàn chỉnh* hoặc tất cả sinh viên *chưa được ghép*

Algorithm 4.2: Thuật toán HAG

Input: Thể hiện SPA-ST, I
Output: Phép ghép ổn định, M.

function HAG (I)
  $M := \emptyset$;
  $v(s_i) := 0, \forall s_i \in \mathcal{S}$;
  while *true* **do**
    s_i := sinh viên chưa ghép và danh sách xếp hạng của s_i chưa rỗng;
    if $\nexists\, s_i$ **then**
      if $|M| = n$ **then** break;
      else
        M' := Escape(M);
        if $M' = M$ **then** break;
        $M := M'$;
        continue;
    for *mỗi* $p_j \in A_i$ **do**
      l_k := giảng viên hướng dẫn đề tài p_j;
      $h(p_j) = rank(s_i, p_j)$-$min(d_k$-$|M(l_k)|, 1)$/2-$(c_j$-$|M(p_j)|)$ /$(2 \times c_j + 1)$;
    $p_j := \text{argmin}(h(p_j) > 0), \forall p_j \in \mathcal{P}$;
    l_k := giảng viên hướng dẫn đề tài p_j;
    if $|M(p_j)| < c_j$ *và* $|M(l_k)| < d_k$ **then**
      $M := M \cup \{(s_i, p_j)\}$;
    else if $|M(p_j)| = c_j$ **then**
      $[s_t, g(s_t)]$:= Choose_Student$(M(p_j), l_k)$;
      if $g(s_t) > n + 1$ *hoặc* $rank(l_k, s_i) < rank(l_k, s_t)$ **then**
        $M := M \setminus \{(s_t, p_j)\} \cup \{(s_i, p_j)\}$;
        if $g(s_t) < n + 1$ **then**
          $rank(s_t, p_j) := 0$;
      else
        $rank(s_i, p_j) := 0$;
    else
      $[s_w, g(s_w)]$:= Choose_Student$(M(l_k), l_k)$;
      if $g(s_w) > n + 1$ *hoặc* $rank(l_k, s_i) < rank(l_k, s_w)$ **then**
        $M := M \setminus \{(s_w, p_u)\} \cup \{(s_i, p_j)\}$, với $p_u = M(s_w)$;
        Repair(p_u, l_k);
        if $g(s_w) < n + 1$ **then**
          $rank(s_w, p_u) := 0$;
      else
        $rank(s_i, p_j) := 0$;
  return M;
end function

không thể tìm được bất kỳ đề tài nào để ghép.

Thuật toán 4.3 được sử dụng để chọn một sinh viên sao cho giá trị hàm $g(s_t)$ là lớn nhất. Gọi X là tập hợp các sinh viên sao cho $X = M(p_j)$, trong đó p_j là đủ số lượng sinh viên tối đa, hoặc $X = M(l_k)$ trong đó l_k đủ số lượng sinh viên tối đa. Với mỗi $s_t \in X$, thuật

Algorithm 4.3: Hàm heuristic $g(s_t)$

Input: Tập các sinh viên X.
Output: Sinh viên s_t và $g(s_t)$.

```
1.  function Choose_Student(X, l_k)
2.      for mỗi s_t ∈ X do
3.          t(s_t) := 0;
4.          for mỗi p_u | rank(s_t, p_u) = rank(s_t, M(s_t)) do
5.              l_z := giảng viên hướng dẫn đề tài p_u;
6.              t(s_t) = t(s_t) + min(d_z − |M(l_z)|, 1) × min(c_j − |M(p_u)|, 1) × n;
7.          r(s_t) := số lượng đề tài được xếp hạng bởi s_t;
8.          g(s_t) := rank(l_k, s_t) + t(s_t) + r(s_t)/(q + 1);
9.      s_t := argmax(g(s_t));
10.     return s_t, g(s_t);
11. end function
```

Algorithm 4.4: Phá vỡ các cặp chặn kiểu *(3bi)*

Input: Phép ghép M
Output: Phép ghép M.

```
1.  function Repair(p_u, l_k)
2.      f := true;
3.      while f = true do
4.          f:= false;
5.          if s_k ∈ M(l_k) và rank(s_k, p_u) < rank(s_k, p_z) | p_z = M(s_k) then
6.              M := M \ {(s_k, p_z)} ∪ {(s_k, p_u)};
7.              p_u := p_z;
8.              f := true;
9.      return M;
10. end function
```

toán sẽ tính toán giá trị hàm $g(s_t)$ (dòng 3-8) và trả về một sinh viên s_t sao cho giá trị hàm $g(s_t)$ là lớn nhất.

Thuật toán 4.4 được sử dụng để phá vỡ các cặp chặn dạng *(3bi)*. Khi một đề tài p_u bị xóa, với mỗi $s_k \in M(l_k)$, nếu $rank(s_k, p_u) < rank(s_k, p_z)$, trong đó $p_z = M(s_k)$, sau đó thuật toán loại bỏ (s_k, p_z) và thêm (s_k, p_u) vào M. Quá trình này lặp lại cho mỗi đề tài bị xóa cho đến khi nó không thể tạo thành các cặp chặn.

Trong trường hợp HAG đạt được một phép ghép *ổn định* nhưng *không hoàn chỉnh*, nghĩa là thuật toán mắc kẹt tối thiểu cục bộ. Ở mỗi lần lặp, đối với mỗi sinh viên $s_i \in \mathcal{S}$, HAG đề xuất một đề tài $p_j \in \mathcal{P}$, có xếp hạng ưu tiên cao nhất trong danh sách xếp hạng của s_i và thêm (s_i, p_j) vào M. Vì có xếp hạng ngang bằng trong danh sách xếp hạng của l_k (p_j được đề xuất bởi l_k), tồn tại một sinh viên s_u khác thích đề tài p_j hoặc p_z nhất (p_z được đề xuất bởi l_k) mà $rank(l_k, s_u) = rank(l_k, s_i)$. Khi s_u đề xuất p_j (p_j đủ số lượng sinh viên tối

Algorithm 4.5: Vượt qua tối thiểu cục bộ

Input: Phép ghép ổn định M.
Output: Phép ghép ổn định M.

```
1.  function Escape(M)
2.      for mỗi sinh viên chưa ghép s_u ∈ 𝒰 do
3.          Khởi tạo lại danh sách xếp hạng của s_u;
4.          while danh sách xếp hạng của s_u không rỗng do
5.              p_z := argmin(rank(s_u, p_z) > 0), ∀p_z ∈ 𝒫;
6.              l_k := giảng viên hướng dẫn đề tài p_z;
7.              for ( mỗi s_i ∈ M(l_k) | rank(l_k, s_i) = rank(l_k, s_u)) do
8.                  if (|M(p_z)|) < c_z) hoặc (s_i ∈ M(p_z) và |M(p_z)| = c_z) then
9.                      if v(s_u) ≥ v(s_i) then
10.                         p_j := M(s_i);
11.                         M := M \ {(s_i, p_j)} ∪ {(s_u, p_z)};
12.                         v(s_u) := v(s_u) + 1;
13.                         Repair(p_j, l_k);
14.                         break;
15.             if M(s_u) ≠ ∅ then
16.                 break;
17.             else
18.                 rank(s_u, p_z) := 0;
19.     return M;
20. end function
```

đa) hoặc p_z (l_k đủ số lượng sinh viên tối đa), nếu chúng ta giữ cặp (s_i, p_j) và từ chối cặp (s_u, p_z) hoặc (s_u, p_j), thì HAG có thể dẫn đến một phép ghép không hoàn chỉnh. Nếu chúng ta thêm (s_u, p_z) hoặc (s_u, p_j) và loại bỏ (s_i, p_j), HAG có thể tạo ra phép ghép hoàn chỉnh. Điều này có nghĩa là, chúng ta có thể thêm (s_u, p_z) hoặc (s_u, p_j) và loại bỏ (s_i, p_j), sau đó s_i đề xuất các đề tài khác. Hơn nữa, chúng ta có thể cải thiện kích thước của các phép ghép ổn định bằng cách sử dụng Thuật toán 4.5 để vượt qua tối thiểu cục bộ. Đối với mỗi sinh viên *chưa ghép* $s_u \in \mathcal{S}$, thuật toán tìm một đề tài p_z sao cho tồn tại một sinh viên s_i trong đó p_j được ghép cho s_i và $v(s_u) \geq v(s_i)$ (dòng 5-9). Sau đó, thuật toán thay thế (s_i, p_j) bằng (s_u, p_z) trong M và tăng giá trị của $v(s_u)$. Khi p_j bị loại bỏ, nó có thể tạo thành các cặp chặn với các sinh viên khác trong $M(l_k)$, do đó chúng ta gọi hàm được chỉ ra trong Thuật toán 4.4 để phá vỡ các cặp chặn. Cần lưu ý rằng điều kiện $v(s_u) \geq v(s_i)$ có nghĩa là số lượng thay thế của s_u nhiều hơn của s_i, tức là s_u được ưu tiên ghép cho $p_z \in l_k$. Theo đó, phép ghép M là *không ổn định* và s_i là sinh viên *chưa ghép*, sau đó s_i đề xuất các đề tài khác trong danh sách xếp hạng của s_i.

Mệnh đề 4.8. HAG *cần* $O(T)$ *thời gian để chọn một đề tài dựa trên giá trị nhỏ nhất của hàm*

$h(p_j)$, *trong đó* T *là độ dài của danh sách xếp hạng cho mỗi sinh viên.*

Chứng minh. Tại mỗi lần lặp, đối với mỗi sinh viên s_i, thuật toán tính giá trị hàm heuristic $h(p_j)$ cho mỗi đề tài trong danh sách xếp hạng của s_i. Gọi T là độ dài danh sách xếp hạng của s_i, nghĩa là số lượng đề tài được xếp hạng bởi s_i. Lưu ý rằng $1 \leq T \leq q$ trong đó q là tổng số đề tài của một thể hiện SPA-ST. Do đó, HAG cần $O(T)$ thời gian để tính toán giá trị hàm heuristic cho mỗi sinh viên. □

Mệnh đề 4.9. *Thuật toán 4.3 cần* $O(E)$ *thời gian thực hiện để chọn một sinh viên dựa trên giá trị lớn nhất của hàm heuristic, trong đó* E *là tổng độ dài của danh sách xếp hạng sinh viên trong tập* X*, trong đó* X *là tập hợp sinh viên, sao cho* $X = M(p_j)$ *nếu* p_j *là đủ số lượng sinh viên tối đa hoặc* $X = M(l_k)$ *nếu* l_k *là đủ số lượng sinh viên tối đa.*

Chứng minh. Thời gian thực hiện của Thuật toán 4.3 được mô tả như sau: Đối với mỗi sinh viên $s_t \in X$, thuật toán xem xét tất cả các đề tài trong danh sách xếp hạng của s_t và tính số đề tài *tiềm năng* có cùng thứ tự ưu tiên với $M(s_t)$ trong danh sách xếp hạng của s_t, ký hiệu là $t(s_t)$. Do đó, quá trình xử lý này cần $O(T)$ thời gian trong đó T là độ dài của danh sách xếp hạng của s_t. Thuật toán 4.3 xem xét tất cả sinh viên trong tập X, do đó cần $O(E)$ thời gian, trong đó E là tổng độ dài của danh sách xếp hạng của tất cả sinh viên trong X, ký hiệu là $E = \sum_{i=1}^{|X|} T_i$, để chọn một sinh viên dựa trên giá trị lớn nhất của hàm heuristic. □

Mệnh đề 4.10. HAG *cần* $O(Z)$ *thời gian để tìm lời giải của một thể hiện* SPA-*ST, trong đó* Z *là tổng độ dài của tất cả danh sách xếp hạng của sinh viên.*

Chứng minh. Mỗi sinh viên đăng ký một đề tài trong danh sách xếp hạng của mình với nhiều nhất ba lần trong vòng lặp chính của thuật toán HAG.

(i) Giả sử rằng $s_i \in \mathcal{S}$ đề xuất đề tài $p_j \in \mathcal{P}$ trong danh sách xếp hạng của s_i. Sau đó, HAG thêm (s_i, p_j) vào phép ghép M.

(ii) Sau khi s_i bị xóa nhưng p_j không bị xóa khỏi danh sách xếp hạng của s_i, theo đó s_i lại đề xuất p_j.

(iii) Nếu s_i là sinh viên *chưa ghép* trong phép ghép ổn định, thì HAG sẽ gọi Thuật toán 4.5 để tìm một đề tài có thể được ghép cho s_i. Do đó, s_i có thể đề xuất p_j một lần nữa.

Vì tất cả sinh viên có thể đăng ký nhiều nhất ba lần các đề tài trong quá trình tìm kiếm của thuật toán HAG, do đó HAG cần $O(3Z) = O(Z)$ thời gian, trong đó Z là tổng độ dài của tất cả danh sách xếp hạng của sinh viên. □

4.3.4. Ví dụ

Phần này trình bày ví dụ mô tả quá trình thực hiện của thuật toán HAG để tìm một phép ghép ổn định với kích thước tối đa cho một thể hiện của SPA-ST trong Bảng 4.5. Bắt đầu từ một phép ghép, $M = \varnothing$, HAG thực hiện các lần lặp được mô tả trong Bảng 4.6. Tại lần lặp thứ nhất, vì s_1 xếp hạng đề tài p_1 và p_7, HAG tính giá trị của hàm $h(p_1)$ và $h(p_7)$ trong danh sách xếp hạng của s_1, chúng ta có $h(p_1)$ = 0.1, $h(p_7)$ = 0.2. Tiếp theo, HAG chọn p_1 vì hàm heuristic có giá trị nhỏ nhất và thêm (s_1, p_1) vào $M = \{(s_1, p_1)\}$. Tại lần lặp thứ hai, HAG tính các giá trị của hàm $h(p_1)$, $h(p_3)$ và $h(p_5)$ trong danh sách xếp hạng của s_2. Sau đó, HAG chọn p_1 vì $h(p_1)$ có giá trị nhỏ nhất và thêm (s_2, p_1) vào $M = \{(s_1, p_1), (s_2, p_1)\}$. Tại lần lặp thứ ba, HAG tính các giá trị của hàm $h(p_1)$, $h(p_2)$ và $h(p_4)$ trong danh sách xếp hạng của s_3. Sau đó, HAG chọn p_2 vì hàm $h(p_2)$ có giá trị nhỏ nhất và thêm (s_3, p_2) vào $M = \{(s_1, p_1), (s_2, p_1), (s_3, p_2)\}$. Tại lần lặp thứ tư, HAG tính giá trị của hàm $h(p_2)$ trong danh sách xếp hạng của s_4. Vì p_2 đã đủ số lượng sinh viên tối đa, HAG xác định giá trị hàm heuristic của $g(s_3) = 3.3$. Thứ hạng của s_4 nhỏ hơn thứ hạng của s_3 trong danh sách xếp hạng của l_1, tức là $rank(l_1, s_4) < rank(l_1, s_3)$, do đó thuật toán loại cặp (s_3, p_2) từ M và xóa p_2 trong danh sách xếp hạng của s_3, sau đó HAG thêm (s_4, p_2) vào $M = \{(s_1, p_1), (s_2, p_1), (s_4, p_2)\}$. HAG tiếp tục thực hiện từng bước cho đến lần lặp thứ mười hai, thuật toán đạt được một phép ghép *ổn định*, M = $\{(s_4, p_2), (s_5, p_1), (s_7, p_5), (s_1, p_7), (s_5, p_6), (s_6, p_8)\}$ với kích thước 6, tuy nhiên M là phép ghép *không hoàn chỉnh*. Vì vậy, HAG gọi Thuật toán 4.5 để cải thiện kích thước phép ghép hiện tại. Sau lần lặp thứ mười bốn, HAG trả về kết quả một phép ghép hoàn chỉnh M = $\{(s_1, p_7), (s_2, p_1), (s_3, p_1), (s_4, p_2), (s_5, p_4), (s_6, p_8), (s_7, p_5)\}$ với $|M| = 7$.

4.3.5. Các kết quả thực nghiệm

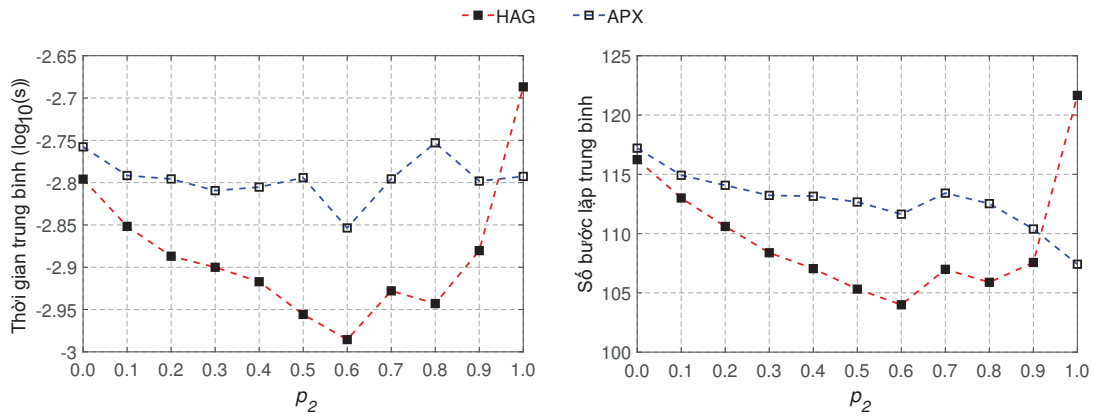
Phần này trình bày một số thực nghiệm để đánh giá hiệu quả về thời gian thực hiện trung bình và chất lượng nghiệm mà thuật toán HAG tìm được so với thuật toán xấp xỉ APX [94].

Thực nghiệm 4.7. Trong thực nghiệm này, luận án đã tạo các ngẫu nhiên các thể hiện SPA-ST với các tham số $(n, m, q, p_1, p_2, C, D)$, trong đó n = $\{100, 200\}$, $m = 0.05n$, $q = 0.1n$, $p_1 \in [0.1, 0.8]$ và $p_2 \in [0.0, 1.0]$ với bước tăng 0.1. Tổng số sinh viên tối đa lần lượt của các đề tài và giảng viên là $C = 1.2n$ và $D = 1.1n$. Trong đó, c_j được phân phối cho mỗi đề tài $p_j \in \mathcal{P}$ sao cho $0.6C/q \leq c_j \leq 1.4C/q$ và d_k được thiết lập của mỗi giảng viên $l_k \in \mathcal{L}$ sao cho $d_k = D/m$.

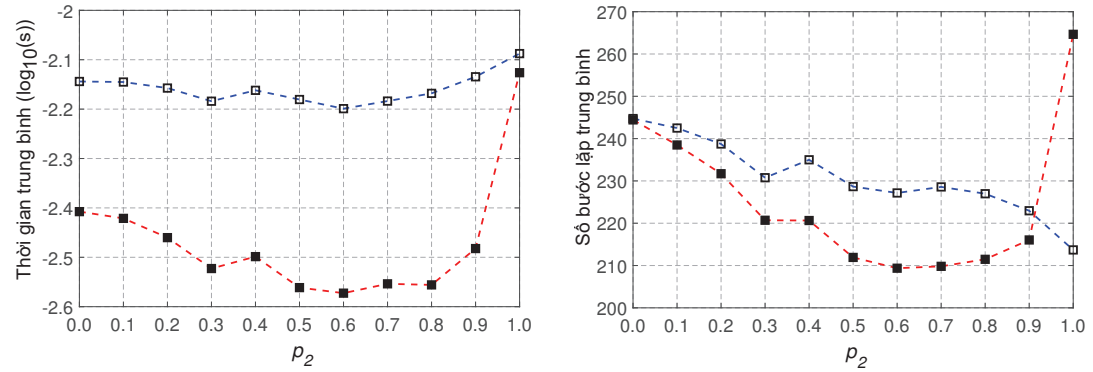
Bảng 4.6: *Ví dụ thực hiện của thuật toán* HAG *cho thể hiện* SPA-ST *trong Bảng 4.5*

Lặp(i)	s_i	p_j	s_t	M
1	s_1	p_1	-	Thêm (s_1, p_1) M =$\{(s_1, p_1)\}$
2	s_2	p_1	-	Thêm (s_2, p_1) M = $\{(s_1, p_1), (s_2, p_1)\}$
3	s_3	p_2	-	Thêm (s_3, p_2) M = $\{(s_1, p_1), (s_2, p_1), (s_3, p_2)\}$
4	s_4	p_2	s_3	Vì $\|M(p_2)\| = c_2$, $rank(l_1, s_3) > rank(l_1, s_4)$ Xóa (s_3, p_2) và thêm (s_4, p_2). Xóa p_2 trong danh sách xếp hạng của s_3 M = $\{(s_1, p_1), (s_2, p_1), (s_4, p_2)\}$.
5	s_5	p_1	s_1	Vì $\|M(p_1)\| = c_1$, $g(s_1) > n$ Xóa (s_1, p_1) và thêm (s_5, p_1) M = $\{(s_2, p_1), (s_4, p_2), (s_5, p_1)\}$
6	s_6	p_2	s_4	Vì $\|M(p_2)\| = c_2$, $rank(l_1, s_6) > rank(l_1, s_4)$. Xóa p_2 trong danh sách xếp hạng của s_6 M = $\{(s_2, p_1), (s_4, p_2), (s_5, p_1)\}$
7	s_7	p_5	-	Thêm (s_7, p_5) M = $\{(s_2, p_1), (s_4, p_2), (s_5, p_1), (s_7, p_5)\}$
8	s_1	p_7	-	Thêm (s_1, p_7) M = $\{(s_2, p_1), (s_4, p_2), (s_5, p_1), (s_7, p_5), (s_1, p_7)\}$.
9	s_3	p_1	s_2	Vì $\|M(p_1)\| = c_1$, $rank(l_1, s_2) > rank(l_1, s_3)$ Xóa (s_2, p_1) và thêm (s_3, p_1). Xóa p_1 trong danh sách xếp hạng của s_2 M = $\{(s_4, p_2), (s_5, p_1), (s_7, p_5), (s_1, p_7), (s_3, p_1)\}$
10	s_6	p_8	-	Thêm (s_6, p_8) M = $\{(s_4, p_2), (s_5, p_1), (s_7, p_5), (s_1, p_7), (s_3, p_1), (s_6, p_8)\}$.
11	s_2	p_3	s_5	Vì $\|M(l_1)\| = d_1$, $rank(l_1, s_2) > rank(l_1, s_5)$. Xóa p_3 trong danh sách xếp hạng s_2 M = $\{(s_4, p_2), (s_5, p_1), (s_7, p_5), (s_1, p_7), (s_5, p_6), (s_6, p_8)\}$
12	s_2	p_5	s_7	Vì $\|M(p_5)\| = c_5$, $rank(l_2, s_2) > rank(l_2, s_7)$. Xóa p_5 trong danh sách xếp hạng của s_2 M = $\{(s_4, p_2), (s_5, p_1), (s_7, p_5), (s_1, p_7), (s_5, p_6), (s_6, p_8)\}$
13	Vì M là chưa hoàn chỉnh, HAG gọi tới Thuật toán 4.5 để cải tiến phép ghép hiện tại, M = $\{(s_4, p_2), (s_7, p_5), (s_1, p_7), (s_3, p_1), (s_6, p_8), (s_2, p_1)\}$			
14	s_5	p_4	-	Thêm (s_5, p_4) M = $\{(s_4, p_2), (s_7, p_5), (s_1, p_7), (s_3, p_1), (s_6, p_8), (s_2, p_1), (s_5, p_4)\}$
Sau 14 *lần lặp, thuật toán* HAG *trả về một phép ghép hoàn chỉnh:* M = $\{(s_1, p_7), (s_2, p_1), (s_3, p_1), (s_4, p_2), (s_5, p_4), (s_6, p_8), (s_7, p_5)\}$ *với* $\|M\| = 7$.				

Đầu tiên, luận án so sánh thời gian thực hiện trung bình và số bước lặp trung bình của HAG với thời gian thực hiện của APX để tìm các kết quả phép ghép hoàn chỉnh cho $n = 100$ và 200 và tính trung bình các kết quả dựa trên các giá trị của p_1 được chỉ ra trong Hình 4.6. Hình 4.6(a) và 4.6(c) chỉ ra rằng HAG chạy nhanh hơn APX cho $n = 100$ và 200 với mọi giá trị p_2. Khi p_2 tăng từ 0.0 tới 0.9 với $n = 100$, thời gian thực hiện của thuật toán HAG trong khoảng từ $10^{-2.98}$ giây đến $10^{-2.68}$ giây, trong khi thuật toán APX cần khoảng từ $10^{-2.85}$ giây



(a) —— $n = 100$, $m = 5$, $q = 10$, $p_1 \in [0.1, 0.8]$ —— (b)



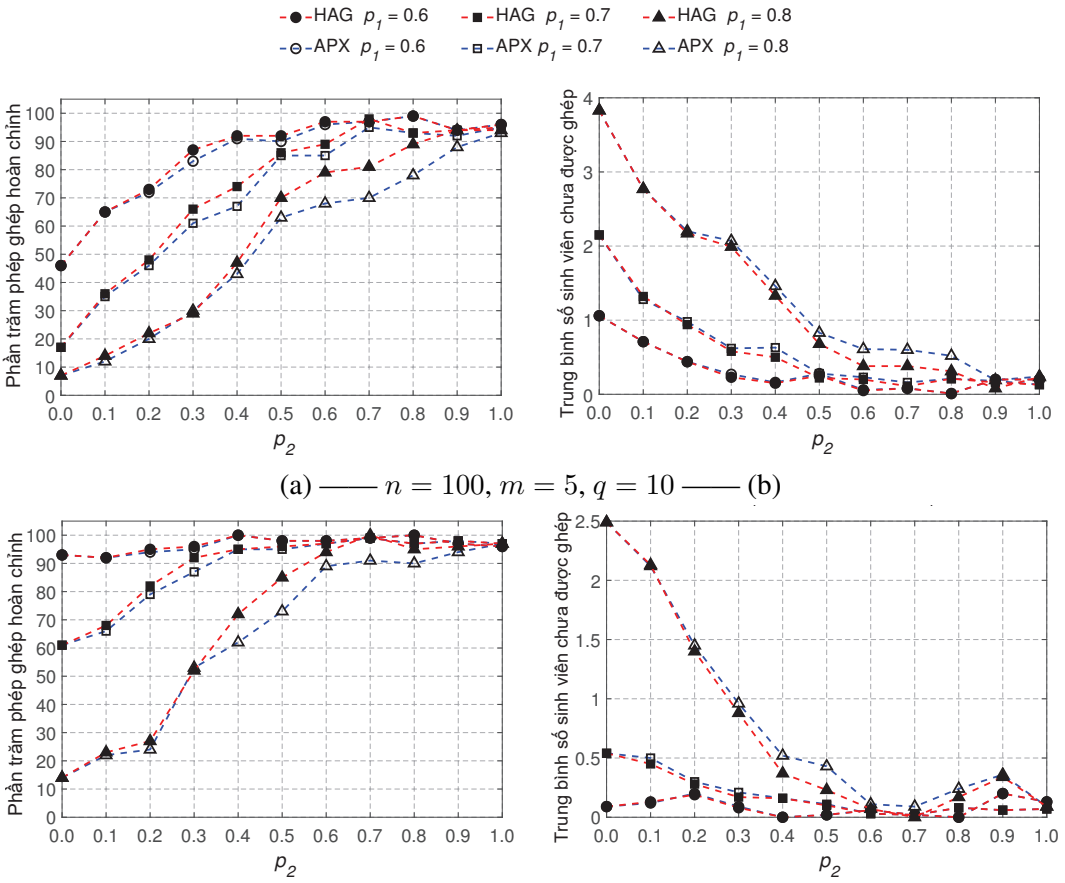
(c) —— $n = 200$, $m = 10$, $q = 20$, $p_1 \in [0.1, 0.8]$ —— (d)

Hình 4.6: *Thời gian thực hiện và số bước lặp của* HAG *và* APX

tới $10^{-2.75}$ giây, nhưng khi $p_2 = 1.0$, tức là các thứ tự ưu tiên trong danh sách xếp hạng sinh viên và giảng viên đều bằng 1, thời gian thực hiện trung bình của HAG tăng, điều này là do HAG cần nhiều thời gian hơn để tìm các đề tài cho sinh viên *chưa ghép* trong Thuật toán 4.5. Khi $n = 200$, thời gian thực hiện trung bình của HAG tăng từ $10^{-2.58}$ giây tới $10^{-2.28}$ giây, trong khi APX cần từ $10^{-2.23}$ giây tới $10^{-2.16}$ giây.

Hình 4.6(b) và 4.6(d) chỉ ra trung bình số bước lặp cần thiết của HAG và APX với $n = 100$ và 200 với p_2. Thuật toán HAG cần ít số lần lặp hơn thuật toán APX với p_2 với mọi giá trị p_2 từ 0.0 tới 0.9 cho trường hợp $n = 100$ và 200. Điều này giải thích rằng HAG chạy nhanh hơn APX trong Hình 4.6(a) và 4.6(c). Tuy nhiên, khi $p_2 = 1.0$, HAG cần số bước lặp nhiều hơn APX với mọi p_1. Do đó, thời gian thực hiện trung bình của HAG tăng. Từ kết quả thực nghiệm chỉ ra rằng HAG hiệu quả hơn APX về thời gian thực hiện và số lần lặp lại với $n = 100$ và 200 với mỗi $p_1 \in [0.1, 0.8]$ và $p_2 \in [0.0, 1.0]$.

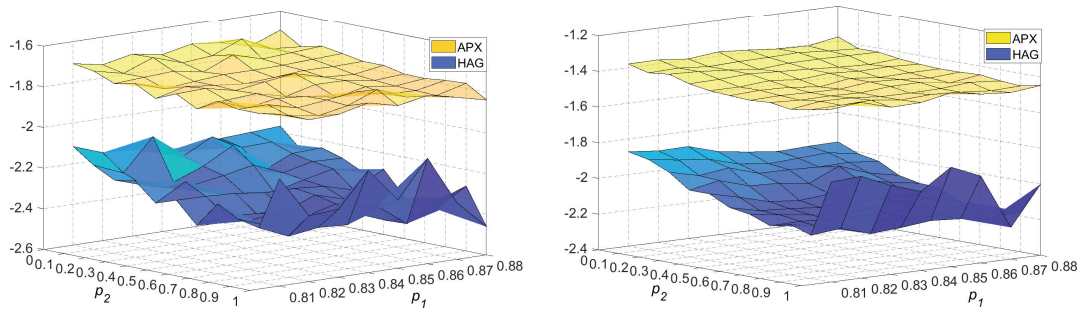
Tiếp theo, thuật toán so sánh phần trăm phép ghép hoàn chỉnh và số lượng trung bình của sinh viên chưa ghép do HAG tìm được phần trăm của APX với mọi giá trị p_1 và p_2. Kết quả thực nghiệm chỉ ra rằng khi p_1 thay đổi từ 0.1 đến 0.5 với bước 0.1, cả HAG và APX đều tìm được phép ghép hoàn chỉnh, do đó chúng ta chỉ chỉ ra kết quả cho giá trị p_1 từ 0.6 đến



Hình 4.7: *Phần trăm phép ghép hoàn chỉnh và số sinh viên chưa được ghép của* HAG *và* APX

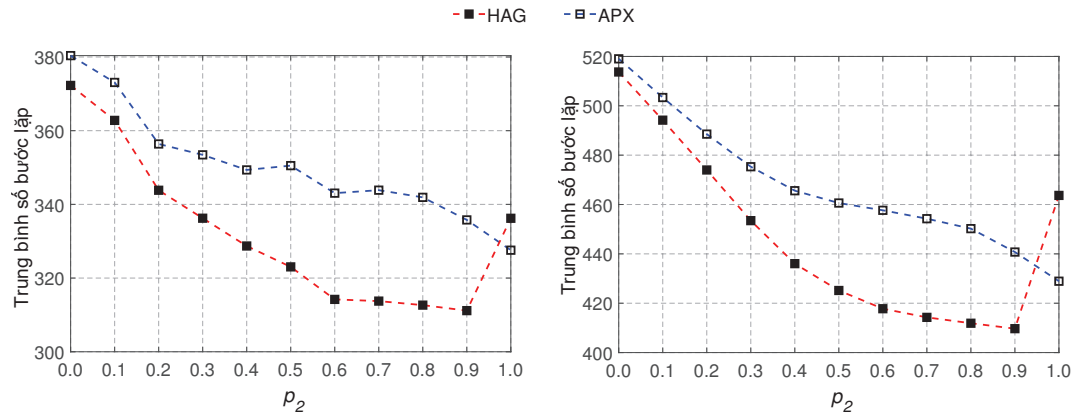
0.8 với mọi giá trị p_2 trong Hình 4.7.

Hình 4.7 chỉ ra rằng khi p_1 tăng, nghĩa là việc tìm kiếm các phép ghép hoàn chỉnh là rất khó, nhưng HAG tìm được phần trăm phép ghép hoàn chỉnh tốt hơn so với APX tìm được. Khi p_2 thay đổi từ 0.1 đến 1.0, HAG tìm được phần trăm phép ghép hoàn chỉnh cao hơn so với APX trong Hình 4.7(a) và 4.7(c). Điều này là do HAG đã sử dụng hai hàm heuristic bao gồm (*i*) tìm một đề tài $p_j \in \mathcal{P}$ có giá trị heuristic nhỏ nhất $h(p_j)$ để ghép mỗi sinh viên; và (*ii*) xác định một sinh viên s_t có giá trị hàm heuristic lớn nhất $g(s_t)$ để loại khi đề tài hoặc giảng viên đã nhận quá số lượng sinh viên tối đa trong phép ghép hiện tại. Nếu tìm được một phép ghép ổn định nhưng vẫn chưa hoàn chỉnh, HAG sẽ cải thiện kích thước của phép ghép ổn định bằng cách ưu tiên các sinh viên *chưa ghép* để ghép với các đề tài. Bằng cách làm này, HAG tìm được phần trăm phép ghép hoàn chỉnh cao hơn và chạy nhanh hơn APX. Tuy nhiên, khi $p_2 = 0$, cả hai thuật toán đều tìm được phần trăm phép ghép hoàn chỉnh như nhau vì tất cả các phép ghép ổn định có cùng kích thước. Hình 4.7(b) và 4.7(d) chỉ ra số sinh viên chưa được ghép trung bình tìm được bởi HAG và APX. Rõ ràng rằng, khi p_2 thay đổi từ 0.1 đến 1.0, HAG tìm được số lượng sinh viên chưa được ghép ít hơn so với APX. Điều này có



(a) $n = 300$, $m = 15$, $q = 30$ (b) $n = 400$, $m = 20$, $q = 40$

Hình 4.8: *Thời gian thực hiện trung bình của* HAG *và* APX



(a) $n = 300$, $m = 15$, $q = 30$ (b) $n = 400$, $m = 20$, $q = 40$

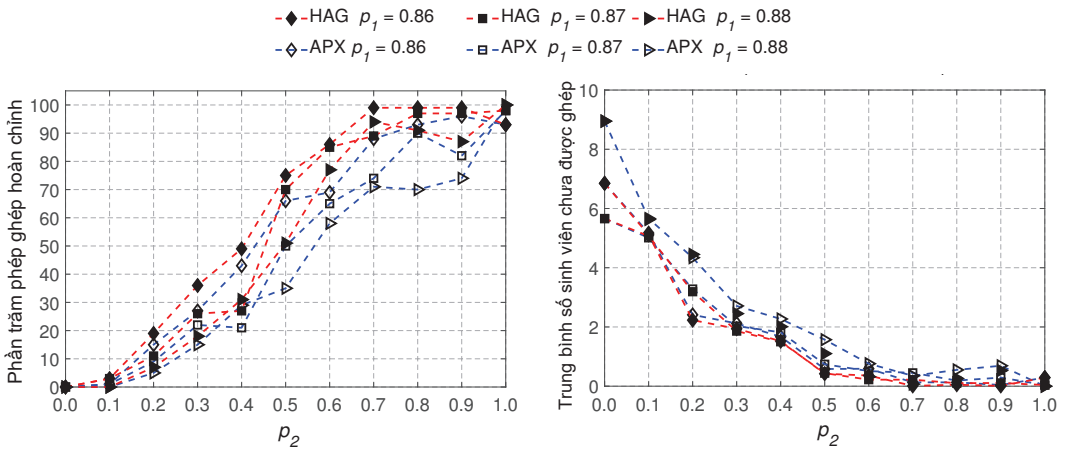
Hình 4.9: *Trung bình số lần lặp* HAG *và* APX *với* $p_1 \in [0.81, 0.88]$

nghĩa là HAG tìm được các phép ghép ổn định tối đa tốt hơn so với các phép ghép tìm được bởi APX khi p_1 tăng trong thực nghiệm này.

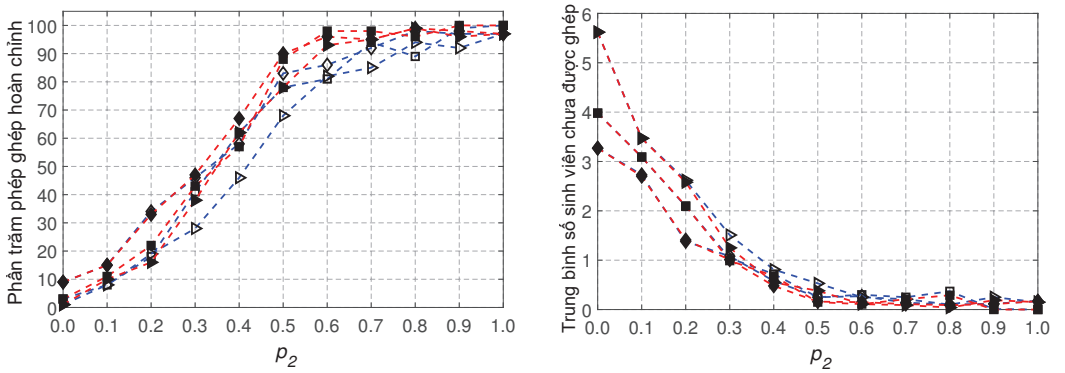
Thực nghiệm 4.8. Từ kết quả của *Thực nghiệm 4.7*, chúng ta có thể thấy rằng khi $p_1 = 0.8$, chất lượng nghiệm mà HAG tìm được tốt hơn của APX. Do đó, luận án tạo 100 thể hiện SPA-ST với $n \in \{300, 400\}$, p_1 được tăng từ 0.81 lên 0.88 với bước tăng 0.01, các tham số m, q, p_2, C và D được thiết lập giống như trong *Thực nghiệm 4.7*. Luận án so sánh thời gian thực hiện trung bình và số lần lặp lại trung bình của HAG và APX tìm được.

Hình 4.8 chỉ ra thời gian thực hiện trung bình của HAG và APX với mọi giá trị là p_1 và p_2. Khi p_2 tăng từ 0.0 lên 0.9, thời gian thực hiện trung bình của HAG và APX giảm cho mọi giá trị của p_1. Tuy nhiên, khi $p_2 = 1.0$, thời gian thực hiện trung bình của HAG tăng nhưng nhỏ hơn thời gian thực hiện của APX. Do đó, HAG chạy nhanh hơn APX cho mọi giá trị p_1 và p_2. Hình 4.9 chỉ ra rằng số lần lặp lại trung bình của HAG và APX cho mọi giá trị của p_2. Khi p_2 giảm từ 0.9 xuống 0.0, số lần lặp trung bình của HAG và APX sẽ tăng. Khi $p_2 = 1.0$, HAG thực hiện nhiều lần lặp hơn so với APX để tìm các kết quả phép ghép hoàn chỉnh.

Tiếp theo, luận án so sánh chất lượng nghiệm tìm được bởi HAG và APX. Hình 4.10(a)



(a) —— $n = 300$, $m = 15$, $q = 30$ —— (b)

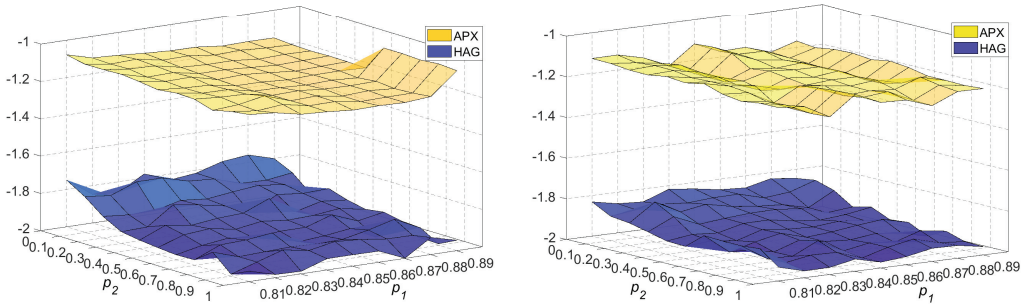


(c) —— $n = 400$, $m = 20$, $q = 40$ —— (d)

Hình 4.10: *Phần trăm phép ghép hoàn chỉnh và trung bình số sinh viên chưa được ghép của* HAG *và* APX

và 4.10(c) chỉ ra phần trăm phép ghép hoàn chỉnh HAG và APX. chúng ta thấy rằng, khi p_1 thay đổi từ 0.86 đến 0.88, tức là số lượng các cặp được chấp nhận trong danh sách xếp hạng của sinh viên và giảng viên giảm, HAG nhận thấy phần trăm phép ghép hoàn chỉnh cao hơn so với APX với $n = 300$ và 400. Lưu ý rằng khi $p_1 \in [0.81, 0.85]$, cả HAG và APX đều tìm được phần trăm phép ghép hoàn chỉnh, do đó luận án không chỉ ra trong các Hình này. Khi p_1 tăng, có nghĩa là việc tìm kiếm phép ghép hoàn chỉnh là rất khó, tuy nhiên, HAG vẫn tìm được phần trăm phép ghép hoàn chỉnh hơn so với APX. Hình 4.10(b) và 4.10(d) mô tả trung bình số sinh viên chưa được ghép của HAG và APX với các giá trị của p_1 từ 0.86 đến 0.88. Khi p_2 tăng, trung bình số sinh viên chưa được ghép của HAG sẽ nhỏ hơn của APX. Với $n = 300$, HAG lấy từ $10^{-2.31}$ giây đến $10^{-2.24}$ giây, trong khi APX lấy từ $10^{-1.78}$ giây đến $10^{-1.76}$ giây. Với $n = 400$, HAG lấy từ $10^{-2.09}$ giây đến $10^{-2.01}$ giây, trong khi APX lấy từ $10^{-1.45}$ giây đến $10^{-1.43}$ giây tương ứng với các giá trị lần lượt là $p_1 = 0.88$ và $p_1 = 0.81$. Trung bình, HAG cần 332 lần lặp, trong khi APX cần thêm khoảng 350 lần lặp với $n = 300$. Với $n = 400$, APX cần 468 lần lặp, trong khi HAG chỉ cần 447 lần lặp để tìm các phép ghép hoàn chỉnh.

Thực nghiệm 4.9. Thực nghiệm này, luận án tạo 100 thể hiện SPA-ST với các tham



(a) $C = 1.4n$, $0.7C \leq D \leq 0.9C$ (b) $C = 1.5n$, $D = 1.2n$

Hình 4.11: *Trung bình thời gian thực hiện của* HAG *và* APX *với* $n = 500$, $m = 25$, $q = 50$

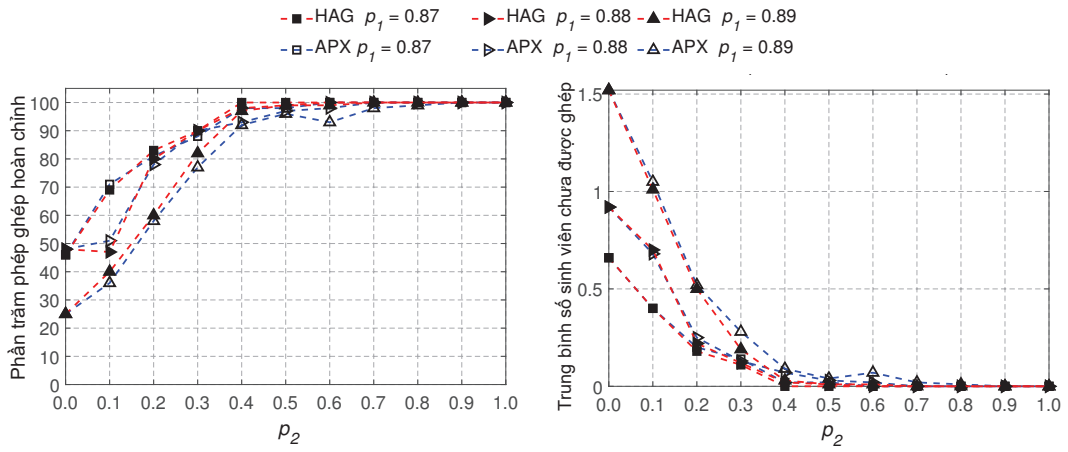
số $(n, m, q, p_1, p_2, C, D)$, trong đó $n = 500$, $m = 0.05n$, $q = 0.1n$, $p_1 \in [0.81, 0.89]$ với bước tăng 0.01 và $p_2 \in [0.0, 1.0]$ với bước tăng 0.1. Tổng số sinh viên tối đa của đề tài và giảng viên bao gồm hai trường hợp:

Trường hợp 1: $C = 1.4n$ và $0.7C \leq D \leq 0.9C$, nghĩa là số sinh viên tối đa c_j của mỗi đề tài p_j sao cho $0.6C/q \leq c_j \leq 1.4C/q$ và số sinh viên tối đa của mỗi giảng viên l_k sao cho $0.7\sum_{k=1}^{|P_k|} c_j \leq d_k \leq 0.9\sum_{k=1}^{|P_k|} c_j$, trong đó P_k là tập các đề tài p_j được hướng dẫn với bởi giảng viên l_k.

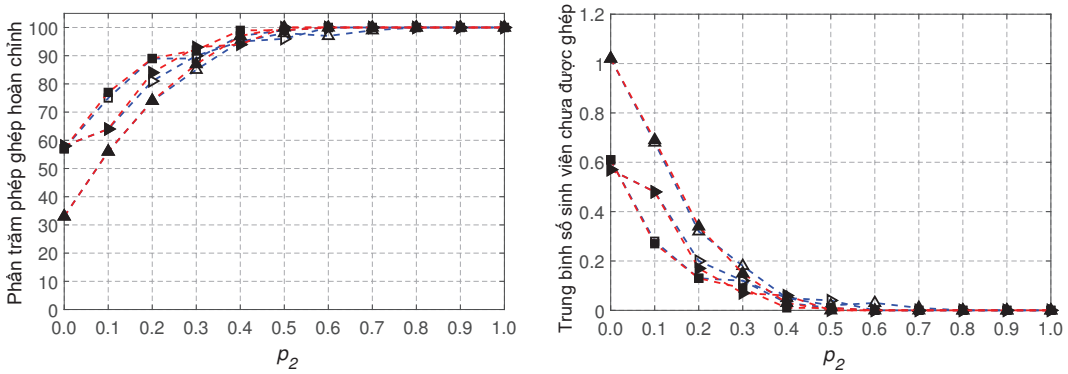
Trường hợp 2: $C = 1.5n$, và $D = 1.2n$, nghĩa là số sinh viên tối đa c_j của mỗi đề tài p_j sao cho $0.6C/q \leq c_j \leq 1.4C/q$ và số sinh viên tối đa của mỗi giảng viên l_k sao cho $d_k = D/m$.

Trước hết, luận án so sánh thời gian thực hiện trung bình của HAG và APX. Hình 4.11 chỉ ra rằng HAG chạy nhanh hơn APX trong cả hai trường hợp với mọi giá trị của p_1 và p_2. Trong *Trường hợp 1*, rõ ràng là HAG chỉ cần $10^{-1.89}$ giây, trong khi APX cần khoảng $10^{-1.14}$ giây. Trong *Trường hợp 2*, HAG cần khoảng $10^{-1.91}$ giây, trong khi APX cần khoảng $10^{-1.17}$ giây. chúng ta thấy rằng HAG chạy nhanh hơn 6 lần so với APX.

Tiếp theo, luận án so sánh chất lượng nghiệm tìm được bởi HAG và APX. Trung bình trên p_2, chúng ta thấy rằng, khi p_1 thay đổi từ 0.81 đến 0.86, phần trăm phép ghép hoàn chỉnh do HAG và APX tìm được tốt hơn phần trăm phép ghép hoàn chỉnh tìm được trong Thực nghiệm 4.8 ($> 90\%$). Khi p_1 tăng từ 0.87 lên 0.89, chất lượng nghiệm tìm được của HAG tốt hơn so với chất lượng nghiệm của APX tìm được, trong đó trung bình số sinh viên chưa được ghép giảm, trong khi phần trăm kết quả phép ghép hoàn chỉnh tăng trong Hình 4.12. Hình. 4.12(c) và 4.12(d) chỉ ra rằng chất lượng nghiệm trong *Trường hợp 2* tốt hơn *Trường hợp 1* trong Hình 4.12(a) và 4.12(c). Tiếp theo, luận án xem xét phần trăm phép ghép hoàn chỉnh tìm được bởi HAG và APX. Rõ ràng rằng, HAG không chỉ tìm được phần trăm phép ghép hoàn chỉnh cao hơn mà còn có số lượng sinh viên chưa được ghép ít hơn so với APX



(a) —— $n = 500$, $m = 25$, $q = 50$, $C = 1.4n$, $0.7C \leq D \leq 0.9C$ —— (b)

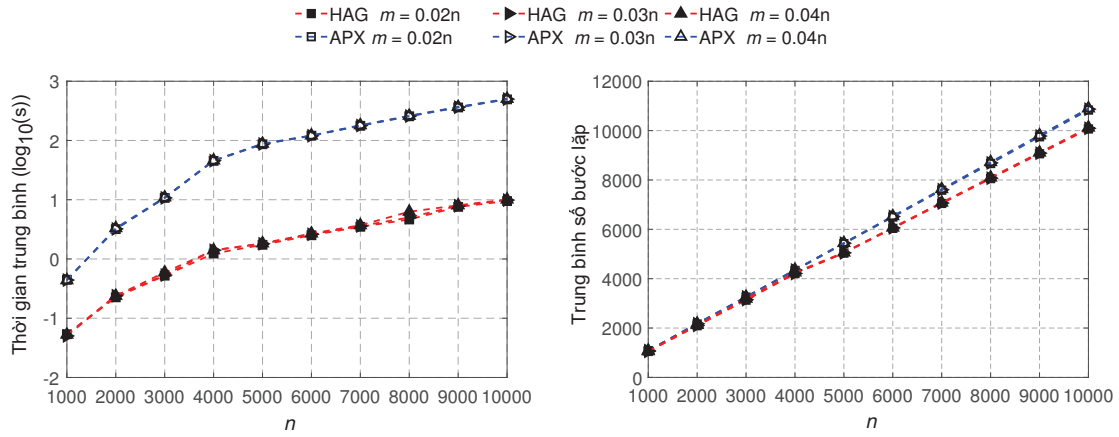


(d) —— $n = 500$, $m = 25$, $q = 50$, $C = 1.5n$, $D = 1.2n$ —— (c)

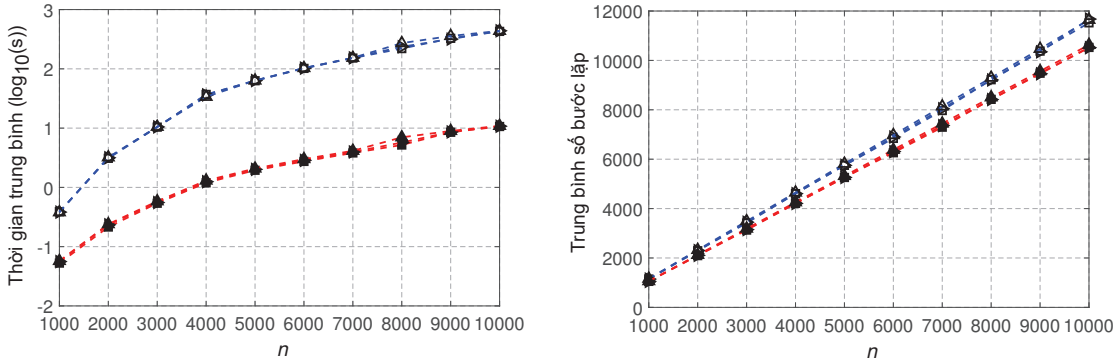
Hình 4.12: *Phần trăm phép ghép hoàn chỉnh và trung bình số sinh viên chưa được ghép* HAG *và* APX

trong các phép ghép ổn định. Trong *Trường hợp 1*, rõ ràng là HAG tìm được từ 82% đến 99.5%, trong khi APX chỉ tìm được từ 79% đến 99.5% các phép ghép hoàn chỉnh. Trong *Trường hợp 2*, HAG tìm được từ 86.1% đến 99.6%, trong khi APX tìm được từ 85.2% đến 99.6% các phép ghép hoàn chỉnh tương ứng với $p_1 = 0.89$ đến $p_1 = 0.81$. Do đó, chúng ta thấy rằng phần trăm phép ghép hoàn chỉnh tìm được bởi HAG và APX trong *Trường hợp 2* cao hơn *Trường hợp 1*. Những kết quả này chứng minh rằng HAG hiệu quả hơn APX về chất lượng nghiệm và thời gian thực hiện.

Thực nghiệm 4.10. Từ các kết quả trong Thực nghiệm 4.9, chúng ta thấy rằng khi tăng n và p_1, HAG tốt hơn APX về chất lượng nghiệm và thời gian thực hiện. Do đó, trong Thực nghiệm này, luận án chọn $n \in [1000, 10000]$ với bước tăng 1000, $m \in [0.02n, 0.04n]$ bước tăng 0.01, $q = 0.1n$, $p_1 = 0.9$, và $p_2 = 0.5$. Tổng số sinh viên tối đa của các đề tài và giảng viên trong hai trường hợp giống như trong Thực nghiệm 4.9. Mặc dù, khi $p_1 = 0.9$, nghĩa là 90% của các đề tài hoặc sinh viên không xếp hạng tương ứng trong danh sách xếp hạng của sinh viên hoặc danh sách xếp hạng của giảng viên. Trong thực nghiệm này, vì số lượng đề tài và sinh viên lớn hơn, nên cả hai thuật toán HAG và APX đều có thể tìm được



(a) —— $C = 1.4n, 0.7C \leq D \leq 0.9C$ —— (b)



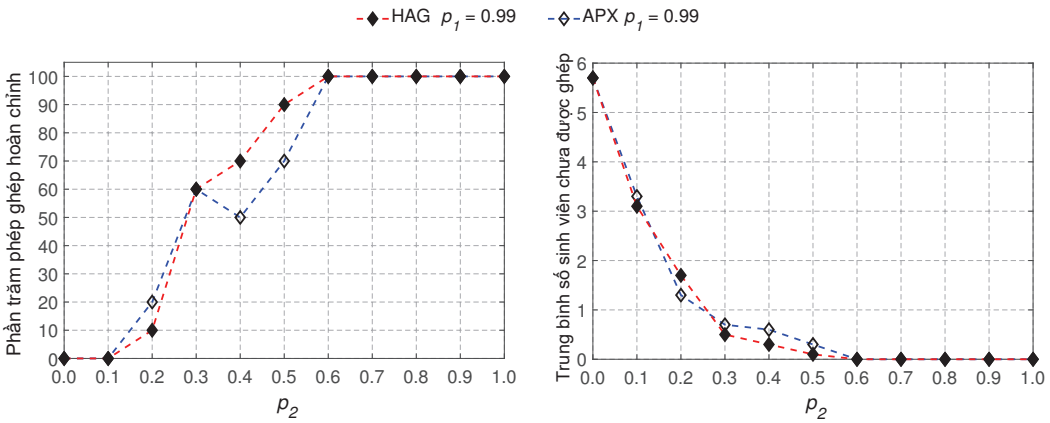
(c) —— $C = 1.5n, D = 1.2n$ —— (d)

Hình 4.13: *Trung bình thời gian thực hiện và số bước lặp của* HAG *và* APX

khoảng 100% các phép ghép hoàn chỉnh. Tuy nhiên, Hình 4.13 chỉ ra rằng HAG chạy nhanh hơn từ 10 đến 80 lần và cần ít lần lặp hơn so với APX trong hai trường hợp.

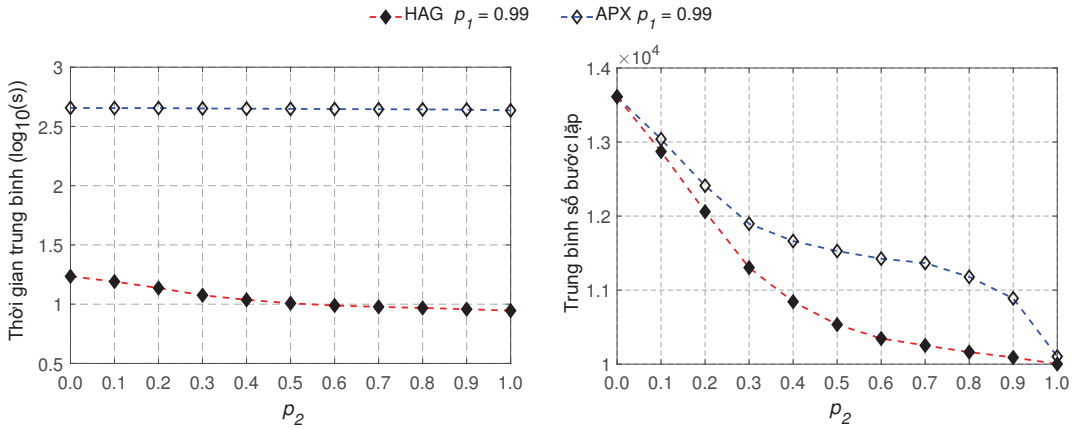
Thực nghiệm 4.11. Từ Thực nghiệm 4.10, chúng ta thấy rằng, khi $p_1 = 0.9$, HAG tốt hơn APX về thời gian thực hiện, nhưng phần trăm phép ghép hoàn chỉnh là như nhau $(\approx 100\%)$. Do đó, trong thực nghiệm này, luận án đã tạo 10 thể hiện SPA-ST với các tham số $n = 10000$, $m = 0.02n$, $q = 0.1n$, $p_1 \in [0.95, 0.99]$ với bước tăng 0.01, $p_2 \in [0.0, 1.0]$ với bước tăng 0.1, $C = 1.2n$ và $D = 1.1n$. Bởi vì khi $p_1 \in [0.95, 0.98]$, HAG và APX tìm được 100% các phép ghép hoàn chỉnh, do đó luận án không chỉ ra trong Hình 4.14. Hình 4.14(a) và 4.14(b) chỉ ra phần trăm phép ghép hoàn chỉnh và trung bình số sinh viên chưa được ghép của HAG và APX với $p_1 = 0.99$. Khi p_2 thay đổi từ 0.0 đến 1.0, chúng ta có 11 điểm để so sánh phần trăm phép ghép hoàn chỉnh mà HAG và APX tìm được, nhưng HAG chỉ tìm được phần trăm phép ghép hoàn chỉnh nhiều hơn so với APX ở 2 điểm $(\approx 18.2\%)$ bao gồm $p_2 = 0.4$ và 0.5. Trung bình của các sinh viên chưa được ghép do HAG tìm được nhỏ hơn so với APX tìm được.

Thời gian thực hiện trung bình theo p_1 và trung bình số lần lặp của HAG và APX được mô tả trong Hình 4.15. Rõ ràng rằng, HAG chỉ cần khoảng $10^{1.1}$ giây, trong khi APX phải cần



(a) Phần trăm phép ghép hoàn chỉnh (b)Trung bình số sinh viên chưa được ghép

Hình 4.14: *Chất lượng nghiệm của* HAG *và* APX *với* $n = 10000$, $m = 200$, $q = 1000$



(a) Trung bình thời gian thực hiện (b)Trung bình số bước lặp

Hình 4.15: HAG *và* APX *với* $n = 10000$, $m = 200$, $q = 1000$

nhiều thời gian hơn khoảng $10^{2.7}$ giây trong Hình 4.15(a). Điều này chỉ ra HAG chạy nhanh hơn APX khoảng 50 lần. Hình 4.15(b) chỉ ra rằng HAG sử dụng ít lần lặp hơn so với APX. Từ những kết quả thực nghiệm này, chúng ta thấy rằng HAG tốt hơn APX về phép ghép hoàn chỉnh và thời gian thực hiện cho bài toán SPA-ST có kích thước lớn.

4.4. Kết luận Chương 4

Bài toán SPA là một biến thể quan trọng của bài toán SMTI [1, 53]. Mặc dù đã có nhiều nghiên cứu để giải quyết bài toán này, tuy nhiên các thuật toán đã đề xuất chưa hiệu quả về thời gian tính toán và chất lượng nghiệm cho bài toán MAX-SPA, đây là một bài toán NP-khó, do đó Chương này trình bày hai thuật toán tìm kiếm heuristic để giải quyết bài toán MAX-SPA-P và MAX-SPA-ST với kích thước lớn.

Thuật toán SPA-P-heuristic giải quyết bài toán SPA-P. Mục tiêu của bài toán là tìm một phép ghép ổn định với kích thước tối đa (MAX-SPA-P). Ý tưởng chính của thuật toán là

cải tiến thuật toán GS [3, 1] bằng cách đề xuất một hàm heuristic đơn giản và hiệu quả trong việc định hướng quá trình sinh viên chọn đề tài để ghép. Các kết quả thực nghiệm đã chỉ ra rằng thuật toán SPA-P-heuristic vượt trội so với SPA-P-approx và SPA-P-promotion về thời gian thực hiện và chất lượng nghiệm cho bài toán MAX-SPA-P kích thước lớn.

Thuật toán HAG giải quyết bài toán SPA-ST với kích thước lớn. Mục tiêu của bài toán là tìm một phép ghép ổn định với kích thước tối đa (MAX-SPA-ST). Thuật toán HAG bắt đầu từ một phép ghép rỗng, mục đích là tìm một phép ghép ổn định với kích thước tối đa bằng cách xác định hai hàm heuristic mới để cải thiện hiệu suất của quá trình tìm kiếm. Nếu thuật toán HAG tìm được phép ghép không hoàn chỉnh, HAG đề xuất chiến lược heuristic để tăng kích thước của phép ghép ổn định. Kết quả thực nghiệm cho thấy thuật toán HAG hiệu quả hơn nhiều so với thuật toán APX về thời gian thực hiện và chất lượng nghiệm cho MAX-SPA-ST kích thước lớn.

KẾT LUẬN

Bài toán SMP là một bài toán nổi tiếng thuộc lớp bài toán thỏa mãn ràng buộc (CSP) được giới thiệu bởi Gale và Shapley vào năm 1962 [1]. Gần đây, bài toán SMP đã nhận được nhiều sự quan tâm của các nhà nghiên cứu trong lĩnh vực Trí tuệ nhân tạo và Tính toán tối ưu bởi các ứng dụng quan trọng của nó trong thực tế. Vì vậy luận án tập trung nghiên cứu tổng quan về bài toán SMP và các biến thể bao gồm SMTI, HRT và SPA. Dựa vào cơ sở lý thuyết và tình hình nghiên cứu của các bài toán trong Chương 1, luận án đã đề xuất các thuật toán tìm kiếm heuristics hiệu quả để giải quyết các bài toán này với kích thước lớn. Các kết quả được trình bày chi tiết trong các Chương 2, Chương 3 và Chương 4.

Các kết quả đạt được. Các kết quả đạt được của luận án bao gồm:

- Đề xuất hai thuật toán MCS và HR hiệu quả giải quyết bài toán MAX-SMTI được công bố trong Chương 2. Các kết quả thực nghiệm đã chỉ ra rằng các thuật toán đề xuất vượt trội về thời gian tính toán và chất lượng nghiệm so với các thuật toán tìm kiếm cục bộ LTIU [45], AS [14] và GSA2 [62] cho bài toán MAX-SMTI.
- Đề xuất hai thuật toán gồm MCA và HS hiệu quả giải quyết bài toán MAX-HRT được công bố trong Chương 3. Các kết quả thực nghiệm trên cùng một bộ dữ liệu được tạo ngẫu nhiên với các tham số được thiết lập trên cùng một môi trường cho thấy rằng các thuật toán đề xuất hiệu quả hơn thuật toán LTIU [11] và thuật toán HP [62] về thời gian thực hiện và chất lượng nghiệm cho bài toán MAX-HRT kích thước lớn.
- Đề xuất hai thuật toán gồm SPA-P-heuristic và HAG cho bài toán SPA-P và SPA-ST được công bố trong Chương 4. Các kết quả thực nghiệm chứng minh rằng thuật toán đề xuất vượt trội về thời gian thực hiện và chất lượng nghiệm so với các thuật toán SPA-P-approx [3], SPA-P-promotion [102], SPA-P-MCH [118] và APX [94] cho bài toán MAX-SPA-P và MAX-SPA-ST kích thước lớn.

Các điểm hạn chế và tồn tại. Bên cạnh các kết quả nghiên cứu đã đạt được, các kết quả trong luận án còn tồn tại một số điểm hạn chế như sau:

- Các thuật toán đã đề xuất được thực nghiệm trên các bộ dữ liệu được tạo ngẫu nhiên, hiện tại chưa có các bộ dữ liệu trong thực tế để áp dụng và đánh giá cho các bài toán nghiên cứu trong Luận án.
- Các biến thể khác của bài toán hôn nhân ổn định với các ràng buộc có giới hạn trên

và giới hạn dưới trong danh sách xếp hạng chưa được nghiên cứu trong Luận án.

Hướng phát triển. Từ các kết quả đạt được và những hạn chế trong luận án này, trong tương lai luận án sẽ tiếp tục các hướng phát triển của bài toán như sau:

- Tiếp tục nghiên cứu các thuật toán heuristics hiệu quả cho các biến thể khác của bài toán hôn nhân ổn định.
- Nghiên cứu các hướng tiếp cận khác cho bài toán hôn nhân ổn định và các biến thể.
- Triển khai, áp dụng các thuật toán đề xuất trong luận án cho các ứng dụng thực tiễn trong nhiều lĩnh vực khác như y tế, giáo dục, kinh tế và xã hội bao gồm hệ thống hỗ trợ tuyển sinh đại học, tìm kiếm việc làm, hệ thống phân phối hàng hóa, và hệ thống phân chia hạ tầng mạng 5G trong tương lai.

DANH MỤC CÁC CÔNG TRÌNH ĐÃ CÔNG BỐ CỦA NGHIÊN CỨU SINH VÀ CỘNG SỰ

1. Các công bố sử dụng trong Luận án

[A.1] Hoang Huu Viet, **Nguyen Thi Uyen**, SeungGwan Lee,TaeChoong Chung, and Le Hong Trang, "*A Max-Conflicts based Heuristic Search for the Stable Marriage Problem with Ties and Incomplete Lists*", Journal of Heuristics (**SCIE - Q2**), vol. 27, no.3, pp. 439–458, 2021.

[A.2] Hoang Huu Viet, **Nguyen Thi Uyen**, Cao Thanh Sơn, and TaeChoong Chung: *A Heuristic Repair Algorithm for the Maximum Stable Marriage Problem with Ties and Incomplete Lists*, in Proceedings of the 34^{th} Australasian Joint Conference on Artificial Intelligence 2022 (AI 2022), Sydney, Australia, Feb.2-4, 2022, pp.494-506, Lecture Notes in Artificial Intelligence 13151 (**SCOPUS**), Springer, ISBN 978-3-030-97545-6.

[A.3] **Nguyen Thi Uyen**, Nguyen Long Giang, Nguyen Truong Thang, and Hoang Huu Viet: *A min-conflicts algorithm for maximum stable matchings of the hospitals/residents problem with ties*, in Proceedings of the 14^{th} International Conference on Computing and Communication Technologies (RIVF 2020), RMIT, Ho Chi Minh, Apr.6-7, 2020, pp.1-6, Lecture Notes in Computer Science (**SCOPUS**), Springer, ISBN 978-1-7281-5377-3.

[A.4] **Nguyen Thi Uyen**, Nguyen Long Giang, Tran Xuan Sang and Hoang Huu Viet "*An efficient heuristics algorithm for solving the Student-Project Allocation with Preferences over Projects*", 24^{th} Hội thảo Quốc gia (VNICT 2021), Thai Nguyen, Việt Nam, Dec. 13-14, pp. 1-6, 2021.

[A.5] **Nguyen Thi Uyen**, Giang L. Nguyen, Canh V. Pham, Tran Xuan Sang and Hoang Huu Viet: "*A Heuristic Algorithm for the Student-Project Allocation Problem with Lecturer Preferences over Students with Ties*, in Proceedings of the 11^{th} International Conference on Computational Data and Social Networks (CSoNET 2022), Tampa, Florida, USA, Dec. 5-7, 2022, in Press, Lecture Notes in Computer Science (**SCOPUS**), Springer.

[B.1] **Nguyen Thi Uyen**, Giang L. Nguyen and Hoang Huu Viet, "*An efficient Heuristic*

search algorithm for the Hospitals/Residents with Ties problem", Applied Artificial Intelligence (đang gửi tạp chí).

[B.2] **Nguyen Thi Uyen**, Nguyen Long Giang and Hoang Huu Viet "*Faster and Simpler Heuristic Algorithm for the Student-Project Allocation with Preferences over Projects*", International Journal of Fuzzy Logic and Intelligent Systems (đang gửi tạp chí).

2. Other publications

[C.1] **Nguyen Thi Uyen** and Tran Xuan Sang, "*An efficient algorithm to find a maximum weakly stable matching for SPA-ST problem*, in Proceedings of the 21^{st} International Conference on Artificial Intelligence and Soft Computing (ICAISC 2022), Zakopane, Poland, Jun. 18-22, 2022, in Press, Lecture Notes in Artificial Intelligence (**SCOPUS**), Springer.

[C.2] Hoang Huu Viet, **Nguyen Thi Uyen**, Cao Thanh Sơn, and Le Hong Trang, "*Một thuật toán tìm kiếm cục bộ giải bài toán phân công địa điểm thực tập cho sinh viên*", 23^{th} Hội thảo Quốc gia (VNICT 2020), Hạ Long, Việt Nam, Nov. 5-6, pp. 271–276, 2020.

TÀI LIỆU THAM KHẢO

Tiếng Anh:

[1] D. Gale and L. S. Shapley (1962), *College admissions and the stability of marriage*, The American Mathematical Monthly, 9 (1), pp. 9–15.

[2] David F. Manlove (2008), *The Hospitals/Residents Problem. In: Kao MY. (eds) Encyclopedia of Algorithms*, Springer, Boston, MA, ISBN: 0978-0-387-30162-4.

[3] David J. Abraham, Robert W. Irving, and David F. Manlove, *The Student-Project Allocation Problem*, in: Proceedings of the 14th International Symposium, Kyoto, Japan, 2003, pp. 474–484.

[4] Katarína Cechlárová and Tamás Fleiner (2005), *On a generalization of the stable roommates problem*, ACM Transactions on Algorithms (TALG), 1 (1), pp. 143–156.

[5] Tamas Fleiner, Robert W. Irving, and David F. Manlove (2007), *Efficient algorithms for generalized Stable Marriage and Roommates problems*, Theoretical Computer Science, 381 (1-3), pp. 162–176.

[6] L Bing-Hong, P Van-Trung, and N Tu.N, *A virtual backbone construction heuristic for maximizing the lifetime of dual-radio wireless sensor networks*, in: IIH-MSP, IEEE, 2015, pp. 64–67.

[7] Kashyab J Ambarani et al., *Enforcing Resource Allocation and VNF Embedding in RAN Slicing*, in: 2021 IEEE Global Communications Conference, IEEE, 2021, pp. 1–6.

[8] Robert W Irving and David F Manlove (2002), *The stable roommates problem with ties*, Journal of Algorithms, 43 (1), pp. 85–105.

[9] David F. Manlove et al. (2002), *Hard Variants of Stable Marriage*, Theoretical Computer Science, 276 (1-2), pp. 261–279.

[10] Robert W Irving, David F Manlove, and Sandy Scott, *Strong stability in the hospitals/residents problem*, in: Annual Symposium on Theoretical Aspects of Computer Science, Springer, 2003, pp. 439–450.

[11] Mirco Gelain et al., *Local Search for Stable Marriage Problems with Ties and Incomplete Lists*, in: Proceedings of 11th Pacific Rim International Conference on Artificial Intelligence, Daegu, Korea, 2010, pp. 64–75.

[12] Adam Kunysz, *An Algorithm for the Maximum Weight Strongly Stable Matching Problem*, in: 29th International Symposium on Algorithms and Computation (ISAAC 2018), vol. 123, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018, 42:1–42:13.

[13] Sofiat Olaosebikan and David Manlove (2020), *Super-stability in the student-project allocation problem with ties*, Journal of Combinatorial Optimization, pp. 1–37.

[14] Danny Munera et al., *Solving Hard Stable Matching Problems via Local Search and Cooperative Parallelization*, in: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, Texas, 2015, pp. 1212–1218.

[15] Deeksha Adil et al. (2018), *Parameterized algorithms for stable matching with ties and incomplete lists*, Theoretical Computer Science, 723 (1), pp. 1–10.

[16] David J. Abraham, Robert W. Irving, and David F. Manlove (2007), *Two algorithms for the Student-Project Allocation problem*, Journal of Discrete Algorithms, 5 (1), pp. 73–90.

[17] Kazuo Iwama, Shuichi Miyazaki, and Naoya Yamauchi, *A 1.875: approximation algorithm for the stable marriage problem*, in: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, New Orleans, Louisiana, 2007, pp. 288–297.

[18] Magnus M. Halldórsson, Kazuo Iwama, and Shuichi Miyazaki (2007), *Improved Approximation Results for the Stable Marriage Problem*, ACM Transactions on Algorithms, 3 (3), pp. 1–18.

[19] David Manlove, Duncan Milne, and Sofiat Olaosebikan, *An integer programming approach to the student-project allocation problem with preferences over projects*, in: International Symposium on Combinatorial Optimization, Springer, 2018, pp. 313–325.

[20] Augustine Kwanashie and David 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, 2013, pp. 263–269.

[21] Hoang Huu Viet et al., *A Bidirectional Local Search for the Stable Marriage Problem*, in: Proceedings of the 2016 International Conference on Advanced Computing and Applications (ACOMP), Can Tho City, Vietnam, 2016, pp. 18–24.

[22] Le Hong Trang, Hoang Huu Viet, and TaeChoong Chung, *Finding "optimal" stable marriages with ties via local search*, in: Proceedings of the 2016 Eighth International Conference on Knowledge and Systems Engineering (KSE2016), Hanoi, Vietnam, 2016, pp. 61–66.

[23] Hoang Huu Viet et al., *An Empirical Local Search for the Stable Marriage Problem*, in: Proceedings of the 14th Pacific Rim International Conference on Artificial Intelligence - PRICAI 2016: Trends in Artificial Intelligence, Phuket, Thailand, 2016, pp. 556–564.

[24] Ngo Anh Vien et al., *Ant Colony based Algorithm for Stable Marriage Problem*, in: Advances and Innovations in Systems, Computing Sciences and Software Engineering, Seattle, WA, 2007, pp. 457–461.

[25] Rakesh V Vohra (2012), *Stable matchings and linear programming*, Current Science, pp. 1051–1055.

[26] Maxence Delorme et al., *Mathematical models for stable matching problems with ties and incomplete lists*, 2019.

[27] Kazuo IwamaShuichi MiyazakiYasufumi MoritaDavid Manlove, *Stable Marriage with Incomplete Lists and Ties*, in: Proceedings of 26th International Colloquium on Automata, Languages, and Programming, Prague, Czech Republic, 1999, pp. 443–452.

[28] Robert W. Irving and David F. Manlove (2009), *Finding large stable matchings*, Journal of Experimental Algorithmics, 14 (2), 1.2–1.2:30.

[29] Alvin E.Roth (1984), *The Evolution of the Labor Market for Medical Interns and Residents: A Case Study in Game Theory*, Journal of Pilitical Economy, 92 (6), pp. 991–1016.

[30] Robert W. Irving, *Matching Medical Students to Pairs of Hospitals: A New Variation on a Well-known Theme*, in: in Proceedings of ESA 1998: the 6th Annual European Symposium, Venice, Italy, 1998, pp. 381–392.

[31] *Canadian Resident Matching Service (CaRMS)*, http://www.carms.ca/.

[32] Bruce M Maggs and Ramesh K Sitaraman (2015), *Algorithmic nuggets in content delivery*, ACM SIGCOMM Computer Communication Review, 45 (3), pp. 52–66.

[33] Kazuo Iwama, Shuichi Miyazaki, and Hiroki Yanagisawa (2010), *Approximation algorithms for the sex-equal stable marriage problem*, ACM Transactions on Algorithms, 7 (1), 2:1–2:17.

[34] M. Nakamura et al., *Genetic algorithm for sex-fair stable marriage problem*, in: Circuits and Systems, 1995. ISCAS '95., 1995 IEEE International Symposium on, Seattle, WA, 1995, pp. 509–512.

[35] Hoang Huu Viet et al., *A Max-Min Conflict Algorithm for the Stable Marriage Problem*, in: Pacific Rim Knowledge Acquisition Workshop, Springer, 2019, pp. 44–53.

[36] Hoang Huu Viet et al. (2020), *A shortlist-based bidirectional local search for the stable marriage problem*, Journal of Experimental & Theoretical Artificial Intelligence, 32 (1), pp. 147–163.

[37] Bertrand Zavidovique, Nikom Suvonvorn, and Guna Seetharaman, *A novel representation and algorithms for (quasi) stable marriages*, in: 2005 Proceedings of the Second International Conference on Informatics in Control, Automation and Robotics (ICINCO), Barcelona, Spain, 2005, pp. 63–70.

[38] Kazuo Iwama, Shuichi Miyazaki, and Hiroki Yanagisawa (2010), *Approximation algorithms for the sex-equal stable marriage problem*, ACM Transactions on Algorithms (TALG), 7 (1), pp. 1–17.

[39] Patricia Everaere, Maxime Morge, and Gauthier Picard, *Minimal Concession Strategy for Reaching Fair, Optimal and Stable Marriages*, in: Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems (AAMAS), St. Paul, MN, USA, 2013, pp. 1319–1320.

[40] Ioannis Giannakopoulos et al., *An Equitable Solution to the Stable Marriage Problem*, in: 2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI), Vietri sul Mare, Italy, 2015, pp. 989–996.

[41] Boming Zhao et al., *Preference-aware task assignment in on-demand taxi dispatching: An online stable matching approach*, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 01, 2019, pp. 2245–2252.

[42] Akhmad Alimudin and Yoshiteru Ishida (2022), *Matching-updating mechanism: A solution for the stable marriage problem with dynamic preferences*, Entropy, 24 (2), p. 263.

[43] Philippe Codognet and Daniel Diaz, *Yet Another Local Search Method for Constraint Solving*, in: Proceedings of the International Symposium on Stochastic Algorithms, Berlin, Germany, 2001, pp. 73–90.

[44] Ian Philip Gent and Patrick 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, 2002, pp. 141–145.

[45] Mirco Gelain et al. (2013), *Local Search Approaches in Stable Matching Problems*, Algorithms, 6 (1), pp. 591–617.

[46] Satoshi Tayu and Shuichi Ueno, *Stable Matchings in Trees*, in: International Computing and Combinatorics Conference, Springer, 2017, pp. 492–503.

[47] Robert W. Irving (1994), *Stable marriage and indifference*, Discrete Applied Mathematics, 48 (3), pp. 261–272.

[48] Kazuo Iwama and Shuichi Miyazaki, *A Survey of the Stable Marriage Problem and Its Variants*, in: Proceedings of the International Conference on Informatics Education and Research for Knowledge-Circulating Society, Washington, DC, USA, 2008, pp. 131–136.

[49] Magnus M. Halldórsson et al., *Improved Approximation of the Stable Marriage Problem*, in: Proceedings of 11th Annual European Symposium on Algorithms, Budapest, Hungary, 2003, pp. 266–277.

[50] Ildikó Schlotter Dániel Marx (2010), *Parameterized Complexity and Local Search Approaches for the Stable Marriage Problem with Ties*, Algorithmica, 58 (1), pp. 170–187.

[51] Keita Nakamura and Naoyuki Kamiyama (2016), *Many-to-many stable matchings with ties in trees*, Journal of the Operations Research Society of Japan, 59 (3), pp. 225–240.

[52] Kazuo Iwama, Shuichi Miyazaki, and Kazuya Okamotoe, *A $(2 - c\frac{logN}{N})$–Approximation Algorithm for the Stable Marriage Problem*, in: Proceedings of the 9th Scandinavian Workshop on Algorithm Theory, Humlebek, Denmark, 2004, pp. 349–361.

[53] Kazuo Iwama et al., *Stable Marriage with Incomplete Lists and Ties*, in: Proceedings of International Colloquium on Automata, Languages, and Programming, Prague, Czech Republic, 1999, pp. 443–452.

[54] Dan Gusfield and Robert W. Irving (1989), *The stable marriage problem: structure and algorithms*, MIT Press Cambridge, ISBN: 0-262-07118-5.

[55] Robert W.Irving and David F.Manlove (2009), *Finding Large Stable Matchings*, Journal of Experimental Algorithmics, 14 (1), pp. 1–2.

[56] Ian P. Gent et al., *A Constraint Programming Approach to the Stable Marriage Problem*, in: Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming, vol. 1, Berlin, Heidelberg, 2001, pp. 225–239.

[57] Kazuo Iwama, Shuichi Miyazaki, and Naoya Yamauchi, *A $(2 - c\frac{1}{\sqrt{N}})$ – Approximation Algorithm for the Stable Marriage Problem*, in: Proceedings of the 16th international conference on Algorithms and Computation, Sanya, Hainan, China, 2005, pp. 902–914.

[58] Robert W.Irving, David F.Manlove, and Sandy Scott (2008), *The stable marriage problem with master preference lists*, Discrete Applied Mathematics, 156 (15), pp. 2959–2977.

[59] Eric McDermid, *A 3/2-Approximation Algorithm for General Stable Marriage*, in: Proceedings of the 36th International Colloquium on Automata, Languages, and Programming, Rhodes, Greece, 2009, pp. 689–700.

[60] David F Manlove and Gregg O'Malley (2008), *Student-project allocation with preferences over projects*, Journal of Discrete Algorithms, 6 (4), pp. 553–560.

[61] Katarzyna Paluch, *Faster and Simpler Approximation of Stable Matchings*, in: Proceedings of the 9th International Workshop on Approximation and Online Algorithms, Saarbrucken, Germany, 2011, pp. 176–187.

[62] Zoltan Király (2013), *Linear Time Local Approximation Algorithm for Maximum Stable Marriage*, Algorithms, 6 (1), pp. 471–484.

[63] Katarzyna Paluch (2014), *Faster and Simpler Approximation of Stable Matchings*, Algorithms, 7 (2), pp. 189–202.

[64] Koki Hamada, Kazuo Iwama, and Shuichi Miyazaki (2016), *The hospitals-residents problem with lower quotas*, Algorithmica, 74 (1), pp. 440–465.

[65] Chi-Kit Lam and C Gregory Plaxton, *Maximum stable matching with one-sided ties of bounded length*, in: International Symposium on Algorithmic Game Theory, Springer, 2019, pp. 343–356.

[66] BS Panda et al., *Hardness and Approximation Results for Some Variants of Stable Marriage Problem*, in: Conference on Algorithms and Discrete Applied Mathematics, Springer, 2022, pp. 252–264.

[67] Danny Munera et al., *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, 2015, pp. 66–77.

[68] William Pettersson et al. (2021), *Improving solution times for stable matching problems through preprocessing*, Computers & Operations Research, 128, p. 105128.

[69] Christian Haas (2021), *Two-sided matching with indifferences: Using heuristics to improve properties of stable matchings*, Computational Economics, 57 (4), pp. 1115–1148.

[70] Joanna Drummond, Andrew Perrault, and Fahiem Bacchus, *SAT is an effective and complete method for solving stable matching problems with couples*, in: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.

[71] Georgios Askalidis et al., *Socially Stable Matchings in the Hospitals/Residents Problem*, in: Proceedings of the 13th international conference on Algorithms and Data Structures, London, ON, Canada, 2013, pp. 85–96.

[72] Esra Erdem et al. (2020), *A general framework for stable roommates problems using answer set programming*, Theory and Practice of Logic Programming, 20 (6), pp. 911–925.

[73] Kitty Meeks and Baharak Rastegari (2020), *Solving hard stable matching problems involving groups of similar agents*, Theoretical Computer Science, 844, pp. 171–194.

[74] Haris Aziz et al. (2020), *Stable Matching with Uncertain Linear Preferences*, Algorithmica, 82 (1), pp. 1410–1433.

[75] Robert W. Irving, David F. Manlove, and Sandy Scott, *The Hospitals/Residents Problem with Ties*, in: in Proceedings of the 7th Scandinavian Workshop on Algorithm Theory, Bergen, Norway, 2000, pp. 259–271.

[76] Kazuo Iwama, Shuichi Miyazaki, and Naoya Yamauchi (2008), *A* $(2 - c\frac{1}{\sqrt{N}})$ *– Approximation Algorithm for the Stable Marriage Problem*, Algorithmica, 51 (1), pp. 342–356.

[77] Zoltan Király (2011), *Better and simpler approximation algorithms for the stable marriage problem*, Algorithmica, 60 (1), pp. 3–20.

[78] David F. Manlove and Gregg O'Malley, *Modelling and Solving the Stable Marriage Problem Using Constraint Programming*, in: Proceedings of the Fifth Workshop on Modelling and Solving Problems with Constraints, IJCAI'05, 2005, pp. 10–17.

[79] Danny Munera et al., *A Local Search Algorithm for SMTI and its extension to HRT Problems*, in: Proceedings of the 3rd International Workshop on Matching Under Preferences, Glasgow, United Kingdom, 2015, pp. 66–77.

[80] Qi Yue, Wenchang Zou, and Wen Hu (), *A new theory of triangular intuitionistic fuzzy sets to solve the two-sided matching problem*.

[81] Zhen Zhang et al. (2019), *Stable two-sided matching decision making with incomplete fuzzy preference relations: A disappointment theory based approach*, Applied Soft Computing, 84, p. 105730.

[82] Chi-Kit Lam and C Gregory Plaxton (2022), *Maximum stable matching with one-sided ties of bounded length*, Theory of Computing Systems, pp. 1–34.

[83] Mert Kimya (2022), *Farsighted objections and maximality in one-to-one matching problems*, Journal of Economic Theory, p. 105499.

[84] Eduard Eiben et al. (2023), *Preference swaps for the stable matching problem*, Theoretical Computer Science, 940, pp. 222–230.

[85] Vijay Kumar Garg, *Keynote Talk: Lattice Linear Predicate Algorithms for the Constrained Stable Marriage Problem with Ties*, in: 24th International Conference on Distributed Computing and Networking, 2023, pp. 2–11.

[86] Christoph Roch et al., *A Quantum Annealing Approach for Solving Hard Variants of the Stable Marriage Problem*, in: International Conference on Innovations for Community Services, 2022, pp. 294–307.

[87] Nitsan Perach and Shoshana Anily (2022), *Stable matching of student-groups to dormitories*, European Journal of Operational Research, 302 (1), pp. 50–61.

[88] Rohan Chowdhury (2022), *A simple matching domain with indifferences and a master list*, Review of Economic Design, pp. 1–25.

[89] Kitty Meeks and Baharak Rastegari (2020), *Solving hard stable matching problems involving groups of similar agents*, Theoretical Computer Science, 844, pp. 171–194.

[90] Patrick Kenekayoro, Promise Mebine, and Bodouowei Godswill Zipamone (2020), *Population based techniques for solving the student project allocation problem*, International Journal of Applied Metaheuristic Computing (IJAMC), 11 (2), pp. 192–207.

[91] Kolos Csaba Ágoston, Péter Biró, and Richárd Szántó (2018), *Stable project allocation under distributional constraints*, Operations Research Perspectives, 5, pp. 59–68.

[92] Ahmed H Abu El-Atta and Mahmoud Ibrahim Moussa, *Student project allocation with preference lists over (student, project) pairs*, in: 2009 Second International Conference on Computer and Electrical Engineering, vol. 1, IEEE, 2009, pp. 375–379.

[93] Marco Chiarandini, Rolf Fagerberg, and Stefano Gualandi (2019), *Handling preferences in student-project allocation*, Annals of Operations Research, 275 (1), pp. 39–78.

[94] Frances Cooper and David Manlove, *A 3/2-Approximation Algorithm for the Student-Project Allocation Problem*, in: 17th International Symposium on Experimental Algorithms (SEA 2018), vol. 103, 2018, 8:1–8:13, DOI: 10.4230/LIPIcs.SEA.2018.8.

[95] Marco Chiarandini, Rolf Fagerberg, and Stefano Gualandi (2019), *Handling preferences in student-project allocation*, Annals of Operations Research, 275 (1), pp. 39–78.

[96] Dimitar Kazakov (2002), *Co-ordination of student-project allocation*, University of York, Department of Computer Science.

[97] Telikepalli Kavitha et al., *Strongly stable matchings in time O (nm) and extension to the hospitals-residents problem*, in: Annual Symposium on Theoretical Aspects of Computer Science, Springer, 2004, pp. 222–233.

[98] Péter Biró et al. (2010), *The college admissions problem with lower and common quotas*, Theoretical Computer Science, 411 (34-36), pp. 3136–3153.

[99] Franz Diebold and Martin Bichler (2017), *Matching with indifferences: A comparison of algorithms in the context of course allocation*, European Journal of Operational Research, 260 (1), pp. 268–282.

[100] Paul R Harper et al. (2005), *A genetic algorithm for the project assignment problem*, Computers & Operations Research, 32 (5), pp. 1255–1265.

[101] Li Pan et al., *Multi-criteria student project allocation: A case study of goal programming formulation with dss implementation*, in: The Eighth International Symposium on Operations Research and Its Applications (ISORA'09), Zhangjiajie, China, 2009, pp. 75–82.

[102] Kazuo Iwama, Shuichi Miyazaki, and Hiroki Yanagisawa (2012), *Improved approximation bounds for the student-project allocation problem with preferences over projects*, Journal of Discrete Algorithms, 13, pp. 59–66.

[103] Biniyam Asmare Kassa (2013), *A linear programming approach for placement of applicants to academic programs*, SpringerPlus, 2 (1), p. 682.

[104] Abigail H Chown, Christopher J Cook, and Nigel B Wilding (2018), *A simulated annealing approach to the student-project allocation problem*, American Journal of Physics, 86 (9), pp. 701–708.

[105] David F Manlove (2015), *The hospitals/residents problem*, Encyclopedia of algorithms, pp. 1–6.

[106] Arif A Anwar and AS Bahaj (2003), *Student project allocation using integer programming*, IEEE Transactions on Education, 46 (3), pp. 359–367.

[107] Jonathan Dye (2001), *A constraint logic programming approach to the stable marriage problem and its application to student-project allocation*, BSc Honours project report, University of York, Department of Computer Science.

[108] M Thorn (2003), *A constraint programming approach to the student-project allocation problem*, BSc Honours project report, University of York, Department of Computer Science.

[109] Hussein M Saber and Jay B Ghosh (2001), *Assigning students to academic majors*, Omega, 29 (6), pp. 513–523.

[110] Robert W. Irving, David F. Manlove, and Gregg O'Malley (2009), *Stable marriage with ties and bounded length preference lists*, Journal of Discrete Algorithms, 7 (1), pp. 213–219.

[111] Viet Hoang Huu et al. (2021), *A max-conflicts based heuristic search for the stable marriage problem with ties and incomplete lists*, Journal of Heuristics, 27 (3), pp. 439–458.

[112] Steven Minton et al. (1992), *Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems*, Artificial intelligence, 58 (1-3), pp. 161–205.

[113] Juwesh Binong, *Solving Student Project Allocation with Preference Through Weights*, in: Proceedings of International Conference on Frontiers in Computing and Systems, Springer, 2021, pp. 423–430.

[114] Fabiyi A Aderanti, RT Amosa, and AA Oluwatobiloba, *Development of student project allocation system using matching algorithm*, in: Int. Conf. Sci. Eng. Environ. Technol, vol. 1, 2016, pp. 153–160.

[115] Murtadha A Gani, Rula A Hamid, et al. (2021), *Optimum Allocation of Graduation Projects: Survey and Proposed Solution*, Journal of Al-Qadisiyah for computer science and mathematics, 13 (1), Page–58.

[116] A Kwanashie et al., *Profile-based optimal matchings in the student/project allocation problem*, in: International Workshop on Combinatorial Algorithms, Springer, 2014, pp. 213–225.

[117] Sofiat Olaosebikan and David Manlove, *An Algorithm for Strong Stability in the Student-Project Allocation Problem with Ties*, in: Conference on Algorithms and Discrete Applied Mathematics, Springer, 2020, pp. 384–399.

[118] Hoang Huu Viet, Van Le Tan, and Thanh Cao Son, *Finding Maximum Stable Matchings for the Student-Project Allocation Problem with Preferences Over Projects*, in: FDSE, Singapore, 2020, pp. 411–422.

PHỤ LỤC A

A.1. Thuật toán Gale-Shapley

Thuật toán Gale-Shapley [1], viết tắt là GS bắt đầu từ một phép ghép rỗng. Tại mỗi bước lặp một người nam m_i độc thân có danh sách xếp hạng chưa rỗng sẽ đề xuất tới người nữ w_j có xếp hạng ưu tiên cao nhất. Nếu w_j chưa được ghép hoặc thích m_i hơn $M(w_j)$ thì w_j sẽ được ghép với m_i, ngược lại w_j từ chối m_i và m_i xoá w_j khỏi danh sách xếp hạng. Thuật toán kết thúc khi tất cả người nam được ghép hoặc những người nam chưa được ghép có danh sách xếp hạng rỗng.

Algorithm A.1: Thuật toán GS

Input: - Thể hiện I, SMTI.
Output: Phép ghép ổn định, M.

function `Main(`I`)`
  for (*mỗi* $m_i \in M$) **do**
    $M(m_i) := \varnothing$;
    $a(m_i) := 1$
  while $\exists m_i \in M | a(m_i) = 1$ **do**
    $m_i :=$ một người nam hoạt động, tức là $a(m_i) = 1$;
    if $\nexists w_k \in W | rank(m_i, w_k) > 0$ **then**
      $a(m_i) := 0$ continue;
    $w_j = argmin(rank(m_i, w_j) > 0)$;
    if $M(w_j) = \varnothing$ **then**
      $M := M \cup \{(m_i, w_j)\}$;
      $a(m_i) := 0$;
    else if $rank(w_j, m_i) < rank(w_j, M(w_j))$ **then**
      $m_k := M(w_j)$;
      $M := M \setminus \{(m_k, w_j)\} \cup \{(m_i, w_j)\}$;
      $rank(m_k, w_j) := 0$;
      $a(m_i) := 0$;
      $a(m_k) := 1$;
    else
      $rank(m_i, w_j) := 0$;
  return M;
end function

A.2. Thuật toán tạo danh sách xếp hạng

Một thể hiện của bài toán SMTI được tạo ngẫu nhiên dựa vào ba tham số (n, p_1, p_2) [44], trong đó n là số người nam và người nữ, p_1 là xác suất tạo danh sách không đầy đủ và p_2 là xác suất tạo danh sách xếp hạng ưu tiên ngang hàng. Thuật toán thực hiện theo các bước như sau:

Bước 1. Tạo ra một danh sách xếp hạng ngẫu nhiên kích thước n cho mỗi người nam và người nữ.

Bước 2. Đối với một người nam m_i và tất cả người nữ w_j trong danh sách xếp hạng, thuật toán tạo một số ngẫu nhiên $0 < p < 1$. Nếu $p < p_1$ thuật toán xóa w_j khỏi danh sách xếp hạng của m_i và xóa m_i khỏi danh sách xếp hạng của w_j.

Bước 3. Nếu bất kỳ người nam hoặc người nữ nào có danh sách xếp hạng rỗng, thuật toán sẽ loại bỏ thể hiện này và quay về Bước 1.

Bước 4. Đối với một người nam m_i, bắt đầu từ người nữ có thứ hạng thứ 2 đến người có thứ hạng ưu tiên thấp nhất trong danh sách xếp hạng, thuật toán tạo ra một số ngẫu nhiên $0 < p < 1$. Nếu $p < p_2$ thì thứ hạng w_j sẽ bằng thứ hạng của người phụ nữ xếp trước nó, nếu không thì thứ hạng w_j tăng lên 1 đơn vị.

Algorithm A.2: Thuật toán tạo danh sách xếp hạng

Input: - Số lượng người nam và người nữ, n.
- Xác suất tạo danh sách rỗng, p_1.
- Xác suất tạo xếp hạng ngang hàng, p_2

Output: Danh sách xếp hạng của người nam và người nữ.

function `Generator`(n, p_1, p_2)
  $f := true$;
  while $f = true$ **do**
    for (*mỗi* $m_i \in \mathcal{M}$) **do**
      Khởi tạo ngẫu nhiên danh sách xếp hạng của m_i;
    for (*mỗi* $w_j \in \mathcal{W}$) **do**
      Khởi tạo ngẫu nhiên danh sách xếp hạng của w_j;
    for (*mỗi* $m_i \in \mathcal{M}$) **do**
      for (*mỗi* $w_j \in \mathcal{W}$) **do**
        if $rand() \leq p_1$ **then**
          $rank(m_i, w_j) := 0$;
          $rank(w_j, m_i) := 0$;
    $f := false$;
    for (*mỗi* $m_i \in \mathcal{M}$) **do**
      if $\nexists w_j \in \mathcal{W} | rank(m_i, w_j) > 0$ **then**
        $f := true$;
        continue;
    for (*mỗi* $w_j \in \mathcal{W}$) **do**
      if $\nexists m_i \in \mathcal{M} | rank(w_j, m_i) > 0$ **then**
        $f := true$;
        continue;
    $rank^{\dagger}(m_i, w_j) := rank(m_i, w_j), \forall m_i \in \mathcal{M}, w_j \in \mathcal{W}$;
    $w_k = argmin(rank(m_i, w_k) > 0)$;
    $r^{th} := 1$;
    while $\nexists w_j \in \mathcal{W} | rank^{\dagger}(m_i, w_j) > 0$ **do**
      $w_j := argmin(rank^{\dagger}(m_i, w_j) > 0)$;
      if $rand() > p_2$ *và* $w_j \neq w_k$ **then**
        $r^{th} := r^{th} + 1$;
      $rank(m_i, w_j) := r^{th}$;
      $rank^{\dagger}(m_i, w_j) := 0$;
    $rank^{\dagger}(w_j, m_i) := rank(w_j, m_i), \forall w_j \in \mathcal{W}, m_i \in \mathcal{M}$;
    $m_k := argmin(rank(w_j, m_k) > 0)$;
    $r^{th} := 1$;
    while $\nexists m_i \in \mathcal{W} | rank^{\dagger}(w_j, m_i) > 0$ **do**
      $m_i := argmin(rank^{\dagger}(w_j, m_i) > 0)$;
      if $rand() > p_2$ *và* $m_i \neq m_k$ **then**
        $r^{th} := r^{th} + 1$;
      $rank(w_j, m_i) := r^{th}$;
      $rank^{\dagger}(w_j, m_i) := 0$;
  return *Danh sách xếp hạng của người nam và người nữ*;
end function