Thuật toán rút gọn sơ đồ quan hệ môn học theo chuẩn CDIO.

Trần Tiến Dũnga\*, Nguyễn Tuấn Minhb, Đào Xuân Trườngb

aBộ môn Công nghệ Phần mềm, Khoa Công nghệ Thông tin, Đại học Công nghiệp Hà Nội

bLớp KTPMCLC-K9, Khoa Công nghệ thông tin, Đại học Công nghiệp Hà Nội

Email: trantd.vn@gmail.com

Tóm tắt

Hoạt động thiết kế một chương trình đào tạo theo tiếp cận CDIO tạo ra một sơ đồ quan hệ học phần − một đồ thị có hướng bao gồm các đỉnh và các cạnh tương ứng biểu diễn các học phần và các quan hệ thứ tự học phần. Nguyên thủy, đồ thị này thường chứa nhiều cạnh đan xen nhau, khó quan sát các quan hệ học phần, không thuận tiện cho việc xuất bản tài liệu cũng như đăng ký học phần của sinh viên. Chúng tôi đã phát triển một thuật toán mới rút gọn một đồ thị có hướng, và áp dụng nó vào rút gọn sơ đồ quan hệ học phần của chương trình Đại học Kỹ thuật phần mềm ở ĐH Công nghiệp Hà Nội. Kết quả nghiên cứu này góp phần chuyên nghiệp hóa hoạt động thiết kế các chương trình đào tạo trong trường đại học.

*Từ khóa: CDIO, quan hệ học phần, rút gọn đồ thị.*

1. Giới thiệu

CDIO là một khung giáo dục kỹ thuật nhấn mạnh các nguyên tắc cơ bản trong nhận thức (*Conceive)*, thiết kế (*Design)*, triển khai (*Implement)*, và vận hành (*Operate*) các hệ thống và sản phẩm thực tế [[1](#_ENREF_1), [2](#_ENREF_2)]. Đây cũng được coi là 04 năng lực cơ bản một chương trình đào tạo kỹ thuật [[3](#_ENREF_3), [4](#_ENREF_4)]. Đạt được các năng lực CDIO này, sinh viên kỹ thuật sẽ có được các kỹ năng cứng và mềm cần thiết khi ra trường, đáp ứng được yêu cầu của xã hội cũng như bắt nhịp được với những thay đổi của môi trường, thậm chí là có thể dẫn dắt sự thay đổi đó [[5](#_ENREF_5)]. Hiện tại, ĐHCNHN đang thiết kế lại các chương trình đào tạo theo tiếp cận CDIO[[6](#_ENREF_6)]. Hoạt động thiết kế một chương trình đào tạo theo tiếp cận CDIO tạo ra một sơ đồ quan hệ học phần − một đồ thị có hướng bao gồm các đỉnh và các cạnh tương ứng biểu diễn các học phần và các quan hệ thứ tự học phần. Một đồ thị *G*(*V*, *E*) được gọi là thành phần liên thông yếu nếu thỏa mãn điều kiện với mỗi cặp đỉnh (*u,v*) *∈ {V×V}*, tồn tại ít nhất một đường đi từ *u* đến *v* hoặc *v* đến *u*. *G* được gọi là thành phần liên thông mạnh nếu tồn tại đồng thời đường đi từ *u* đến *v* và *v* đến *u*[[7](#_ENREF_7)]. Nếu *G* là một đồ thị liên thông yếu, một đồ thị con *H*(*V,E’*) *⊆* *G* được gọi là một rút gọn hay phân nhánh tối ưu của *G* nếu: *1)* Tồn tại một đỉnh sao cho từ đó có một đường đi duy nhất đến các đỉnh còn lại. *2)* Tổng trọng số các cạnh của *H* là lớn nhất [[8](#_ENREF_8), [9](#_ENREF_9)]. Nguyên thủy, một đồ thị quan hệ học phần theo tiếp cận CDIO thường không phân nhánh tối ưu và chứa nhiều cạnh đan xen nhau, khó quan sát các quan hệ học phần, không thuận tiện cho việc xuất bản tài liệu cũng như đăng ký học phần của sinh viên. Bài toán tìm các đồ thị phân nhánh tối ưu của đồ thị quan hệ học phần theo tiếp cận CDIO có thể giải quyết bằng các thuật toán đồ thị. Tuy nhiên, nhiều thuật toán toán tìm phân nhánh tối ưu hiện nay, điển hình Edmond và Giles [[10](#_ENREF_10), [11](#_ENREF_11)], chỉ cho phép tìm các đồ thị phân nhanh tối ưu của từ một đồ thị liên thông yếu, song tính chất liên thông yếu thường không có được trong các đồ thị quan hệ học phần.

Chúng tôi đã phát triển một thuật toán mới bằng việc mở rộng thuật toán thuật toán tìm phân nhánh tối ưu của Edmond và Giles [[10](#_ENREF_10), [11](#_ENREF_11)] để áp dụng vào rút gọn sơ đồ quan hệ học phần của chương trình Đại học Kỹ thuật phần mềm ở ĐHCNHN và đã trả về 1000 phương án rút gọn khác nhau. Chúng tôi cũng đã tìm ra các cạnh luôn có mặt trong các 1000 đồ thị rút gọn này. Chúng được đánh giá là các quan hệ học phần thiết yếu trong chương trình đào tạo. Kết quả nghiên cứu này góp phần làm chuyên nghiệp hóa hoạt động thiết kế các chương trình đào tạo trong trường đại học.

|  |
| --- |
|  |
| **Hình 1. Một ví dụ về đồ thị**. Các ô tròn với các đỉnh và mũi tên tương ứng thể hiện các nút (node) với các cạnh và trọng số của cạnh đó. (A) Một đồ thị liên thông yếu nếu giữa hai đỉnh bất kì luôn có ít nhất một đường đi từ một đỉnh đến đỉnh còn lại. (B) Phân nhánh tối ưu là một đồ thị con thỏa mãn 2 điều kiện: 1) Tồn tại duy nhất một đỉnh có đường đi đến tất cả các cạnh còn lại (đỉnh 10), 2) Tổng trọng số trên các cạnh là lớn nhất. |

2. Phương pháp và tư liệu thuật toán

2.1. Tìm đồ thị liên thông yếu

Trước khi thực hiện thuật toán rút gọn, mô hình quan hệ học phần cần được đảm bảo tính liên thông yếu. Theo định nghĩa đồ thị liên thông yếu thì giữa 2 đỉnh bất kỳ cần có ít nhất một đường đi từ một đỉnh đến đỉnh còn lại. Vì vậy nên, chúng tôi đã áp dụng thuật toán Floyd-Warshall [[12-14](#_ENREF_12)] với mục đích tìm đường đi ngắn nhất giữa tất cả các cặp đỉnh trong đồ thị có trọng số. Cho một đồ thị *G(V,E)*, thuật toán này sẽ tìm đường đi ngắn nhất giữa 2 cặp đỉnh bất kì *(u,v) ∈ E* và lưu vào mảng 2 chiều *adj* với *adj[u][v]* là đường đi ngắn nhất từ đỉnh *u* đến đỉnh *v*[[15](#_ENREF_15)]. Không khó để nhận ra rằng, nếu đỉnh *u* có đường đi đến đỉnh *v* thì sẽ phải tồn tại một giá trị là độ dài đường đi ngắn nhất từ *u* đến *v*, hay nói cách khác là .

*B1:* thì

*B2:*.

*B3:.*

*B4*: thì lặp lại B1.

Thông tin về các *input node* (đỉnh mà không có cạnh đi đến) bị loại trong *B1* sẽ được lưu vào tập *V’* và *E’* để thêm lại sau này. Thuật toán chỉ kết thúc khi mọi đỉnh con lại đều có đường đi đến các đỉnh khác (được kiểm tra ở *B4*) nên sau khi chạy thuật toán thì đồ thị kết quả đã thỏa mãn được tính chất liên thông yếu.

2.2. Thuật toán tìm phân nhánh tối ưu

2.2.1. Các bước thực hiện

Phần này sẽ nêu lên các bước của thuật toán tìm phân nhánh tối ưu của Edmond và Giles[[10](#_ENREF_10), [11](#_ENREF_11)], kèm theo cách cài đặt hiệu quả nhất theo Tarjan và Ziegler[[16](#_ENREF_16), [17](#_ENREF_17)]. Với dữ liệu đầu vào là một đồ thị liên thông yếu *G*(*V,E*)*,* thuật toán sẽ cho ra một đồ thị con *G’(V,H)* là phân nhánh tối ưu của đồ thị ban đầu, gọi *S* là thành phần liên thông mạnh của *G’* mà có một cạnh (*u,v*) sao cho *v S* và(*u,v*), *W(u)* là thành phần liên thông yếu của *G’* chứa đỉnh *u*.

Bước đầu:

*F1:*

*F1*: Chọn một đỉnh và cạnh .

*F3*:.

Các bước chung:

*G1:* Tìm

*G2:* Nếu thì .

*G3:* Nếu thì , quay lại *G1*.

*G4:* Tìm chuỗi *S*1, (*x1, y1*), *S*2, (*x2, y2*),….,*S*k,(*xk, yk*) sao cho mỗi *S*i là một thành phần liên thông mạnh của *G’*, (*xi, yi*) ∈ *H*, *y*i *∈ S*i, *x*i *∈ S*i+1 , *Sk* = *S*, (*xk, yk*) = (*u*,*v*), *xk* ∈ *S*1.

*G5*: Tìm (*xj, yj*)*c*(*xj, yj*) = (*xi, yi*)).

*G6*: *c*(*x*, *y*) = *c*(*x*, *y*) – *c*(*xi, yi*) + *c*(*x*j, *y*j).

*G7*: .

*G8:* Nếu thì lặp lại *G1*.

Bước cuối:

*R1:* Tìm thành phần liên thông mạnh *R* của *G’* chứa nhiều hơn một đỉnh.

*R2:* Tìm chuỗi *S*1, (*x*1, *y*1)….*S*k, (*x*k, *y*k) trong bước *G5*, sao cho (*x*k, *y*k) được thêm vào từ *H* để hình thành nên *R*.

*R3:* Tìm cạnh (*xj, yj*)*c*(*xj, yj*) = (*cxi, yi*)).

*R4:* *H* = *H* – {(*xj, yj*)}

Sau khi chạy thuật toán trên thì các cạnh còn lại trong *H* chính là phân nhánh tối ưu cần tìm, việc còn lại cần làm sau đó là thêm các cạnh và đỉnh từ tập *V’* và *E’* đã tìm được ở *B4* vào *G’.*

2.2.2. Chứng minh tính đúng đắn của thuật toán

Giả sử thuật toán này được áp dụng cho đồ thị *G(V,E)*. Các bổ đề sau được suy ra từ kết quả của Edmonds và Karp[[10](#_ENREF_10), [18](#_ENREF_18)], cho rằng *G*’(*V*,*H*).

*Bổ đề 1:* Mỗi thành phần liên thông mạnh *S* của *G’* có nhiều nhất một cạnh (*u*,*v*) *∈ H* với *v* *∈ S*, *u ∈ V – S*. Mỗi thành phần liên thông yếu *W* của *G’* chứa đúng một thành phần gốc.

*Bổ đề 2:* Giả sử *S* là một thành phần gốc bất kỳ của *G’* và *W* là thành phần liên thông yếu chứa *S* và *v* *∈ S*, thì với mọi *w ∈ W* có duy nhất một đường đi thuộc *G’* từ *v* đến *w*.

Các bổ đề này có thể dễ dàng suy ra từ quá trình thêm các cạnh vào *H*. Khi thuật toán hoàn thành, *H* sẽ chứa một phân nhánh tối ưu, được suy ra từ bổ đề 2. Tất cả những gì chúng ta cần xác định là các đỉnh *v* riêng biệt là thành phần gốc của mỗi thành phần liên thông yếu trong *G’*. Các đỉnh sẽ giúp ta xác định phân nhánh chứa các lối đi đơn giản suy ra từ bổ đề 2. Nếu ta sử dụng bước cuối của thuật toán để tìm ra các đỉnh đó, thì phân nhánh kết quả sau đó sẽ là tối ưu.

2.3. Dữ liệu quan hệ môn học

Chúng tôi sưu tầm sơ đồ quan hệ học phần nguyên thủy của chuyên ngành Kỹ thuật phần mềm ở Bộ môn Công nghệ Phần mềm, Khoa Công nghệ Thông tin từ kết quả tổng hợp bộ dữ liệu phiếu khảo sát mức độ liên hệ giữa 58 học phần từ các giảng viên Khoa CNTT và chuyên gia đào tạo. Sau khi loại các học phần độc lập, kết quả là một đồ thị liên thông gồm 51 đỉnh và 217 cạnh. Các cạnh trên đồ thị cho người xem thấy được các học phần đầu vào và đầu ra của mỗi học phần từ kỳ 1 đến kỳ 8 trong suốt 4 năm đại học. Sau khi trực quan hóa đồ thị này bằng phần mềm NodeXL (xin xem NodeXL trong [[19](#_ENREF_19), [20](#_ENREF_20)]), sơ đồ quan hệ học phần có hình dạng như Hình 2.

|  |
| --- |
|  |
| **Hình 2. Sơ đồ quan hệ học phần nguyên thủy của chuyên ngành Kỹ thuật phần mềm**. Đây là một đồ thị có nhiều cạnh đan nhau, khó nhìn gồm 51 đỉnh và 217 cạnh tương ứng là các học phần và các quan hệ thứ tự học phần trong đó các học phần cùng học kỳ được đặt cùng một cột. |

**3. Kết quả và thảo luận**

**3.1. Thuật toán rút gọn sơ đồ quan hệ**

Sơ đồ quan hệ học phần nguyên thủy chứa nhiều mối quan hệ đan xen nhau khiến cho người xem cảm thấy bối rối, gần như không thể quan sát bằng mắt được. Chúng tôi đã áp dụng thuật toán tìm phân nhánh tối ưu để rút gọn số lượng các quan hệ sao cho từ các học phần ở học kỳ 1 có thể duyệt qua tất cả các học phần khác với một số lượng quan hệ ít nhất. Nói cách khác, thuật toán cho phép tìm một đồ thị con chứa một đỉnh có đường đi đến tất cả các đỉnh còn lại và tổng trọng số trên tất cả các cạnh là lớn nhất. Tuy nhiên, việc làm như vậy sớm bộc lộ ra 2 nhược điểm: *1)* Đồ thị ban đầu không đảm bảo tính liên thông yếu khiến cho đồ thị kết quả bị mất tính liên thông. *2)* Việc cho tất cả các cạnh ban đầu có trọng số bằng 1 dẫn đến hệ quả tất yếu là không còn tồn tại chỉ một đồ thị con có tổng trọng số lớn nhất nữa, điều này cũng có nghĩa là không còn phương án tối ưu nhất theo như mô tả thuật toán ban đầu.

Với hạn chế này, chúng tôi đề xuất một thuật toán mới được mở rộng từ thuật toán tìm phân nhánh tối ưu gốc. Một vài thay đổi có thể kể đến là: (1) Loại bỏ các *input node* cho đến khi đồ thị được đảm bảo tính liên thông yếu (kiểm tra bằng thuật toán Floyd-Warshall) và thêm lại sau khi chạy thuật toán cải tiến, (2) Thay đổi thứ tự duyệt cạnh của thuật toán tìm phân nhánh tối ưu gốc (chọn cạnh ngẫu nhiên thay vì luôn chọn cạnh có trọng số lớn nhất). Bằng cách này, chúng tôi đã tìm ra được rất nhiều kết quả rút gọn thỏa mãn yêu cầu thay vì chỉ 1 như thuật toán gốc.

|  |
| --- |
|  |
| Hình 3. Kết quả rút gọn mô hình quan hệ môn học sau khi chạy thuật toán tìm phân nhánh tối ưu gốc. Mô hình này chưa đảm bảo tính liên thông (ví dụ như môn ‘Toán giải tích’ và ‘Phương pháp tính trong lập trình’ bị tách biệt với phần còn lại), ngoài ra một vài môn học không xuất hiện do đồ thị ban đầu chưa đảm bảo tính liên thông yếu nên sau khi rút gọn không còn cạnh chứa môn đó nữa. |

|  |
| --- |
|  |
| Hình 4. Thống kê số lượng cạnh chung xét theo trên số lượng phương án tìm được. Số lượng phương án càng tăng thì số lượng cạnh chung giữa các phương án càng giảm, tuy nhiên con số đó chỉ dừng lại ở 15 (xem phụ lục bảng S2). |

|  |
| --- |
|  |
| Hình 5. Một trong số các phương án rút gọn mô hình quan hệ môn học sau khi chạy thuật toán cải tiến. Đồ thị đã được đảm bảo tính liên thông và bao gồm 9 input node (xem thêm phần phụ lục Bảng S1) được tô đậm bằng màu xanh cùng với 15 cạnh chung của 1000 phương án được tô tậm bằng màu đỏ. |

3.2. Phương pháp tìm các quan hệ môn học thiết yếu

Sau khi chạy thuật toán cải tiến, đã có rất nhiều kết quả được tìm ra, tất cả các phương án đều đơn giản hóa mô hình quan hệ ban đầu đến mức tổi thiểu (từ 133 cạnh xuống chỉ còn 47). Tuy nhiên đề tài này sẽ nghiên cứu và tìm hiểu điểm chung giữa 1000 kết quả đầu tiên. Đã có 15 trên tổng số 47 cạnh chung được tìm thấy trong tất cả 1000 phương án kể trên, các cạnh đó sẽ được thể hiện trên 1 phương án tại Hình 4. Từ đó có thể kết luận rằng 15 cạnh đó là những cạnh thiết yếu không thể thiếu trong mọi trường hợp. Đề tài này cũng xác định được rằng trong số đó có 8 cạnh là các cạnh được thêm của các *input node* và 7 cạnh còn lại là các cạnh đầu vào duy nhất của các *node*. Danh sách các *input node* đã được thống kê trong Bảng S1 phần phụ lục.

4. Kết luận

Đề tài này đã tập trung vào việc rút gọn mô hình quan hệ môn học của các môn trong khoa Công nghệ Thông tin, Đại học Công nghiệp Hà Nội nhằm mục đích đơn giản hóa, hỗ trợ cho việc xây dựng chương trình đào tạo mới theo tiếp cận CDIO. Bằng cách xây dựng và cải tiến một thuật toán mới, người thực hiện đề tài đã có thể vượt qua những hạn chế và thiếu sót của thuật toán tìm phân nhánh tối ưu gốc và thỏa mãn các yêu cầu cần đạt được của đề tài. Việc rút gọn đã cho ra một vài thông tin thú vị về chương trình đào tạo, cụ thể như các môn học gần như bắt buộc phải có ban đầu trước khi học các môn khác (*input node)*, hay là những quan hệ giữa các môn học mà không thể thay đổi trong mọi phương án. Đây cũng là một trong những nghiên cứu đầu tiên tại Việt Nam về việc áp dụng thuật toán vào các mô hình hoạt động và đơn giản hóa nó. Kết quả của đề tài này có thể sử dụng để tham khảo và phát triển tiếp nhằm xác định phương án thực sự tối ưu nhất trong số các phương án đã được tìm thấy sau khi chạy thuật toán.

Lời cảm ơn. Xin trân trọng cảm ơn thầy Trần Tiến Dũng, Khoa CNTT, Đại học Công nghiệp Hà Nội đã hướng dẫn thực hiện đề tài này.

5. Tài liệu tham khảo

1. Karl-Frederik Berggren, D.B., Edward F. Crawley, Ingemar Ingemarsson, William T.G. Litant, Johan Malmqvist & Sören Östlund, *CDIO: An international initiative for reforming engineering education.* World Transactions on Engineering and Technology Education, 2003. **2**(1): p. 49-52.

2. Wang, Z. and C. Liu, *Exploration of CDIO engineering education mode for BEEE.* 2013 8th International Conference on Computer Science & Education, 2013. **13**(2): p. 1127-1130.

3. Campbell, D.A., et al., *An extended CDIO syllabus framework with preparatory engineering proficiencies*, in *5th International CDIO Conference*2009, CDIO Initiative: Singapore Polytechnic, Singapore.

4. Edström, K., *The role of CDIO in engineering education research: Combining usefulness and scholarliness.* European Journal of Engineering Education, 2017: p. 1-15.

5. Edward, F.C. and B. Doris, *Program Evaluation Aligned With The Cdio Standards*, 2005, ASEE Conferences: Portland, Oregon. p. 1-18.

6. *Phương pháp CDIO trong giáo dục các ngành kỹ thuật đại học Công nghiệp Hà Nội.* haui.edu.vn, 2011.

7. Duckworth, W. and B. Mans, *Randomised Algorithms for Finding Small Weakly-Connected Dominating Sets of Regular Graphs*, in *Algorithms and Complexity: 5th Italian Conference, CIAC 2003, Rome, Italy, May 28–30, 2003. Proceedings*, R. Petreschi, G. Persiano, and R. Silvestri, Editors. 2003, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 83-95.

8. Dunbar, J.E., et al., *On weakly connected domination in graphs.* Discrete Mathematics, 1997. **167-168**(Supplement C): p. 261-269.

9. Kurucz, A., *Bimodal Logics with a "Weakly Connected" Component without the Finite Model Property.* Notre Dame J. Formal Logic, 2017. **58**(2): p. 287-299.

10. Edmonds, J., *Optimum Branchings.* JOURNAL OF RESEARCH of the National Bureau of Standards 1967. **71B**(4): p. 233-240.

11. Giles, R., *Optimum matching forests I: Special weights.* Mathematical Programming, 1982. **22**(1): p. 1-11.

12. Aini, A. and A. Salehipour, *Speeding up the Floyd–Warshall algorithm for the cycled shortest path problem.* Applied Mathematics Letters, 2012. **25**(1): p. 1-5.

13. Akanksha Singh, P.K.M., *Performance Analysis of Floyd Warshall Algorithm vs Rectangular Algorithm.* nternational Journal of Computer Applications 2014. **107**(16): p. 23-27.

14. Jian, M., L. Ke-ping, and L.y. Zhang. *A Parallel Floyd-Warshall algorithm based on TBB*. in *2010 2nd IEEE International Conference on Information Management and Engineering*. 2010.

15. Kairanbay Magzhan, H.M.J., *A Review And Evaluations Of Shortest Path Algorithms.* INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH, 2013. **2**(6): p. 99-104.

16. Tarjan, R.E., *Finding optimum branchings.* Networks, 1977. **7**(1): p. 25-35.

17. Ziegler, V., *Approximating optimum branchings in linear time.* Information Processing Letters, 2009. **109**(3): p. 175-178.

18. Karp, R.M., *A simple derivation of Edmonds' algorithm for optimum branchings.* Networks, 1971. **1**(3): p. 265-272.

19. Smith, M.A. *NodeXL: Simple network analysis for social media*. in *2013 International Conference on Collaboration Technologies and Systems (CTS)*. 2013.

20. Smith, M.A., et al., *Analyzing (social media) networks with NodeXL*, in *Proceedings of the fourth international conference on Communities and technologies*2009, ACM: University Park, PA, USA. p. 255-264.

6. Phụ lục

|  |  |
| --- | --- |
| **STT** | **Tên môn học** |
| 1 | Toán giải tích |
| 2 | Đại số tuyến tính |
| 3 | Xác suất thống kê |
| 4 | Kỹ thuật số |
| 5 | Giới thiệu ngành |
| 6 | Toán rời rạc |
| 7 | Lập trình căn bản |
| 8 | Kỹ thuật lập trình |
| 9 | Kỹ năng giao tiếp |

**Bảng S1. Thông kê các *input node* của tất cả các phương án**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Từ** | **Đến** | **Chứa *input node*** |
| 1 | Toán giải tích | Phương pháp tính trong lập trình | Có |
| 2 | Xác suất thống kê | An toàn và bảo mật thông tin | Có |
| 3 | Kỹ thuật số | Cơ sở lập trình nhúng | Có |
| 4 | Giới thiệu ngành | Nhập môn Công nghệ phần mềm | Có |
| 5 | Toán rời rạc | Kiến trúc máy tính | Có |
| 6 | Toán rời rạc | Cơ sở dữ liệu | Có |
| 7 | Cơ sở dữ liệu | Hệ quản trị cơ sở dữ liệu |  |
| 8 | Kỹ thuật lập trình | Đồ họa máy tính | Có |
| 9 | Thiết kế phần mềm | Công nghệ đa phương tiện |  |
| 10 | Kiến trúc máy tính | Nguyên lý hệ điều hành |  |
| 11 | Nguyên lý hệ điều hành | Mạng máy tính |  |
| 12 | Thiết kế Web | Tích hợp hệ thống phần mềm |  |
| 13 | Kỹ năng giao tiếp | Quản lý dự án công nghệ thông tin | Có |
| 14 | Thực tập chuyên ngành | Thực tập doanh nghiệp |  |
| 15 | Thực tập doanh nghiệp | Đồ án TN |  |

**Bảng S2. Thống kê 15 cạnh chung của 1000 phương án sau khi chạy thuật toán tìm phân nhánh tối ưu cải tiến.**