BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC CÔNG NGHỆ TP.HCM

TRÍ TUỆ NHÂN TẠO

Biên Soạn:

PGS.TS. Võ Đình Bảy

TS. Vũ Thanh Hiền

TS. Huỳnh Quốc Bảo

www.hutech.edu.vn

TRÍ TUỆ NHÂN TẠO

Các ý kiến đóng góp về tài liệu học tập này, xin gửi về e-mail của ban biên tập:

tailieuhoctap@hutech.edu.vn

MỤC LỤC

MỤC LỤC I
HƯỚNG DẪN III
BÀI 1: TỔNG QUAN VỀ TRÍ TUỆ NHÂN TẠO 1
1.1 MỤC TIÊU MÔN HỌC 1
1.2 LỊCH SỬ PHÁT TRIỂN CỦA TRÍ TUỆ NHÂN TẠO 3
1.3 QUÁ TRÌNH PHÁT TRIỂN CỦA TTNT 4
1.4 ỨNG DỤNG CỦA TTNT 8
BÀI 2: TÔ MÀU ĐỒ THỊ 11
2.1 THUẬT GIẢI TÔ MÀU TỐI ƯU 11
2.1.1 Đồ thị 11
2.1.2 Phân loại đồ thị 11
2.1.3 Một số thuật ngữ cơ bản 12
2.1.4 Tô màu đồ thị 13
2.1.5 Tô màu tối ưu 16
2.2 THUẬT GIẢI THAM LAM 20
2.3 THUẬT GIẢI LAI GHÉP 27
2.3.1 Cơ sở lý thuyết 27
2.3.2 Cấu trúc thuật toán di truyền tổng quát 28
2.3.3 Các công thức của thuật giải di truyền 30
2.3.4 Bài toán minh họa 30
BÀI 3: CÁC PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI 33
3.1 TÌM KIẾM CƠ BẢN 33
3.1.1 Phát biểu bài toán 33
3.1.2 Không gian tìm kiếm 35
3.1.3 Các chiến lược tìm kiếm 38
3.1.4 Các giải thuật cơ bản 38
3.2 TÌM KIẾM THEO KINH NGHIỆM (HEURISTIC) 42
3.2.1 Hàm Heuristic là gì? 42
3.2.2 Tìm kiếm leo đồi (Hill climbing – Pearl 1984) 43
3.2.3 Tìm kiếm tốt nhất đầu tiên (Best – first – search) 44
3.3 TÌM KIẾM VÀ TỐI ƯU HÓA 47
3.4 BÀI TOÁN PHÂN CÔNG CÔNG VIỆC 65
BÀI 4: BIỂU DIỄN VÀ XỬ LÝ TRI THỨC 67
4.1 LOGIC MỆNH ĐỀ 67
4.1.1 Các qui luật logic cơ bản 68
4.1.2 Dạng chuẩn hội (Conjunctive normal form – CNF) 69
4.1.3 Câu dạng Hom 70

4.1.4 Luật Suy Diễn 70
4.1.5 Các giải thuật liên quan đến Logic mệnh đề 71
4.2 BIỂU DIỄN TRI THỨC 72
4.2.1 Định nghĩa 72
4.2.2 Phân loại 72
4.2.3 Biểu diễn tri thức 73
4.2.4 Logic vị từ 74
4.3 THUẬT GIẢI VƯƠNG HẠO 76
4.4 HỢP GIẢI 77
4.4.1 Qui trình INDO 78
4.4.2 Qui tắc 78
4.4.3 4.4.3. Thuật toán 79
4.4.4 Hợp giải tuyến tính 80
4.5 MẠNG NGỮ NGHĨA 81
4.5.1 Khái niệm 81
4.5.2 Ưu điểm và nhược điểm của mạng ngữ nghĩa 82
4.5.3 Cơ chế suy diễn 83
4.5.4 Xử lý ngôn ngữ tự nhiên 90
BÀI 5: HỌC MÁY 92
5.1 GIỚI THIỆU HỌC MÁY 92
5.2 QUY TRÌNH HỌC MÁY 97
5.3 PHÂN LỚP 99
5.3.1 Thuật toán ID3 101
5.3.2 Thuật toán C4.5 104
5.3.3 Thuật giải Học Quy nạp ILA (ILA - Inductive Learning Algorithm) 105
5.3.4 Bộ phân lớp Bayes 110
5.4 GOM CỤM DỮ LIỆU 115
5.4.1 Thuật toán k – means 119
5.4.2 Thuật toán k – medoids 123
TÀI LIỆU THAM KHẢO 126

HƯỚNG DẪN

MÔ TẢ MÔN HỌC

Giáo trình "Trí tuệ nhân tạo" được soạn nhằm cung cấp các kiến thức nền tảng về trí tuệ nhân tạo. Cụ thể là cách giải quyết các bài toán tô màu đồ thị, các phương pháp tìm kiếm lời giải bài toán, các phương pháp biểu diễn và xử lý tri thức và bài toán phân lớp.

NỘI DUNG MÔN HỌC

- **BÀI 1: TỔNG QUAN VỀ TRÍ TUỆ NHÂN TẠO (TTNT)**

Bài này cung cấp lịch sử hình hành và phát triển của trí tuệ nhân tạo. Các khái niệm cơ bản, định nghĩa. Các lĩnh vực nghiên cứu và ứng dụng cơ bản.

- **BÀI 2: TÔ MÀU ĐỒ THỊ**

Bài này cung cấp giải thuật tối ưu và tham lam để giải bài toán tô màu.

- **BÀI 3: CÁC PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI**

Bài này cung cấp các phương pháp tìm kiếm trong không gian trang thái bao gồm các kỹ thuật tìm kiếm mù và heuristic.

- **BÀI 4: BIỂU DIỄN VÀ XỬ LÝ TRI THỨC**

Bài này cung cấp các khái niệm liên quan đến xứ lý và biểu diễn tri thức (mệnh đề, logic vị từ, ...). Các thuật toán liên quan Vương Hạo và Robinson

- **BÀI 5: HỌC MÁY - PHƯƠNG PHÁP HỌC CÓ GIÁM SÁT**

Bài này cung cấp các khái niệm liên quan đến phân lớp và dự đoán. Các thuật giải phân lớp cơ bản

- **BÀI 6: HỌC MÁY - PHƯƠNG PHÁP HỌC KHÔNG GIÁM SÁT**

Bài này cung cấp các khái niệm liên quan đến phân cụm. Các thuật giải về học bán giám sát và học tăng cường.

KIẾN THỨC TIỀN ĐỀ

Môn học đòi hỏi sinh viên có kiến thức nền tảng về cấu trúc dữ liệu và lý thuyết đồ thị.

YÊU CẦU MÔN HỌC

Sinh viên xem trước tài liệu và làm các bài thực hành đầy đủ. Để học tốt môn này, sinh viên cần xem qua mỗi bài giảng để nắm lý thuyết và áp dụng kiến thức vào các bài thực tập.

CÁCH TIẾP CẬN NỘI DUNG MÔN HỌC

Để học tốt môn này, người học cần ôn tập các bài đã học, trả lời các câu hỏi và làm đầy đủ bài tập; đọc trước bài mới và tìm thêm các thông tin liên quan đến bài học. Đối với mỗi bài học, người học đọc trước mục tiêu và tóm tắt bài học, sau đó đọc nội dung bài học. Kết thúc mỗi ý của bài học, người đọc trả lời câu hỏi ôn tập và kết thúc toàn bộ bài học, người đọc làm các bài tập. Tìm đọc thêm các tài liệu khác liên quan đến bài học và làm thêm bài tập.

PHƯƠNG PHÁP ĐÁNH GIÁ MÔN HỌC

- Điểm quá trình: 50%. Hình thức và nội dung do giảng viên quyết định, phù hợp với quy chế đào tạo và tình hình thực tế tại nơi tổ chức học tập.
- Điểm thi: 50%. Hình thức dựa vào chấm vấn đáp đồ án môn học.

BÀI 1: TỔNG QUAN VỀ TRÍ TUỆ NHÂN TẠO

Học xong bài này sinh viên có thể nắm được:

- *Ý nghĩa, mục đích môn học, lịch sử hình thành và phát triển của Trí tuệ nhân tạo (AI).*
- *Các khái niệm cơ bản, định nghĩa của AI.*
- *Các lĩnh vực nghiên cứu và ứng dụng cơ bản.*
- *Những vấn đề chưa được giải quyết trong AI.*

1.1 MỤC TIÊU MÔN HỌC

Trí tuệ nhân tạo (Artificial Intelligence - AI) là một ngành thuộc lĩnh vực khoa học máy tính đã được rất nhiều các nhà nghiên cứu quan tâm. Trí tuệ nhân tạo cung cấp các phương pháp luận về cách giải quyết vấn đề cho các bài toán tìm kiếm, các phương pháp biểu diễn và xử lý tri thức và bài toán phân lớp.

Mục tiêu của Trí tuệ nhân tạo nhằm giải thích các hoạt động thông minh, tìm hiểu cơ chế thông minh của con người, xây dựng cơ chế hiện thực sự thông minh và áp dụng các hiểu biết này để có thể tự động hóa các hành vi thông minh vào các máy móc phục vụ cho con người.

Về mặt kỹ thuật, Trí tuệ nhân tạo phát triển ra các máy thông minh để giải quyết vấn đề thực tế dùng các kỹ thuật Trí tuệ nhân tạo. Về mặt khoa học, Trí tuệ nhân tạo phát triển các khái niệm và thuật ngữ để hiểu được các hành xử thông minh của sinh vật nói chung.

Để có thể học tốt môn Trí tuệ nhân tạo, yêu cầu người học cần phải nắm vững các nội dung liên quan đến các nguyên tắc lập trình, cấu trúc dữ liệu và giải thuật. Có kiến thức về ngôn ngữ lập trình C/C ++ và / hoặc Java.

Trí tuệ nhân tạo là gì?

Hiện nay có khá nhiều định nghĩa khác nhau về Trí tuệ nhân tạo, tuy nhiên định nghĩa thông dụng nhất về Trí tuệ nhân tạo như sau:

Trí tuệ nhân tạo là trí tuệ được biểu diễn bởi bất kỳ một hệ thống nhân tạo nào. Hệ thống đó sẽ mô phỏng các quá trình hoạt động trí tuệ của con người, bao gồm quá trình học tập, lập luận và tự sửa lỗi.

Hoặc

Trí tuệ nhân tạo là khoa học nghiên cứu các hành vi thông minh nhằm giải quyết các vấn đề được đặt ra đối với các chương trình máy tính.

Từ khi chiếc máy tính điện tử đầu tiên ra đời, các nhà khoa học máy tính đã hướng đến phát hiển hệ thống máy tính sao cho nó có khả năng thông minh như loài người. Trong một số lĩnh vực, máy tính có thể thực hiện tốt hơn hoặc tương đương con người, đặc biệt là các hệ thống thông minh.

Có nhiều cách tiếp cận để xây dựng một hệ thống thông minh, chẳng hạn là nghiên cứu cách bộ não người sản sinh ra trí thông minh của loài người như thế nào rồi ta bắt chước nguyên lý đó, nhưng cũng có những cách khác sử dụng nguyên lý hoàn toàn khác với cách sản sinh ra trí thông minh của loài người mà vẫn làm ra cái máy thông minh như hoặc hơn người; cũng giống như máy bay hiện nay bay tốt hơn con chim do nó có cơ chế bay không phải là giống như cơ chế bay của con chim.

Như vậy, trí tuệ nhân tạo là khả năng của máy tính khi thực hiện các công việc mà con người thường phải xử lý và kết quả thực hiện của máy là tốt hơn hoặc tương đương với con người. Hay nói cách khác, đánh giá sự thông minh của máy không phải dựa trên nguyên lý thực hiện nhiệm vụ đó có giống cách con người thực hiện hay không mà dựa trên kết quả hoặc cách ứng xử bên ngoài của máy có giống với kết quả hoặc cách ứng xử của con người hay không.

Các nhiệm vụ của con người thường xuyên phải thực hiện như: giải bài toán (tìm kiếm, chứng minh, lập luận), học, giao tiếp, thể hiện cảm xúc, thích nghi với môi trường xung quanh, v.v., và dựa trên kết quả thực hiện các nhiệm vụ đó để kết luận rằng một ai đó có là thông minh hay không.

Trong môn học này, chúng ta sẽ tìm hiểu các phương pháp làm cho máy tính biết cách giải bài toán, biết cách lập luận, biết cách học, v.v.

1.2 LỊCH SỬ PHÁT TRIỂN CỦA TRÍ TUỆ NHÂN TẠO

Mong muốn làm cho máy có những khả năng thông minh của con người đã có từ nhiều thế kỷ trước, tuy nhiên Trí tuệ nhân tạo thực sự xuất hiện khi con người sáng tạo ra máy tính.

Alan Turing - nhà toán học lỗi lạc người Anh, người được xem là cha đẻ của ngành Tin học do đã đưa ra cách hình thức hóa các khái niệm thuật toán và tính toán trên máy Turing. Đây là một mô hình máy tính trừu tượng mô tả bản chất việc xử lý các ký hiệu hình thức, mô hình máy tính trừu tượng này có đóng góp quan trọng cho Trí tuệ nhân tạo vào năm 1950, gọi là phép thử Turing.

Mùa hè năm 1956 tại trường Dartmouth ở Mỹ, hội nghị đầu tiên về Trí tuệ nhân tạo do Marvin Minsky và John McCarthy tổ chức với sự tham dự của nhiều nhà khoa học, trong đó có Allen Newell và Herbert Simon. Nhiều người tham gia hội nghị Dartmouth sau này đã trở thành những nhà khoa học đầu ngành nghiên cứu về Trí tuệ nhân tạo trong suốt nhiều thập kỷ, trong số đó có giáo sư Donald Michie, người đã lập ra phòng thí nghiệm Trí tuệ nhân tạo nổi tiếng tại đại học Edinburgh ở Anh. Tại hội nghị Dartmouth, McCarthy đã đề nghị tên gọi 'Artificial Intelligence', mặc dù còn tranh cãi trong một thời gian, tuy vậy tên gọi này vẫn được thừa nhận và được dùng rộng rãi cho đến ngày nay.

Những người sáng lập ngành Trí tuệ nhân tạo đều đã lần lượt được nhận giải Turing của ACM (Hội Tin học lớn nhất thế giới) – được xem là giải Nobel của Tin học, mỗi năm thường chỉ trao cho một người: Minsky (1969), McCarthy (1971), Newell và Simon (1975). Khoảng năm 1965 Simon tuyên bố: "*Máy móc trong vòng hai mươi năm nữa sẽ có khả năng làm tất cả mọi việc con người làm*", hoặc năm 1967 Minsky tiên đoán: "*Quãng một thế hệ nữa, việc tạo ra trí thông minh nhân tạo sẽ cơ bản được giải quyết*". Tuy nhiên khoảng 15 năm sau đó, các nghiên cứu về Trí tuệ nhân tạo không thu được nhiều kết quả như kỳ vọng.

Đến đầu những năm 1980, có thể nói ngành khoa học Trí tuệ nhân tạo được hồi sinh, bùng nổ và có nhiều cuộc thi liên quan mang tầm cỡ quốc tế. Ý tưởng cơ bản để

phát triển Trí tuệ nhân tạo trong giai đoạn này là sự thông minh của máy tính không thể chỉ dựa trên việc suy diễn logic mà phải dựa vào tri thức con người, và dùng khả năng suy diễn của máy để khai thác các tri thức này. Cốt lõi của Trí tuệ nhân tạo có thể diễn giải bởi công thức Trí tuệ nhân tạo = Tri thức + Suy diễn, tiêu biểu trong giai đoạn này đó là sự ra đời của các hệ chuyên gia, chẳng hạn như hệ DENDRAL, hệ MYCIN.

Đến những năm 1990, các nghiên cứu về các chủ đề Trí tuệ nhân tạo như xử lý ngôn ngữ tự nhiên, phương pháp mạng neuron, tính toán tiến hóa, trò chơi, ...đã có nhiều kết quả và đóng vai trò to lớn trong việc nghiên cứu Trí tuệ nhân tạo sau này.

Đến nay, AI được hầu hết các quốc gia trên thế giới tập trung nghiên cứu. Mỹ, Trung Quốc, Ấn Độ, Nhật Bản, Hàn Quốc có thể xem là các nước có nhiều thành tựu trong lĩnh vực Trí tuệ nhân tạo nói chung và robot nói riêng.

1.3 QUÁ TRÌNH PHÁT TRIỂN CỦA TTNT

Thực ra nguồn gốc ra đời trí tuệ nhân tạo AI không hề sớm như đã nói, nhưng nó là thành quả tất yếu của sự phát triển khoa học và công nghệ, là giải pháp giải quyết những bài toán khó của sự phát triển loài người trong tương lai. Dưới đây chúng ta cùng điểm lại những cột mốc của lịch sử phát triển trí tuệ nhân tạo AI

Ngày nay, việc tiếp tục nghiên cứu và cải tiến trí tuệ nhân tạo AI về các công nghệ nền tảng đã thể hiện rõ trong các kỹ năng tự động hóa và lý luận có thể được tích hợp trong điện thoại, máy tính và máy móc.... Trí tuệ nhân tạo AI theo cách nào đang trở thành một thực tế nền tảng của thế giới hiện nay.

Nghiên cứu lịch sử trí tuệ nhân tạo AI sớm vào những năm thập niên 60 đã khám phá các vấn đề mà công nghệ này có thể giải quyết. Vào những năm 1960, Bộ Quốc phòng Hoa Kỳ đã quan tâm đến loại công việc này và bắt đầu đào tạo máy tính để bắt chước lý luận cơ bản của con người. Ví dụ, Cơ quan Dự án Nghiên cứu Quốc phòng Tiên tiến (DARPA) đã hoàn thành các dự án lập bản đồ đường phố vào những năm 1970. Và DARPA đã sản xuất trợ lý cá nhân thông minh vào năm 2003...

Công việc ban đầu này đã mở đường cho tự động hóa và lý luận chính thức mà chúng ta thấy trong các máy tính ngày nay, bao gồm các hệ thống hỗ trợ quyết định và hệ thống tìm kiếm thông minh có thể được thiết kế để bổ sung và tăng cường khả năng của con người.

1943: Warren McCullough và Walter Pitts xuất bản cuốn "A Logical Calculus of Ideas Immanent in Nervous Activity", dịch ra là "Một tính toán logic của những ý tưởng tiềm ẩn trong hoạt động thần kinh". Bài viết đề xuất mô hình toán học đầu tiên để xây dựng một mạng lưới thần kinh.

1949: Trong cuốn sách "The Organization of Behavior: A Neuropsychological Theory"- *Tổ chức hành vi: Một lý thuyết thần kinh học,* Donald Hebb đề xuất lý thuyết về các hệ thống con đường thần kinh được tạo ra từ các kết nối giữa các tế bào thần kinh trở nên mạnh mẽ hơn.

1950

- Alan Turing xuất bản "Computing Machinery and Intelligence" – Máy tính và trí thông minh, đề xuất Thử nghiệm Turing, một phương pháp để xác định xem một máy tính có thông minh hay không.
- Đại học Harvard Marvin Minsky và Dean Edmonds xây dựng SNARC, máy tính mạng thần kinh đầu tiên.
- Claude Shannon xuất bản bài báo "Lập trình máy tính để chơi cờ".
- Isaac Asimov xuất bản "Ba định luật về robot".

1952: Arthur Samuel phát triển một chương trình tự học để chơi cờ.

1954: Thí nghiệm dịch máy Georgetown-IBM tự động dịch 60 câu tiếng Nga được chọn cẩn thận sang tiếng Anh.

1956: Cụm từ trí tuệ nhân tạo lần đầu tiên được nói đến tại "Dự án nghiên cứu mùa hè về trí tuệ nhân tạo". Với sự dẫn đầu bởi John McCarthy, hội nghị, trong đó xác định phạm vi và mục tiêu của AI, được coi là sự ra đời của trí tuệ nhân tạo như chúng ta biết ngày nay. Allen Newell và Herbert Simon trình diễn Nhà lý luận logic (LT), chương trình lý luận đầu tiên.

1958: John McCarthy phát triển ngôn ngữ lập trình AI Lisp và xuất bản bài báo "Programs with Common Sense". Bài viết đã đề xuất nhà tư vấn giả thuyết, một hệ thống AI hoàn chỉnh với khả năng học hỏi kinh nghiệm hiệu quả như con người.

1959

- Allen Newell, Herbert Simon và JC Shaw giải quyết vấn đề chung (GPS), một chương trình được thiết kế để bắt chước giải quyết vấn đề của con người.

- Herbert Gelernter phát triển chương trình Định lý hình học.
- Arthur Samuel đồng xu với thuật ngữ học máy khi còn ở IBM.
- John McCarthy và Marvin Minsky đã tìm thấy Dự án Trí tuệ nhân tạo MIT.

1963: John McCarthy bắt đầu Phòng thí nghiệm AI tại Stanford.

1966: Báo cáo của Ủy ban Tư vấn xử lý ngôn ngữ tự động (ALPAC) của chính phủ Hoa Kỳ nêu chi tiết về sự thiếu tiến bộ trong nghiên cứu dịch máy, một sáng kiến lớn của chiến tranh lạnh với lời hứa dịch tự động tiếng Nga. Báo cáo ALPAC dẫn đến việc hủy bỏ tất cả các dự án MT do chính phủ tài trợ.

1969: Các hệ thống chuyên gia thành công đầu tiên được phát triển trong DENDRAL, một chương trình XX và MYCIN, được thiết kế để chẩn đoán nhiễm trùng máu, được tạo ra tại Stanford.

1972: Ngôn ngữ lập trình logic PRITAL được tạo ra.

1973: "Báo cáo Lighthill", nêu chi tiết về sự thất bại trong nghiên cứu AI, được chính phủ Anh công bố, từ đây dẫn đến việc cắt giảm nghiêm trọng tài trợ cho các dự án trí tuệ nhân tạo.

1974 - 1980: Liên tiếp là sự thất vọng với sự phát triển của AI dẫn đến sự cắt giảm DARPA lớn trong các khoản trợ cấp học thuật. Kết hợp với báo cáo ALPAC trước đó và "Báo cáo Lighthill" năm trước, tài trợ trí tuệ nhân tạo làm khô và các quầy nghiên cứu. Thời kỳ này được gọi là "Mùa đông AI đầu tiên."

1980: Tập đoàn thiết bị kỹ thuật số phát triển R1 (còn được gọi là XCON), hệ thống chuyên gia thương mại thành công đầu tiên. Được thiết kế để định cấu hình các đơn đặt hàng cho các hệ thống máy tính mới, R1 khởi đầu sự bùng nổ đầu tư vào các hệ thống chuyên gia sẽ tồn tại trong phần lớn thập kỷ, kết thúc hiệu quả "Mùa đông AI" đầu tiên.

1982: Bộ Thương mại Quốc tế và Công nghiệp Nhật Bản khởi động dự án Hệ thống máy tính thế hệ thứ năm đầy tham vọng. Mục tiêu của FGCS là phát triển hiệu năng giống như siêu máy tính và một nền tảng để phát triển trí tuệ nhân tạo AI.

1983: Đáp lại FGCS của Nhật Bản, chính phủ Hoa Kỳ khởi động Sáng kiến điện toán chiến lược để cung cấp nghiên cứu được tài trợ bởi DARPA trong điện toán tiên tiến và trí tuệ nhân tạo.

1985: Các công ty đang chi hơn một tỷ đô la một năm cho các hệ thống chuyên gia và toàn bộ ngành công nghiệp được gọi là thị trường máy Lisp mọc lên để hỗ trợ họ. Các công ty như Symbolics và Lisp Machines Inc. xây dựng các máy tính chuyên dụng để chạy trên ngôn ngữ lập trình AI Lisp.

1987-1993

- Khi công nghệ điện toán đám mây được cải thiện, có nhiều lựa chọn thay thế rẻ hơn xuất hiện và thị trường máy Lisp sụp đổ vào năm 1987, mở ra "Mùa đông AI thứ hai". Các chuyên gia AI rất chật vật và không được sự ủng hộ trong giai đoạn này.
- DARPA kết thúc Sáng kiến Điện toán Chiến lược vào năm 1993 sau khi chi gần 1 tỷ đô la và không đạt được kỳ vọng như đã tính toán.

1991: Lực lượng Hoa Kỳ triển khai DART, một công cụ lập kế hoạch và lập kế hoạch hậu cần tự động, trong Chiến tranh vùng Vịnh.

2005: STANLEY, một chiếc xe tự lái, chiến thắng DARPA Grand Challenge.

Quân đội Hoa Kỳ bắt đầu đầu tư vào các robot tự hành như "Big Dog" của Boston Dynamic và "PackBot" của iRobot.

2008: Google tạo ra những bước đột phá trong nhận dạng giọng nói và giới thiệu tính năng này trong ứng dụng iPhone.

2011: Watson của IBM tuyên bố cạnh tranh về Jeopardy!

2012: Andrew Ng, người sáng lập dự án Google Brain Deep Learning, cung cấp một mạng lưới thần kinh bằng cách sử dụng thuật toán 10 triệu video YouTube dưới dạng tập huấn luyện. Mạng lưới thần kinh đã học cách nhận ra một con mèo mà không được cho biết con mèo là gì.

2014: Google tạo ra chiếc xe tự lái đầu tiên để vượt qua bài kiểm tra lái xe của nhà nước.

2016: AlphaGo của Google DeepMind đánh bại nhà vô địch thế giới cờ vây Lee Sedol. Sự phức tạp của trò chơi Trung Quốc cổ đại được coi là một trở ngại lớn để giải tỏa trong AI.

Với chặng đường lịch sử trí tuệ nhân tạo phát triển nhanh chóng cả về kiến thức lẫn tiến bộ trong trí tuệ nhân tạo đã và đang thúc đẩy mạnh mẽ nhu cầu về các dịch

vụ kỹ thuật số mới để giúp khai thác công nghệ này với tiềm năng cao nhất. Cải thiện các sản phẩm đã có trên thị trường để tất cả chúng ta đưa vào sử dụng trong cuộc sống hàng ngày sẽ là cốt lõi cho tương lai của trí tuệ nhân tạo AI.

Hiện nay, trí tuệ nhân tạo là đột phá công nghệ mới nhất, là ngành khoa học đang định hình lại xã hội của chúng ta. Đồng thời trí tuệ nhân tạo có tác động sâu sắc đến các ngành công nghiệp máy móc và công ty cung cấp năng lượng.

1.4 ỨNG DỤNG CỦA TTNT

Trí tuệ nhân tạo là một công nghệ thú vị và được ứng dụng ngày càng nhiều trong các lĩnh vực khác nhau, từ việc phục vụ đời sống hàng ngày của con người cho đến giáo dục, tài chính ngân hàng, y tế. Ứng dụng thực tế của trí tuệ nhân tạo có thể giúp nâng cao hiệu suất lao động, cải thiện chất lượng sống của con người, đem đến cơ hội tăng trưởng kinh tế, phát triển kinh doanh cho các doanh nghiệp.

Một số lĩnh vực Trí tuệ nhân tạo hiện đang được nhiều nhóm tham gia nghiên cứu, bao gồm:

- Học máy (machine learning)
- Xử lý ngôn ngữ tự nhiên (natural language processing)
- Giao diện người máy thông minh (human machine Interface)
- Người máy (robot)
- Mạng neural
- Hệ mờ (fuzzy system)
- Tính toán tiến hóa (evolutionary computation)
- Hệ chuyên gia (expert system)
- Xử lý âm thành và xử lý tiếng nói
- Lập luận và giải quyết vấn đề tự động, ...
- Ngoài ra, Trí tuệ nhân tạo có nhiều ứng dụng trong trong các lĩnh vực quân sự, mô phỏng, y khoa, robot, giao thông, bảo mật, camera thông minh, nhà thông minh, trò chơi thông minh, ...

Trí tuệ nhân tạo được ứng dụng rộng rãi để giải quyết các vấn đề của cuộc sống thực tế trên hầu hết các lĩnh vực, chẳng hạn như:

- **Trong lĩnh vực chăm sóc sức khỏe**: Trí tuệ nhân tạo góp phần cải thiện tình trạng sức khỏe bệnh nhân, và giúp giảm chi phí điều trị. Một trong những hệ thống công nghệ chăm sóc sức khỏe tốt nhất phải kể đến là IBM Watson, IBM Watson có khả năng hiểu được các ngôn ngữ tự nhiên và có khả năng phản hồi các câu hỏi được yêu cầu hoặc cho phép bệnh nhân tra cứu thông tin về tình hình sức khoẻ của mình. IBM Watson có thể lướt duyệt cùng lúc hàng triệu hồ sơ bệnh án để cung cấp cho các bác sĩ những lựa chọn điều trị dựa trên bằng chứng chỉ trong vòng vài giây nhờ khả năng tổng hợp dữ liệu khổng lồ và tốc độ xử lý mạnh mẽ. IBM Watson khai thác dữ liệu bệnh nhân và các nguồn dữ liệu sẵn có khác nhằm tạo ra giả thuyết và từ đó xây dựng một lược đồ điểm tin cậy giúp bác sĩ đưa ra quyết định điều trị cuối cùng. Ngoài ra, ứng dụng Trí tuệ nhân tạo nổi bậc khác trong lĩnh vực này cần phải kể đến là chatbot - chương trình máy tính trực tuyến để trả lời các câu hỏi và hỗ trợ khách hàng, sắp xếp các cuộc hẹn hoặc trợ giúp bệnh nhân thông qua quá trình thanh toán và các trợ lý y tế ảo cung cấp phản hồi y tế cơ bản.
- **Trong lĩnh vực kinh doanh**: Các tác vụ mà con người thực hiện lặp đi lặp lại giờ đây đã được tự động hoá quy trình bằng robot. Các thuật toán Machine Learning được tích hợp trên các nền tảng phân tích và CRM (Customer Relationship Management - quản lý quan hệ khách hàng) để khám phá các thông tin về cách phục vụ khách hàng tốt hơn. Chatbots được tích hợp trên các trang web nhằm cung cấp dịch vụ ngay lập tức cho khách hàng. Một số hệ thống trợ lý ảo nổi tiếng giúp sắp xếp, nhắc cuộc họp, tìm kiếm thông tin như Google Assistant, Alexa, Siri. Hiện nay các hệ thống này đã bắt đầu được tích hợp vào trong các thiết bị gia dụng như máy giặt, tủ lạnh, lò vi sóng, ... giúp người sử dụng có thể điều khiển thiết bị bằng câu lệnh thoại.
- **Trong lĩnh vực giáo dục**: Công nghệ thực tế ảo làm thay đổi cách dạy và học. Sinh viên có thể đeo kính VR và có cảm giác như đang ngồi trong lớp nghe giảng bài hay nhập vai để chứng kiến những trận đánh giả lập, ngắm nhìn di tích, điều này giúp mang lại cảm xúc và ghi nhớ sâu sắc nội dung học. Hoặc khi đào tạo nghề phi công, học viên đeo kính sẽ thấy phía trước là cabin và học lái máy bay như thật để thực hành giúp giảm thiểu rủi ro trong quá trình bay thật.

- **Trong lĩnh vực tài chính**: Trí tuệ nhân tạo áp dụng cho các ứng dụng tài chính cá nhân như Mint hay Turbo Tax giúp tăng cường các định chế tài chính.
- **Trong lĩnh vực pháp luật**: Quá trình khám phá, chọn lọc thông qua các tài liệu trong luật pháp thường áp đảo đối với con người. Tự động hóa quá trình này giúp tiết kiệm thời gian và quá trình làm việc hiệu quả hơn. Các trợ lý ảo giúp trả lời các câu hỏi đã được lập trình sẵn.
- **Trong lĩnh vực sản xuất**: Đây là lĩnh vực đi đầu trong việc kết hợp robot vào luồng công việc. Robot công nghiệp được sử dụng để thực hiện các nhiệm vụ đơn lẻ và đã được tách ra khỏi con người. Xe tự động lái Tesla là một ứng dụng điển hình trong lĩnh vực này.
- **Trong lĩnh vực bảo mật thông tin**: rất nhiều hệ thống nhận diện và bảo mật thông minh được xây dựng, phải kể đến như FaceID - bảo mật thông qua nhận diện khuôn mặt của Apple, Facebook với khả nhận diện khuôn mặt để gợi ý tag. Bên cạnh các nước phương Tây thì Trung Quốc hiện đang là quốc gia đi đầu trong việc sử dụng Trí tuệ nhân tạo để nhận diện và quản lý công dân.

Mặc dù ưu thế của máy tính so với bộ não người là khả năng tính toán và lưu trữ rất lớn. Tuy nhiên, phải cần nhiều thời gian nữa mới có thể so sánh hoạt động của máy tính với hệ thần kinh của con người, ít nhất cũng là ở các mặt sau:

- Chưa tự sinh ra được các heuristic để giải quyết các vấn đề bài toán có độ phức tạp cao.
- Công nghệ vi mạch hiện đại cho phép xây dựng các bộ đa xử lý nhưng máy tính chưa thể hoạt động song song như bộ não của con người.
- Khả năng diễn giải: Con người có thể xem xét cùng một vấn đề theo những phương pháp khác nhau, từ đó diễn giải theo cách dễ hiểu nhất. Ngược lại, sự linh hoạt này chưa thể mô phỏng được trong các hệ thống Trí tuệ nhân tạo.
- Logic rời rạc và tính liên tục: Một thách thức với các hệ thống Trí tuệ nhân tạo là khả năng kết hợp các phương pháp xử lý thông tin trong môi trường liên tục với các thao tác xử lý thông tin rời rạc.
- Khả năng học: Mặc dù hiện nay máy tính có nhiều tính năng cao, tuy vậy cũng chưa thể mô phỏng được hoàn toàn khả năng học giống bộ não của con người.
- Khả năng tự tổ chức: Cho tới nay, con người chưa thể tạo lập được các hệ thống Trí tuệ nhân tạo có khả năng tự tổ chức, tự điều khiển hoạt động của nó để thích nghi với môi trường, ...

BÀI 2: TÔ MÀU ĐỒ THỊ

Học xong bài này sinh viên có thể nắm được:

- *Các khái niệm, thuộc tính cơ bản về đồ thị.*
- *Các kỹ thuật tô màu tối ưu, tô màu tham lam.*
- *Thuật giải lai ghép.*

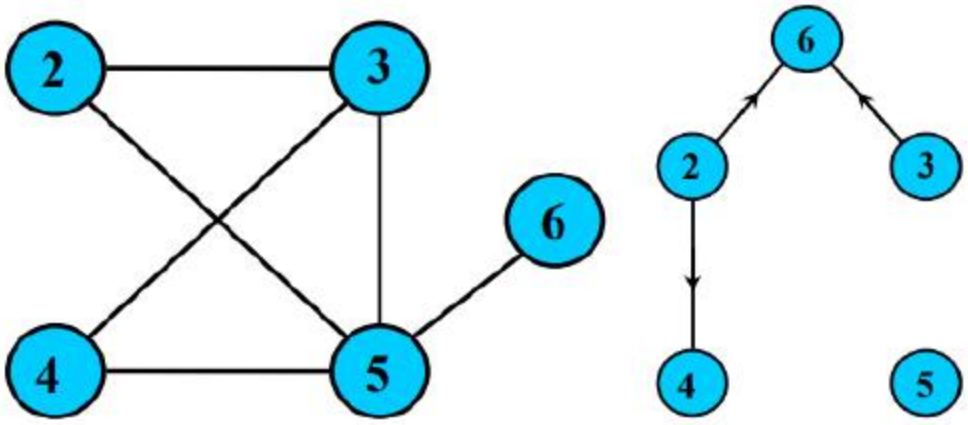
2.1 THUẬT GIẢI TÔ MÀU TỐI ƯU

2.1.1 Đồ thị

Một đồ thị là một tập hợp hữu hạn các đỉnh và tập các cạnh nối các đỉnh với nhau.

Kí hiệu: $G = (V, E)$, G (Graph) – đồ thị, V (vertex) - tập đỉnh, E (edge) - tập cạnh.

Ví dụ 2.1: Cho tập $V = \{2, 3, 4, 5, 6\}$, quan hệ này được biểu diễn bằng đồ thị Hình 2.1.



Hình 2.1: (a) Đồ thị vô hướng; (b) Đồ thị có hướng

2.1.2 Phân loại đồ thị

a. Đồ thị vô hướng $G = (V, E)$ trong đó:

- V là tập các đỉnh

- E là tập hợp các cạnh, mỗi cạnh là một cặp không thứ tự $(u, v) \in V$.
- (u, v) được gọi là cạnh nối đỉnh u và đỉnh v, ta có $(u, v) \equiv (v, u)$

b. Đồ thị có hướng $G = [V, E]$ trong đó:

- V là tập các đỉnh
- E là tập các cạnh, mỗi cạnh là một cặp có thứ tự $[u, v] \in V$
- $[u, v]$ được gọi là cung nối từ đỉnh u đến đỉnh v, với $[u, v] \neq [v, u]$

2.1.3 Một số thuật ngữ cơ bản

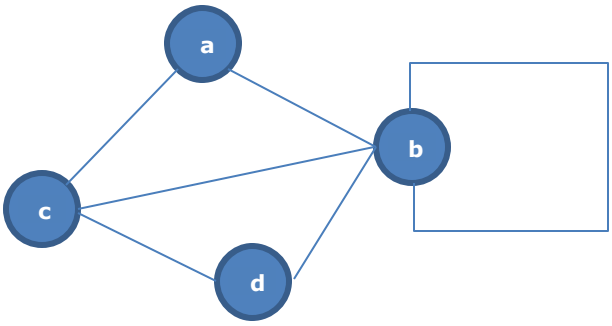
Với cạnh $e = (u, v) \in E$; $u, v \in V$, khi đó:

- e là cạnh liên thuộc u và v.
- u, v được gọi là kề nhau hay láng giềng của nhau.
- u, v gọi là hai đầu mút của cạnh e.
- Nếu $e = [u, v]$ thì u gọi là đỉnh xuất phát và v gọi là đỉnh đích của cung e.
- Nếu $u \equiv v$ thì e được gọi là khuyên.
- Nếu có $e' = (u, v)$ thì e và e' được gọi là cạnh kép.

Bậc của đỉnh

Cho đồ thị vô hướng $G = (V, E)$. Bậc của đỉnh v, ký hiệu $deg(v)$, là số cạnh kề với v, trong đó một khuyên tại một đỉnh được đếm hai lần cho bậc của đỉnh ấy.

Cho đồ thị vô hướng $G = (V, E)$, bậc của các đỉnh như sau:



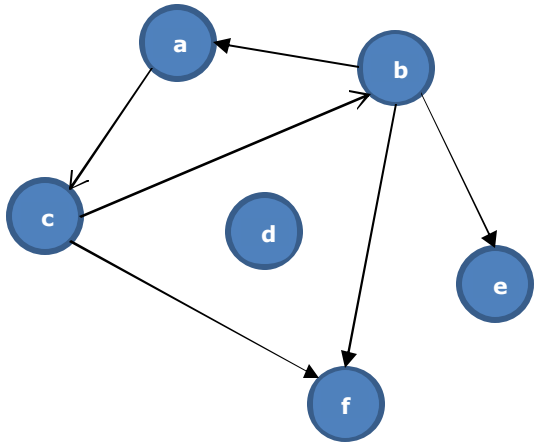
- Bậc đỉnh a: $deg(a) = 2$
- Bậc đỉnh b: $deg(b) = 5$
- Bậc đỉnh c: $deg(c) = 3$
- Bậc đỉnh d: $deg(d) = 2$

Cho đồ thị có hướng $G = (V, E)$, $v \in V$

- $deg^-(v)$ = số cung có đỉnh cuối là v, gọi là bậc vào của v.

- $deg^+(v)$ = số cung có đỉnh đầu là v, gọi là bậc ra của v
- $deg(v) = deg^-(v) + deg^+(v)$
- Đỉnh bậc 0 gọi là đỉnh cô lập
- Đỉnh bậc 1 gọi là đỉnh treo

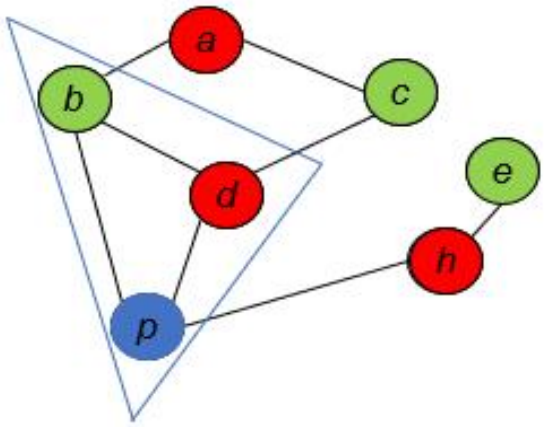
Cho đồ thị có hướng $G = [V, E]$, bậc của các đỉnh như sau:



- Bậc đỉnh a: $deg^-(a) = 2$; $deg^+(a) = 1$
- Bậc đỉnh b: $deg^-(b) = 1$; $deg^+(b) = 3$
- Bậc đỉnh c: $deg^-(c) = 1$; $deg^+(c) = 2$
- Bậc đỉnh d: $deg^-(d) = 0$; $deg^+(d) = 0$
- Bậc đỉnh e: $deg^-(e) = 1$; $deg^+(e) = 0$
- Bậc đỉnh f: $deg^-(f) = 2$; $deg^+(f) = 0$

2.1.4 Tô màu đồ thị

Tô màu đồ thị là một trong những bài toán cơ bản được ứng dụng nhiều trong tin học, nhằm giải quyết các vấn đề thực tiễn liên quan đến sắp xếp lịch thi - phòng thi, tô màu bản đồ, ... Tô màu đồ thị là việc thực hiện gán màu cho mỗi đỉnh của đồ thị, sao cho hai đỉnh kề nhau không cùng một màu, và số màu được sử dụng là ít nhất. Số màu ít nhất có thể sử dụng để tô màu đồ thị được gọi là sắc số của đồ thị đó.



Hình.2.2: Đồ thị được tô màu với ít nhất 3 màu

Thuật toán tô màu được mô tả đơn giản như sau:

Input: đồ thị $G = (V, E)$.

Output: đồ thị $G = (V, E)$ có các đỉnh đã được tô màu.

Các bước của thuật toán:

Bước 1:

- *Tính giá trị bậc của các đỉnh $v_i \in V$.*
- *Sắp xếp bậc của các đỉnh theo thứ tự giảm dần $d(v_1) > d(v_2) > ... > d(v_n)$, và lưu vào danh sách $V' = [v_1 v_2, ..., v_n]$. Ban đầu tất cả các đỉnh trong V (V') đều chưa được tô màu.*
- *Khởi tạo màu c = 1*

Bước 2:

- *Tô màu c cho đỉnh đầu tiên trong danh sách V'.*
- *Duyệt lần lượt các đỉnh khác trong V' (nếu có) và chỉ tô màu c cho các đỉnh không kề đỉnh đã có màu c.*

Bước 3:

- *Nếu tất cả các đỉnh trong V đã được tô màu thì thuật toán kết thúc → đồ thị sử dụng c màu để tô.*
- *Ngược lại, nếu vẫn còn đỉnh chưa được tô thì chuyển sang **Bước 4.***

Bước 4:

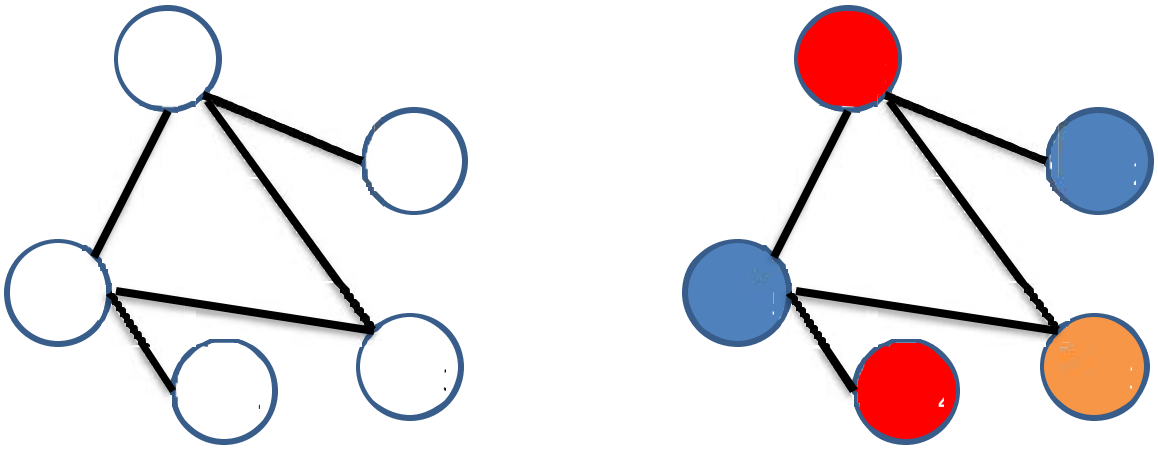
- *Loại khỏi danh sách V' các đỉnh đã tô màu.*
- *Sắp xếp lại các đỉnh trong V' theo thứ tự bậc giảm dần.*
- *c = c + 1 và lặp lại **Bước 2.***

Mã giả chương trình

```
c=1;
số đỉnh đã tô = 0;
mọi đỉnh đều chưa được tô;
do
```

```
{
        for i=1 to n
        if (đỉnh i là chưa xét và có thể tô được bằng màu c)
        {
                tô đỉnh i bằng màu c, đỉnh i trở thành đỉnh đã xét
                số đỉnh đã được tô = số đỉnh đã được tô + 1
        }
        c++;
}
    while (số đỉnh đã tô < n);
```

Ví dụ 2.2: Cho đồ thị như Hình 2.3, sử dụng thuật toán ở trên để tô màu cho các đỉnh của đồ thị.



Hình 2.3: Minh hoạ tô màu đồ thị

Bước 1: Ta có đồ thị có 5 đỉnh được đánh số 1, 2, 3, 4, 5 với các bậc tương ứng với từng đỉnh theo thứ tự là 3, 1, 2, 1, 3. Do đó, *V'* ban đầu có thứ tự là [1, 5, 3, 2, 4], i =1.

Bước 2: Tô màu 1- red cho đỉnh 1.

Lần lượt duyệt các đỉnh còn lại trong *V'*: đỉnh 5 kề đỉnh 1 (đỉnh 1 đã tô màu 1 - red) nên chưa tô màu cho đỉnh 5. Tương tự các đỉnh 3, 2 đều kề với đỉnh 1 nên đỉnh 3, 2 cũng chưa được tô màu.

Đỉnh 4 không kề với đỉnh 1, do đó thực hiện tô màu 1 cho đỉnh 4 → đỉnh 4 có màu 1 - red.

Bước 3: Kiểm tra thấy vẫn còn các đỉnh trong V chưa được tô màu nên chuyển sang bước 4.

Bước 4: Loại bỏ các đỉnh 1, 4 đã được tô màu ra khỏi *V'*, sắp xếp lại *V'* theo thứ tự bậc giảm dần, *V'*= [5, 3, 2]. Ta có i = 2, thực hiện lặp lại Bước 2:

- **Bước 2(1):** Tô màu 2 - blue cho đỉnh 5. Lần lượt duyệt các đỉnh còn lại trong *V'*. Đỉnh 3 kề đỉnh 5 (đã tô màu 2 - blue) nên chưa tô màu cho đỉnh 3.

Đỉnh 2 không kề với đỉnh 5, do đó thực hiện tô màu 2 cho đỉnh 2 → đỉnh 2 có màu 2 – blue.

- **Bước 3(1):** Kiểm tra thấy vẫn còn đỉnh 3 chưa được tô màu nên chuyển sang bước 4.
- **Bước 4(1):** Loại bỏ các đỉnh 5, 2 đã được tô màu ra khỏi *V'*, *V'*= [3]. Ta có i = 3.

Lặp lại bước 2:

- **Bước 2(2):** Tô màu 3 - green cho đỉnh 3.
- **Bước 3(2):** Kiểm tra thấy tất cả các đỉnh trong *V* đã được tô màu, thuật toán dừng lại.

Kết quả: Số màu cần thiết phải sử dụng là 3 màu (màu 1-red: đỉnh 1, 4; màu 2-blue: đỉnh 5, 2; màu 3-green: đỉnh 3).

2.1.5 Tô màu tối ưu

Cho một đồ thị $G = (V, E)$ gồm n đỉnh (thành phố), quan hệ giữa đỉnh v_i và đỉnh v_j, kí hiệu C_{ij}, là 1 nếu đỉnh v_i có cung nối với đỉnh v_j và 0 nếu ngược lại.

$$C_{ij} = \begin{cases} 1, \text{nếu đỉnh } v_i \text{ có cung nối đỉnh } v_j \\ 0, \text{nếu đỉnh } v_i \text{ không có cung nối đỉnh } v_j \end{cases}$$

Các bước của thuật toán

Bước 1: Tính bậc các đỉnh

- *Tính bậc của tất cả các đỉnh.*

Bước 2: Tô màu

- *Khởi tạo i = 1*
- *Tô màu i cho đỉnh có bậc lớn nhất.*

Bước 3: Hạ bậc & cấm tô

- *Gán bậc của đỉnh đã được tô màu i là 0.*
- *Hạ bậc các đỉnh có quan hệ với đỉnh được tô màu i, bậc = bậc – 1.*
- *Cấm tô màu i cho các đỉnh có quan hệ với đỉnh được tô màu i.*

Bước 4:

- *Lặp lại bước 2 cho đến khi tất cả các đỉnh đều được tô màu.*

Mã giả chương trình

while (còn đỉnh có bậc > 0)

{

Tìm đỉnh có bậc lớn nhất chưa được tô màu (giả sử đỉnh v_i).
Tô màu j cho đỉnh v_i.
Gán bậc của đỉnh được tô $v_i = 0$.
Giảm bậc các đỉnh kề với đỉnh v_i 1 đơn vị.
Cấm tô màu j cho các đỉnh kề với đỉnh v_i.

}

Sau khi kết thúc vòng lặp trên có thể còn đỉnh chưa được tô nhưng tất cả các đỉnh lúc này đều đã có bậc bằng 0 – nghĩa là không thể hạ bậc được nữa. Khi đó màu của các đỉnh chưa được tô chính là màu nhỏ nhất hợp lệ trong danh sách màu.

Ví dụ 2.3: Một công ty có 8 đài phát thanh A, B, C, D, E, F, G, H có khoảng cách (km) được cho trong ma trận sau:

	A	B	C	D	E	F	G	H
A	0	100	50	30	200	150	40	120
B		0	30	80	120	50	200	150
C			0	120	100	30	80	50
D				0	50	120	150	30
E					0	200	120	120
F						0	180	150
G							0	50
H								0

Do yêu cầu kỹ thuật nên các đài có khoảng cách ≥ 100km không được dùng chung một trạm phát sóng. Hãy lắp đặt các trạm phát sóng sao cho số trạm cần lắp là ít nhất.

Giải quyết

1. Xác định đồ thị

- Đỉnh: gồm các đài phát thanh
- Cung: nối giữa 2 đài có khoảng cách ≥ 100km.

Ta có ma trận quan hệ (hay đồ thị quan hệ như sau):

	A	B	C	D	E	F	G	H	Bậc
A	0	1	0	0	1	1	0	1	4
B	1	0	0	0	1	0	1	1	4
C	0	0	0	1	1	0	0	0	2
D	0	0	1	0	0	1	1	0	3
E	1	1	1	0	0	1	1	1	6
F	1	0	0	1	1	0	1	1	5
G	0	1	0	1	1	1	0	0	4
H	1	1	0	0	1	1	0	0	4

2. Áp dụng thuật giải để tô màu

Bước 1: Tô màu (i = 1) cho đỉnh E có bậc cao nhất.

i		A	B	C	D	E	F	G	H
1	Bậc	4	4	2	3	6	5	4	4

Bước 2: Hạ bậc và cấm tô màu i cho các đỉnh có quan hệ với E

	Cấm	1	1	1		1	1	1	1
i		A	B	C	D	E	F	G	H
1	Bậc	3	3	1	3	0	4	3	3

Bước 3: Quay lại Bước 1

Tô màu 2 (i=2) cho đỉnh F (bậc 4)

	Cấm	1,2	2	1	2	1	2	1,2	1,2
i		A	B	C	D	E	F	G	H
1	Bậc	3	3	1	3	0	4	3	3
2		2	3	1	2	--	0	2	2
		1	0	1	2	-	-	1	1

Tô màu 2 cho B (bậc 3), không có quan hệ với F.

	Cấm	1,2	2	1	2	1	2	1,2	1,2
i		A	B	C	D	E	F	G	H
1	Bậc	3	3	1	3	0	4	3	3
2		2	3	1	2	--	0	2	2
		1	0	1	2	-	-	1	1

Tô màu 1 cho đỉnh D (bậc 2, không có quan hệ với E)

	Cấm	1,2	1, 2	1	1, 2	1	2	1,2	1,2
i		A	B	C	D	E	F	G	H
1	Bậc	3	3	1	3	0	4	3	3
2		2	3	1	2	--	0	2	2
		1	0	1	2	-	-	1	1
3		1	-	0	0	-	-	0	1

Tô màu 3 cho đỉnh A (bậc 1)

	Cấm	3	1, 2	1	1, 2	1	2	1,2	1,2,3
i		A	B	C	D	E	F	G	H
1	Bậc	3	3	1	3	0	4	3	3
2		2	3	1	2	--	0	2	2
		1	0	1	2	-	-	1	1
3		1	-	0	0	-	-	0	1
		0	-	0	-	-	-	0	0

Tô màu 3 cho đỉnh G

	Cấm	3	1, 2	1	1	2	1	2	3	1,2,3
i		A	B	C	D	E	F	G	H	
1	Bậc	3	3	1	3	0	4	3	3	
2		2	3	1	2	--	0	2	2	
		1	0	1	2	-	-	1	1	
3		1	-	0	0	-	-	0	1	
		0	-	0	-	-	-	0	0	

Tô màu 4 cho đỉnh H

	Cấm	3	1, 2	1	1, 2	1	2	3	4
i		A	B	C	D	E	F	G	H
1	Bậc	3	3	1	3	0	4	3	3
2		2	3	1	2	--	0	2	2
		1	0	1	2	-	-	1	1
3		1	-	0	0	-	-	0	1
		0	-	0	-	-	-	0	0

Kết quả: Số màu cần tô là 4

- Màu 1: D, E.
- Màu 2: B, C, F
- Màu 3: A, G
- Màu 4: H

2.2 THUẬT GIẢI THAM LAM

Thuật toán tham lam[1] là một trong số các phương pháp giải quyết bài toán tối ưu. Các thuật toán tham lam dựa vào sự đánh giá tối ưu cục bộ địa phương (local optimum) để đưa ra quyết định tức thì tại mỗi bước lựa chọn, với hy vọng cuối cùng sẽ tìm được lời giải tối ưu toàn cục (global optimum). Thuật toán tham lam có thể tìm được lời giải tối ưu mà cũng có thể không. Trong trường hợp thuật toán tham lam không tìm được lời giải tối ưu, chúng ta thường thu được một lời giải có chất lượng chấp nhận được trong khoảng thời gian chấp nhận được để lời giải đó là có ý nghĩa.

Thuật toán tô màu tham lam

- ***Bước 1***: *Dùng màu thứ nhất tô cho một đỉnh tùy ý và các đỉnh khác có thể tô còn lại (đỉnh không có cạnh nối nhau).*
- ***Bước 2***: *Dùng màu thứ hai tô tiếp cho các đỉnh có thể tô còn lại.*

[1] https://vi.wikipedia.org/wiki/Giải_thuật_tham_lam

- ***Bước 3***: *Lặp lại cho đến khi nào tô tất cả các đỉnh được tô màu hết.*

Ví dụ 2.4: Một cửa hàng sách mới, mới nhập về 12 cuốn sách thuộc các loại sau: Truyện cười (A, C, D, G), âm nhạc (B, H, K), lịch sử (E, J, L), khoa học (F, I). Hãy sắp xếp các quyển sách này vào kệ sao cho số kệ sử dụng là ít nhất mà tuân theo các yêu cầu sau:

- Các quyển sách cùng loại không được để chung cùng kệ.
- Quyển A không để chung với sách khoa học.
- Quyển L không để chung với sách âm nhạc.

Giải quyết

1. Xác định đồ thị

- Đỉnh: các quyển sách
- Cạnh: các sách không được để chung cùng một kệ sách

2. Xây dựng ma trận kề.

	A	C	D	G	B	H	K	E	J	L	F	I	Bậc
A	0	1	1	1	0	0	0	0	0	0	1	1	5
C	1	0	1	1	0	0	0	0	0	0	0	0	3
D	1	1	0	1	0	0	0	0	0	0	0	0	3
G	1	1	1	0	0	0	0	0	0	0	0	0	3
B	0	0	0	0	0	1	1	0	0	1	0	0	3
H	0	0	0	0	1	0	1	0	0	1	0	0	3
K	0	0	0	0	1	1	0	0	0	1	0	0	3
E	0	0	0	0	0	0	0	0	0	1	0	0	1
J	0	0	0	0	0	0	0	0	0	1	0	0	1
L	0	0	0	0	0	1	1	1	1	0	0	0	4
F	1	0	0	0	0	0	0	0	0	0	0	1	2
I	1	0	0	0	0	0	0	0	0	0	1	0	2

Tính bậc của từng đỉnh trong ma trận.

Đỉnh	A	C	D	G	B	H	K	E	J	L	F	I
Bậc	5	3	3	3	3	3	3	1	1	4	2	2

Tô màu theo nguyên lý Greedy

Đỉnh	A	C	D	G	B	H	K	E	J	L	F	I
Bậc	5	3	3	3	3	3	3	1	1	4	2	2
Màu 1	1	1	1	1	0	0	0	0	0	0	1	1

Tô màu từ trái qua phải, bắt đầu tô màu cho đỉnh A trước. Gán giá trị 1 cho đỉnh A (màu 1), những đỉnh nào có giá trị 1 trên ma trận kề sẽ không được tô màu 1 (màu đỏ). Các đỉnh còn lại có thể tô màu 1 là B, H, K, E, J, L.

Đỉnh	A	C	D	G	B	H	K	E	J	L	F	I
Bậc	5	3	3	3	3	3	3	1	1	4	2	2
Màu 1	1	1	1	1	1	1	1	0	0	1	1	1

Tô màu 1 cho đỉnh B, cấm tô màu 1 cho đỉnh H, K, L. Còn lại đỉnh E và J, ưu tiên tô màu 1 cho đỉnh E và cấm tô màu 1 cho đỉnh J vì ma trận kề đã xác định đỉnh J phải không nằm chung kệ với đỉnh E.

Đỉnh	A	C	D	G	B	H	K	E	J	L	F	I
Bậc	5	3	3	3	3	3	3	1	1	4	2	2
Màu 1	1	1	1	1	1	1	1	1	1	1	1	1

Tương tự, lần lượt tô hết các đỉnh với các màu khác nhau.

Đỉnh	A	C	D	G	B	H	K	E	J	L	F	I
Bậc	5	3	3	3	3	3	3	1	1	4	2	2
Màu 1	1	1	1	1	1	1	1	1	1	1	1	1
Màu 2	1	2	2	2	1	2	2	1	2	2	2	2
Màu 3	1	2	3	3	1	2	3	1	2	3	2	3
Màu 4	1	2	3	4	1	2	3	1	2	4	2	3

Kết luận: 12 cuốn sách trên được xếp vào 4 kệ (được tô 4 màu ở trên).

- Kệ 1: A, B, E
- Kệ 2: C, H, J, F
- Kệ 3: D, K, I
- Kệ 4: G, L

Thuật giải sắp thứ tự và tham lam

Bước 1:

- *Sắp xếp các đỉnh theo chiều giảm dần của bậc.*
- *i = 0*

Bước 2:

- *i = i+1*
- *Tô màu i cho tất cả các đỉnh có thể tô được (xét từ trái sang).*

Bước 3:

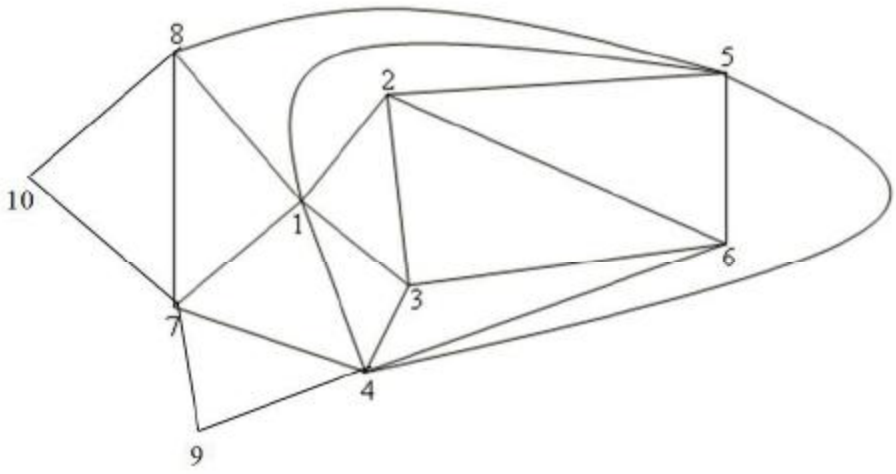
- *Lặp lại bước 2 cho đến khi tất cả các đỉnh đều được tô màu.*

Ví dụ 2.5: Một hội thảo khoa học được tổ chức với 10 chủ đề khác nhau ký hiệu là: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. Mỗi chủ đề được diễn ra trong một buổi, trong đó các chủ đề sau không được diễn ra đồng thời trong cùng một buổi: (1, 2, 3); (1, 4, 5); (3, 4, 6); (2, 5, 6); (1, 7, 8); (7, 8, 10); (4, 7, 9); (4, 7); (5, 8). Hãy lập lịch tổ chức hội thảo sao cho số buổi hội thảo diễn ra là ít nhất có thể.

Hướng dẫn

Bước 1: Xây dựng đồ thị

- Gọi mỗi chủ đề là một đỉnh của đồ thị
- Nối hai chủ đề không diễn ra đồng thời là một cạnh của đồ thị
- Theo bài ra ta có đồ thị sau:



Hình 2.4: Đồ thị quan hệ giữa các chủ đề của hội thảo

- Ma trận kề của đồ thị như sau

	1	2	3	4	5	6	7	8	9	10	Bậc
1	0	1	1	1	1	0	1	1	0	0	6
2	1	0	1	0	1	1	0	0	0	0	4
3	1	1	0	1	0	1	0	0	0	0	4
4	1	0	1	0	0	1	1	0	1	0	5
5	1	1	0	1	0	1	0	1	0	0	5
6	0	1	1	1	1	0	0	0	0	0	4
7	1	0	0	1	0	0	0	1	1	1	5
8	1	0	0	0	1	0	1	0	0	1	3
9	0	0	0	1	0	0	1	0	0	0	2
10	0	0	0	0	0	0	1	1	0	0	2

Bước 2: Tô màu theo thuật toán tham lam

Đỉnh	1	2	3	4	5	6	7	8	9	10
Bậc	6	4	4	5	5	4	5	3	2	2

Sắp xếp các đỉnh theo bậc giảm

Đỉnh	1	4	5	7	2	3	6	8	9	10
Bậc	6	5	5	5	4	4	4	3	2	2

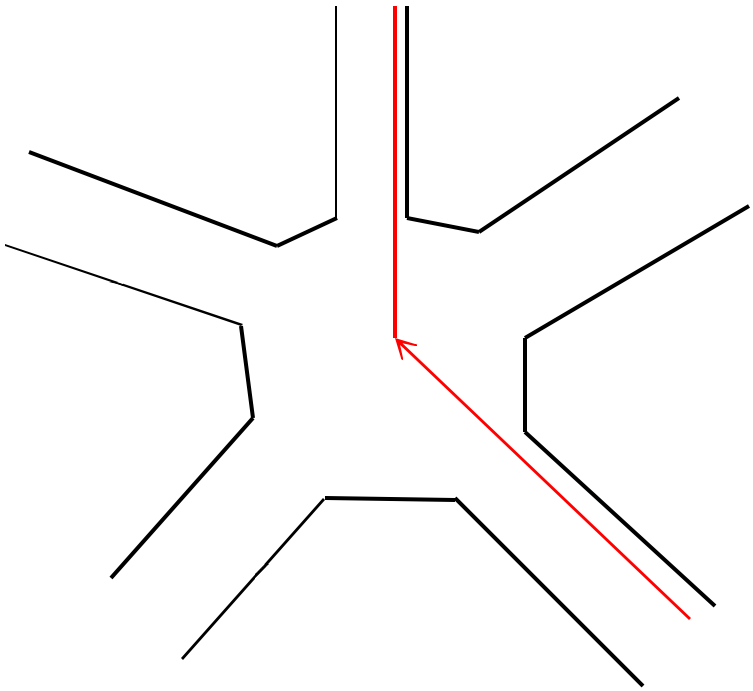
Tô màu các đỉnh

Đỉnh	1	4	5	7	2	3	6	8	9	10
Màu 1	x						x		x	x
Màu 2		x			x			x		
Màu 3			x	x		x				

Kết luận: Số buổi cần để tổ chức tất cả các buổi hội thảo là 3

- Buổi 1: Các chủ đề 1, 6, 9, 10
- Buổi 2: Các chủ đề 2, 4, 8
- Buổi 3: Các chủ đề 3, 5, 7

Ví dụ 2.6: Cho ngã năm giao thông như sau trong đó BE là đường 1 chiều.



Yêu cầu:

- Xác định đồ thị.
- Tô màu đồ thị.

  Sao cho tại mỗi thời điểm, các tuyến lưu thông không giao nhau.

Giải quyết

- Đỉnh là các tuyến đường.
- Cung là các tuyến đường có giao nhau.

  Như vậy:

- Các tuyến AE, DC, BA không giao với bất kỳ tuyến nào khác (nghĩa là toàn dòng và toàn cột =0)
- Các tuyến đường có cùng đỉnh xuất phát hoặc cùng đích đến thì không giao nhau.
- Các tuyến song song thì không giao nhau.

- Còn lại phải xem xét.

	AC	AD	AE	BA	BC	BD	BE	CA	CD	CE	DA	DC	DE	Bậc
AC	0	0	0	0	0	1	1	0	1	1	1	0	0	5
AD	0	0	0	0	0	0	1	0	0	1	0	0	1	3
AE	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BA	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BC	0	0	0	0	0	0	0	1	1	1	0	0	0	3
BD	1	0	0	0	0	0	0	1	0	1	1	0	1	5
BE	1	1	0	0	0	0	0	1	0	0	1	0	0	4
CA	0	0	0	0	1	1	1	0	0	0	0	0	0	3
CD	1	0	0	0	1	0	0	0	0	0	1	0	1	4
CE	1	1	0	0	1	1	0	0	0	0	1	0	0	5
DA	1	0	0	0	0	1	1	0	1	1	0	0	0	5
DC	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DE	0	1	0	0	0	1	0	0	1	0	0	0	0	3

Áp dụng thuật toán tô màu để thực hiện.

Ví dụ 2.7: Học kì II năm 2009 -2010, Phòng ĐT muốn tổ chức thi các môn A, B, C, D, E, F, G, H, I biết rằng các môn sau không được thi chung một buổi. ABC, AE, BCD, BHI, EFG, EI, GHI. Hãy sắp xếp lịch thi sao cho số buổi thi cần sắp là ít nhất.

Hướng dẫn:

- Xác định đồ thị:
  - Đỉnh: các môn thi.
  - Cung: nối giữa hai môn không được thi chung buổi.

- Ta có 9 đỉnh: a, b, c, d, e, f, g, h, i. Ma trận quan hệ (hay đồ thị quan hệ như sau)

	A	B	C	D	E	F	G	H	I	Bậc
A	0	1	1	0	1	0	0	0	0	3
B	1	0	1	1	0	0	0	1	1	5
C	1	1	0	1	0	0	0	0	0	3
D	0	1	1	0	0	0	0	0	0	2
E	1	0	0	0	0	1	1	0	1	4
F	0	0	0	0	1	0	1	0	0	2
G	0	0	0	0	1	1	0	1	1	4
H	0	1	0	0	0	0	1	0	1	3
I	0	1	0	0	1	0	1	1	0	4

Áp dụng thuật toán tô màu để thực hiện.

2.3 THUẬT GIẢI LAI GHÉP

Thuật toán di truyền là thuật toán tối ưu ngẫu nhiên dựa trên cơ chế chọn lọc tự nhiên và tiến hóa di truyền. Nguyên lý cơ bản của thuật toán di truyền đã được Holland giới thiệu vào năm 1962. Cơ sở toán học đã được phát triển từ cuối những năm 1960 và đã được giới thiệu trong quyển sách đầu tiên của Holland, Adaptive in Natural and Artificial Systems. Thuật toán di truyền được ứng dụng đầu tiên trong hai lĩnh vực chính: tối ưu hóa và học tập của máy. Trong lĩnh vực tối ưu hóa thuật toán di truyền được phát triển nhanh chóng và ứng dụng trong nhiều lĩnh vực khác nhau như tối ưu hàm, xử lý ảnh, bài toán hành trình người bán hàng, nhận dạng hệ thống và điều khiển. Thuật toán di truyền cũng như các thuật toán tiến hóa nói chung, hình thành dựa trên quan niệm cho rằng, quá trình tiến hóa tự nhiên là quá trình hoàn hảo nhất, hợp lý nhất và tự nó đã mang tính tối ưu. Quan niệm này có thể xem như một tiên dề dúng, không chứng minh được, nhưng phù hợp với thực tế khách quan. Quá trình tiến hóa thể hiện tính tối ưu ở chỗ, thế hệ sau bao giờ cũng tốt hơn (phát triển hơn, hoàn thiện hơn) thế hệ trước bởi tính kế thừa và dấu tranh sinh tồn.

2.3.1 Cơ sở lý thuyết

Thuật toán di truyền gồm có bốn quy luật cơ bản là lai ghép, đột biến, sinh sản và chọn lọc tự nhiên như sau:

- **Quá trình lai ghép (phép lai):** quá trình này diễn ra bằng cách ghép một hay nhiều đoạn gen từ hai nhiễm sắc thể cha-mẹ để hình thành nhiễm sắc thể mới mang đặc tính của cả cha lẫn mẹ. Phép lai này có thể mô tả như sau: Chọn ngẫu nhiên hai hay nhiều cá thể trong quần thể. Giả sử chuỗi nhiễm sắc thể của cha và mẹ đều có chiều dài là m. Tìm điểm lai bằng cách tạo ngẫu nhiên một con số từ 1 đến m - 1. Như vậy, điểm lai này sẽ chia hai chuỗi nhiễm sắc thể cha-mẹ thành hai nhóm nhiễm sắc thể con là m1 và m2. Hai chuỗi nhiễm sắc thể con lúc này sẽ là m11+m22 và m21+m12. Đưa hai chuỗi nhiễm sắc thể con vào quần thể để tiếp tục tham gia quá trình tiến hóa.
- - **Quá trình đột biến (phép đột biến)**: quá trình tiến hóa được gọi là quá trình đột biến khi một hoặc một số tính trạng của con không được thừa hưởng từ hai chuỗi nhiễm sắc thể cha-mẹ. Phép đột biến xảy ra với xác suất thấp hơn rất nhiều lần so với xác suất xảy ra phép lai. Phép đột biến có thể mô tả như sau: Chọn ngẫu nhiên một số k từ khoảng $1 \geq k \geq m$. Thay đổi giá trị của gen thứ k. Đưa nhiễm sắc thể con vào quần thể để tham gia quá trình tiến hóa tiếp theo.
- **Quá trình sinh sản và chọn lọc (phép tái sinh và phép chọn)**

Phép tái sinh: là quá trình các cá thể được sao chép dựa trên độ thích nghi của nó. Độ thích nghi là một hàm được gán các giá trị thực cho các cá thể trong quần thể của nó. Phép tái sinh có thể mô phỏng như sau: Tính độ thích nghi của từng cá thể trong quần thể, lập bảng cộng dồn các giá trị thích nghi đó (theo thứ tự gán cho từng cá thể) ta được tổng độ thích nghi. Giả sử quần thể có n cá thể. Gọi độ thích nghi của cá thể thứ i là Fi, tổng dồn thứ i là Ft. Tổng độ thích nghi là Fm. Tạo số ngẫu nhiên F có giá trị trong đoạn từ 0 đến Fm Chọn cá thể k đầu tiên thỏa mãn $F \geq Ft$ đưa vào quần thể của thế hệ mới.

Phép chọn: là quá trình loại bỏ các cá thể xấu và để lại những cá thể tốt. Phép chọn được mô tả như sau: Sắp xếp quần thể theo thứ tự độ thích nghi giảm dần Loại bỏ các cá thể cuối dãy, chỉ để lại n cá thể tốt nhất.

2.3.2 Cấu trúc thuật toán di truyền tổng quát

Thuật toán di truyền bao gồm các bước sau:

Bước 1: Khởi tạo quần thể các nhiễm sắc thể.

Bước 2: Xác định giá trị thích nghi của từng nhiễm sắc thể.

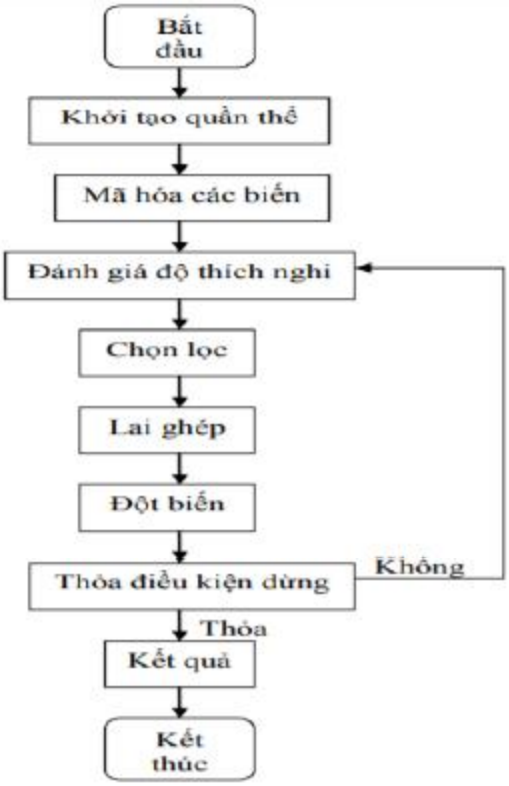
Bước 3: Sao chép lại các nhiễm sắc thể dựa vào giá trị thích nghi của chúng và tạo ra những nhiễm sắc thể mới bằng các phép toán di truyền.

Bước 4: Loại bỏ những thành viên không thích nghi trong quần thể.

Bước 5: Chèn những nhiễm sắc thể mới vào quần thể để hình thành một quần thể mới.

Bước 6: Nếu mục tiêu tìm kiếm đạt được thì dừng lại, nếu không trở lại bước 3.

Sơ đồ thuật toán:



Cấu trúc thuật giải di truyền tổng quát:

Bắt đầu

t =0;

Khởi tạo P(t)

Tính độ thích nghi cho các cá thể thuộc P(t);

Khi (điều kiện dừng chưa thỏa) lặp:

t = t + 1;

Chọn lọc P(t)

Lai P(t)

Đột biến P(t)

Hết lặp

Kết thúc

2.3.3 Các công thức của thuật giải di truyền

Tính độ thích nghi eval(v_i) của mỗi nhiễm sắc thể v_i (i =1... kích thước quần thể):

$$eval(v_i) = \frac{f(v_i)}{\sum_{i=1}^{kichthuocquanthe} f(v_i)} \text{ với } f(v_i) \text{ là hàm mục tiêu}$$

Tìm tổng giá trị thích nghi quần thể:

$$F = \sum_{i=1}^{kichthuocquanthe} eval(v_i)$$

Tính xác suất chọn pi cho mỗi nhiễm sắc thể v_i:

$$p_i = \frac{eval(v_i)}{\sum_{i-1}^{kichthuocquanthe} eval(v_i)}$$

Tính xác suất tích lũy qi cho mỗi nhiễm sắc thể:

$$q_i = \sum_{j-1}^{i} p_i$$

Tiến trình chọn lọc được thực hiện bằng cách quay bánh xe rulet kích thước quần thể lần. Mỗi lần chọn ra một nhiễm sắc thể từ quần thể hiện hành vào quần thể mới theo cách sau: Phát sinh một số ngẫu nhiên r trong khoảng [0, 1].

Nếu $r < q_1$ thì chọn nhiễm sắc thể v_1, ngược lại chọn nhiễm sắc thể v_i ($2 \le i \le$ kích thước quần thể) sao cho $q_{i-1} < r \le q_i$

2.3.4 Bài toán minh họa

Tìm đáp số cho phương trình $X^2 = 64$. Đây là một bài toán đơn giản để giúp ta có thể hiểu rõ hơn các bước của thuật toán di truyền.

Giải bài toán di truyền theo các bước sau:

Bước 1: Chúng ta sử dụng hệ nhị phân để xây dựng mô hình bài toán.Ta dùng 4 bit nhị phân để mã hóa cho các đáp số của bài toán.Gỉa sử ta không biết đáp số của bài toán, ta sẽ chọn 4 số trong các đáp số có thể có và ký hiệu cho các đáp số đó.

Bảng chọn lựa:

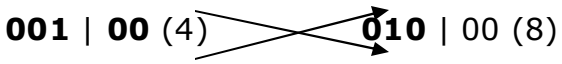
Thứ tự	Nhị phân	Thập phân
1	**0** 0100	4
2	**1** 0101	21
3	**0** 1010	10
4	**1** 1000	24

Bước 2: Tìm hàm số thích nghi và tính hệ số thích nghi cho từng đáp số. Ta chọn hàm số thích nghi sau: $f(X) = 1000 – (X_2 – 64)$. Vậy, đáp số nào có hệ số thích nghi f gần bằng 1000 nhất thì đó là đáp số. Khảo sát kết quả tính được:

Thứ tự	Nhị phân	Thập phân (X)	X_2 - 64	Hệ số thích nghi f(x)
1	**0** 0100	4	-48	952
2	**1** 0101	21	377	623
3	**0** 1010	10	36	964
4	**1** 1000	24	512	488

Bước 3: Ta thấy, hệ số thích nghi của các đáp số vẫn còn cách xa 1000. Do đó, cần tạo ra các đáp số mới bằng cách biến hóa các đáp số cũ. Ta thấy, số 4 và 10 có hệ số thích nghi cao hơn nên được chọn để tạo sinh và biến hóa. Đồng thời số 21 và 24 có hệ số thích nghi thấp sẽ bị loại.

Gỉa sử ta lai ghép hai số 4 và 10 theo hình sau:



010 | 10 (10) 001 | **10** (6)

Bước 4: Tính hệ số thích nghi cho quần thể mới

Thứ tự	Nhị phân	Thập phân (X)	X_2 - 64	Hệ số thích nghi f(x)
1	**0** 0100	4	-48	952
2	**0** 0101	10	36	964
3	**0** 1010	8	0	1000
4	**0** 1000	6	28	968

Bước 5: May mắn chúng ta đã tìm được kết quả là X =8 với hệ số thích nghi cao nhất là 1000.

Vậy kết quả của bài toán là X = 8.

Thuật toán di truyền đã chứng tỏ tính hữu ích của nó khi được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau của cuộc sống.

Trong lĩnh vực điều khiển tự động, thuật toán di truyền có thể được sử dụng để xác định thong số tối ưu cho các bộ điều khiển. Thông số bộ điều khiển được mã hóa thành các nhiễm sắc thể, thông qua mô phỏng, các nhiễm sắc thể này được đánh giá và lựa chọn thong qua mức độ thích nghi của chúng (cũng chính là các chỉ tiêu chất lượng của hệ thống). Kết quả của thuật toán sẽ cho một bộ điều khiển có thong số tốt nhất.

Trong y học, cấu trúc của các chất hóa học được mã hóa thành các nhiễm sắc thể hoặc đồ thị. Thuật toán di truyền sẽ lai ghép, lựa chọn để tạo ra các nhiễm sắc Hỗ trợ ôn tập thể mới (các chất hóa học mới). Và trong thực tế đã có rất nhiều loại thuốc mới được tạo ra như vậy.

BÀI 3: CÁC PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

Học xong bài này sinh viên có thể nắm được:

- *Các khái niệm liên quan về bài toán tìm kiếm, không gian trạng thái.*
- *Các chiến lược tìm kiếm mù (blind search)*
- *Các kỹ thuật tìm kiếm lời giải theo kinh nghiệm (heuristic).*

3.1 TÌM KIẾM CƠ BẢN

3.1.1 Phát biểu bài toán

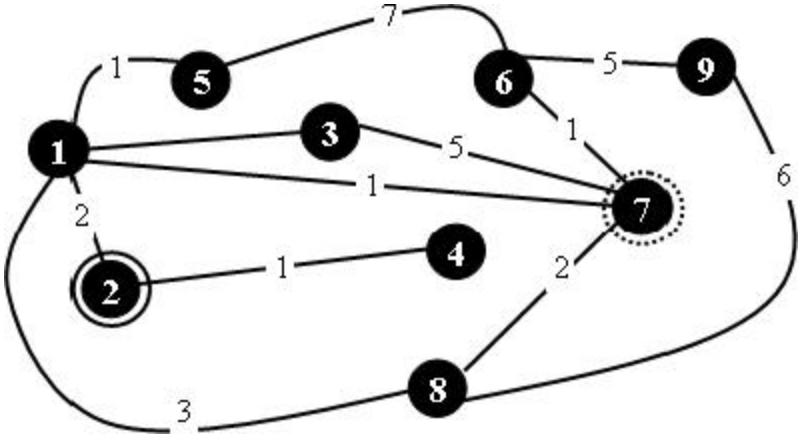
Vấn đề tìm kiếm có thể hiểu là tìm một đối tượng thỏa mãn một số yêu cầu nào đó trong một tập hợp rộng lớn các đối tượng. Có rất nhiều bài toán được quy về vấn đề tìm kiếm như các trò chơi (cờ vua, cờ caro, ...) có thể xem như vấn đề tìm kiếm. Trong số rất nhiều nước đi được phép thực hiện, ta phải tìm ra các nước đi dẫn tới kết quả là người chơi thắng.

Một cách tổng quát, nhiều bài toán tìm kiếm phức tạp đều có dạng "Tìm đường đi trong đồ thị" hay nói một cách đầy đủ hơn là "xuất phát từ một đỉnh của một đồ thị, tìm đường đi hiệu quả nhất đến một đỉnh nào đó".

Cho trước 2 trạng thái T_0 và T_G, hãy xây dựng chuỗi trạng thái $T_0, T_1, T_2, T_3, ..., T_n = T_G$ sao cho Σ_1^n cost (T_{i-1}, T_i) thoả mãn một điều kiện cho trước (thường là nhỏ nhất).

Trong đó T_i thuộc tập hợp không gian trạng thái S bao gồm tất cả các trạng thái có thể có của bài toán và cost (T_{i-1}, T_i) là chi phí để biến đổi từ trạng thái T_{i-1} sang trạng thái T_i. Từ một trạng thái T_i ta có nhiều cách để biến đổi sang trạng thái T_{i+1}. Khi nói đến một biến đổi cụ thể từ T_{i-1} sang T_i ta sẽ dùng thuật ngữ hướng đi.

Đa số các bài toán thuộc dạng mà chúng ta đang mô tả đều có thể biểu diễn dưới dạng đồ thị. Trong đó một trạng thái là một đỉnh của đồ thị. Tập hợp *S* bao gồm tất cả các trạng thái chính là tập hợp bao gồm tất cả đỉnh của đồ thị. Việc biến đổi từ trạng thái T_{i-1} sang T_i là việc đi từ đỉnh đại diện cho T_{i-1} sang đỉnh đại diện cho T_i theo cung nối giữa hai đỉnh này.



Hình 3.1: Mô hình tổng quát của bài toán phải giải quyết bằng phương pháp tìm kiếm lời giải

Trong quá trình nghiên cứu giải quyết các bài toán, người ta đã đưa ra những nhận xét như sau:

- Có nhiều bài toán cho đến nay vẫn chưa tìm ra một cách giải theo kiểu thuật toán và cũng không biết là có tồn tại thuật toán hay không.
- Có nhiều bài toán đã có thuật toán để giải nhưng không chấp nhận được vì thời gian giải theo thuật toán đó quá lớn hoặc các điều kiện cho thuật toán khó đáp ứng.
- Có những bài toán được giải theo những cách giải vi phạm thuật toán nhưng vẫn chấp nhận được.

Trong thực tiễn có nhiều trường hợp người ta chấp nhận các cách giải thường cho kết quả tốt (không phải lúc nào cũng tốt) nhưng ít phức tạp và hiệu quả. Nếu giải một bài toán bằng thuật toán tối ưu đòi hỏi máy tính thực hiện nhiều năm thì chúng ta có thể sẵn lòng chấp nhận một giải pháp gần tối ưu mà chỉ cần máy tính chạy trong vài ngày hoặc vài giờ.

Các cách giải chấp nhận được nhưng không hoàn toàn đáp ứng đầy đủ các tiêu chuẩn của thuật toán thường được gọi là các thuật giải. Khái niệm mở rộng này của

thuật toán đã mở cửa cho chúng ta trong việc tìm kiếm phương pháp để giải quyết các bài toán được đặt ra.

Một trong những thuật giải thường được đề cập đến và sử dụng trong khoa học trí tuệ nhân tạo là các cách giải theo kinh nghiệm (Heuristic). Tìm kiếm theo kinh nghiệm (heuristic) là dựa vào kinh nghiệm và sự hiểu biết của chúng ta về vấn đề cần giải quyết để xây dựng nên hàm đánh giá hướng dẫn sự tìm kiếm. Trong tìm kiếm mù (blind search), chúng ta không có hiểu biết gì về các đối tượng để hướng dẫn tìm kiếm mà chỉ đơn thuần là xem xét theo một hệ thống nào đó tất cả các đối tượng để phát hiện ra đối tượng cần tìm.

Trong bài này, chúng ta sẽ nghiên cứu các chiến lược tìm kiếm mù (blind search) như tìm kiếm theo bề rộng (breadth-first search) và tìm kiếm theo độ sâu (depth-first search) và các chiến lược tìm kiếm theo kinh nghiệm.

3.1.2 Không gian tìm kiếm

Để giải quyết một vấn đề nào đó bằng phương pháp tìm kiếm, đầu tiên chúng ta phải xác định được không gian tìm kiếm. Cơ sở của các bài toán tìm kiếm bao gồm: trạng thái đầu (trạng thái xuất phát), các hành động biến đổi trạng thái và trạng thái kết thúc (trạng thái đích). Vấn đề đặt ra là xác định dãy các trạng thái hợp lý để sao cho từ trạng thái xuất phát có thể đến được trạng thái đích.

Quá trình thực hiện tìm kiếm, trước hết phải tìm cách biểu diễn bài toán trong không gian tìm kiếm. Không gian tìm kiếm bao gồm tất cả các đối tượng mà chúng ta cần quan tâm, có thể là không gian liên tục, không gian các vector hoặc không gian các đối tượng rời rạc.

Không gian trạng thái của một bài toán là tập các trạng thái có thể có của bài toán, các trạng thái của bài toán đó đạt được bằng cách thực hiện chuỗi các hành động xuất phát từ trạng thái ban đầu. Để biểu diễn một vấn đề trong không gian trạng thái, ta cần xác định các yếu tố sau:

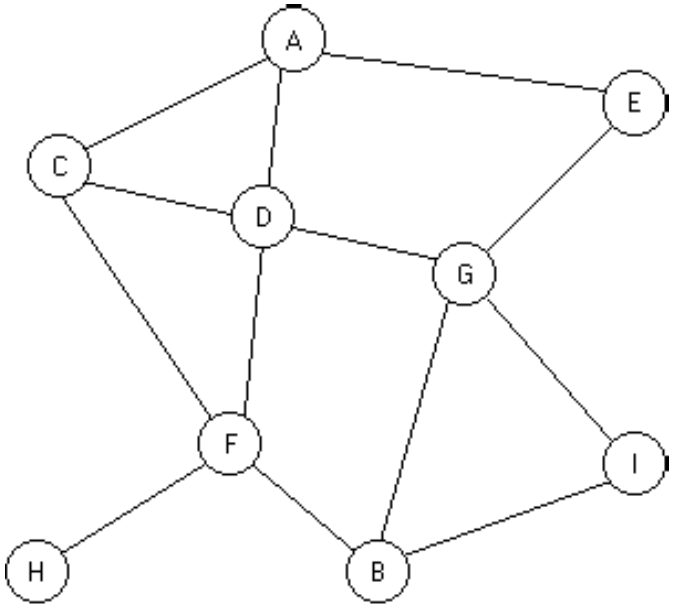
- Trạng thái ban đầu.
- Tập hợp các toán tử, trong đó mỗi toán tử mô tả một hành động hoặc một phép biến đổi có thể đưa một trạng thái tới một trạng thái khác.

- Tập trạng thái kết thúc.

Gọi U là không gian trạng thái, T_0 là trạng thái ban đầu ($T_0 \in U$), T_i là các trạng thái kết thúc, $T_i \subseteq U$. Mỗi toán tử R xem như một ánh xạ $R: U \to U$.

Chúng ta có thể biểu diễn không gian trạng thái bằng đồ thị, trong đó mỗi đỉnh của đồ thị tương ứng với một trạng thái. Nếu có toán tử R biến đổi trạng thái u thành trạng thái v, thì có cung gán nhãn R đi từ đỉnh u tới đỉnh v. Khi đó một đường đi trong không gian trạng thái sẽ là một đường đi trong đồ thị.

Ví dụ 3.1: Một du khách có trong tay bản đồ mạng lưới giao thông nối các thành phố của một quốc gia. Giả sử du khách đang ở thành phố A và muốn tìm đường đi tới thăm thành phố B. Các thành phố có trong các bản đồ là các trạng thái (đã thăm hoặc chưa thăm), thành phố A là trạng thái ban đầu, B là trạng thái kết thúc. Khi đã đi đến một thành phố nào đó (thành phố D), du khách có thể đi theo các con đường để đi tới các thành phố khác. Các con đường để đi đến các thành phố sẽ được biểu diễn bởi các toán tử. Một toán tử biến đổi một trạng thái thành một trạng thái khác. Ở thành phố D, du khách có thể đi tới thành phố C, F, G. Như vậy sẽ có ba toán tử dẫn trạng thái D tới các trạng thái C, F và G. Vấn đề của du khách bây giờ sẽ là tìm một dãy toán tử để đưa trạng thái ban đầu A tới trạng thái kết thúc B.

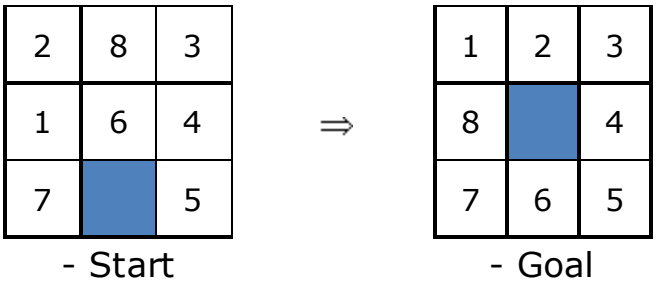


Hình 3.2: Tìm đường đi từ thành phố A đến thành phố B

Ví dụ 3.2: Trong trò chơi cờ vua, mỗi cách bố trí các quân trên bàn cờ là một trạng thái. Trạng thái ban đầu là sự sắp xếp các quân lúc bắt đầu cuộc chơi. Mỗi nước đi hợp lệ là một toán tử, nó biến đổi một tình huống trên bàn cờ thành một tình huống khác.

Trong bài toán tìm đường của du khách, chỉ có một trạng thái đích, đó là thành phố B. Trong nhiều bài toán có thể có nhiều trạng thái đích và ta không thể xác định trước được các trạng thái đích. Nói chung trong phần lớn các vấn đề hay, ta chỉ có thể mô tả các trạng thái đích là các trạng thái thỏa mãn một số điều kiện nào đó.

*Ví dụ 3.3***:** Xét ví dụ về các không gian trạng thái được xây dựng cho Bài toán 8 số, chúng ta có ma trận 3 x 3 ô và các giá trị đánh từ 1 đến 8 được xếp vào tám ô, còn lại một ô trống, như trong Hình 3.3(a) bên trái. Trong bài toán này, chúng ta có thể chuyển dịch các giá trị ở cạnh ô trống vào ô còn trống. Vấn đề là cần tìm ra một dãy các chuyển dịch để biến đổi tình huống ban đầu thành một tình huống xác định nào đó giả sử như Hình 3.3 (b) bên phải.



Hình 3.3: Minh hoạ bài toán 8 số (taci)

Tương ứng với các quy tắc chuyển dịch các số, chúng ta có bốn toán tử: ***up*** (đẩy lên trên), ***down*** (đẩy xuống dưới), ***left*** (đẩy sang trái), ***right*** (đẩy sang phải). Các toán tử này chỉ là các toán tử bộ phận, từ trạng thái ban đầu chúng ta chỉ có thể áp dụng các toán tử ***down, left, right***.

Trong ví dụ trên, việc tìm ra một biểu diễn thích hợp để mô tả các trạng thái của vấn đề là khá dễ dàng. Tuy nhiên, có nhiều bài toán mà việc tìm kiếm một biểu diễn thích hợp cho các trạng thái của vấn đề là hoàn toàn không đơn giản. Việc tìm ra dạng biểu diễn tốt cho các trạng thái đóng vai trò hết sức quan trọng trong quá trình giải quyết một vấn đề. Có thể nói rằng, nếu tìm được dạng biểu diễn tốt cho các trạng thái của vấn đề, thì vấn đề hầu như đã được giải quyết.

3.1.3 Các chiến lược tìm kiếm

Để giải quyết một vấn đề bằng tìm kiếm trong không gian trạng thái, đầu tiên ta xác định:

- Trạng thái ban đầu.
- Tập các toán tử.
- Tập T_i các trạng thái kết thúc. (T_i có thể không được xác định cụ thể gồm các trạng thái nào mà chỉ được chỉ định bởi một số điều kiện nào đó).

Khi chúng ta biểu diễn một vấn đề cần giải quyết thông qua các trạng thái và các toán tử thì việc tìm lời giải của vấn đề được quy về việc tìm đường đi từ trạng thái ban đầu tới một trạng thái kết thúc nào đó.

Các chiến lược tìm kiếm thành hai loại:

- Tìm kiếm mù: chiến lược tìm kiếm này không có hiểu biết gì về các đối tượng hay một hướng dẫn tìm kiếm. Chúng ta chỉ phát triển các trạng thái ban đầu cho tới khi gặp một trạng thái đích nào đó. Tìm kiếm mù bao gồm hai kỹ thuật là tìm kiếm theo chiều rộng (DFS -Depth First Search) và tìm kiếm theo chiều sâu (BFS - Breath First Search).
- Tìm kiếm theo kinh nghiệm (tìm kiếm heuristic): dựa vào kinh nghiệm và hiểu biết về vấn đề cần giải quyết để xây dựng hàm đánh giá hướng dẫn sự tìm kiếm. Trong quá trình phát triển các trạng thái, chúng ta sẽ chọn ra một trạng thái được đánh giá là tốt nhất để phát triển, do đó tốc độ tìm kiếm sẽ nhanh hơn.

3.1.4 Các giải thuật cơ bản

3.1.4.1 Tìm kiếm chiều sâu (Depth-First Search)

Ý tưởng: Bắt đầu mở rộng từ nút gốc, sau đó mở rộng một trong các nút ở mức sâu nhất của cây. Nếu tìm đến một điểm cụt (tức là đến nút không phải nút đích và lại không mở rộng được nữa), lùi về từng đỉnh để tìm và duyệt những nhánh tiếp theo. Quá trình duyệt chỉ dừng lại khi tìm thấy đỉnh cần tìm hoặc tất cả đỉnh đều đã được duyệt qua. Tại mỗi bước, trạng thái chọn để phát triển là trạng thái được sinh ra sau các trạng thái chờ phát triển khác.

Thuật toán:

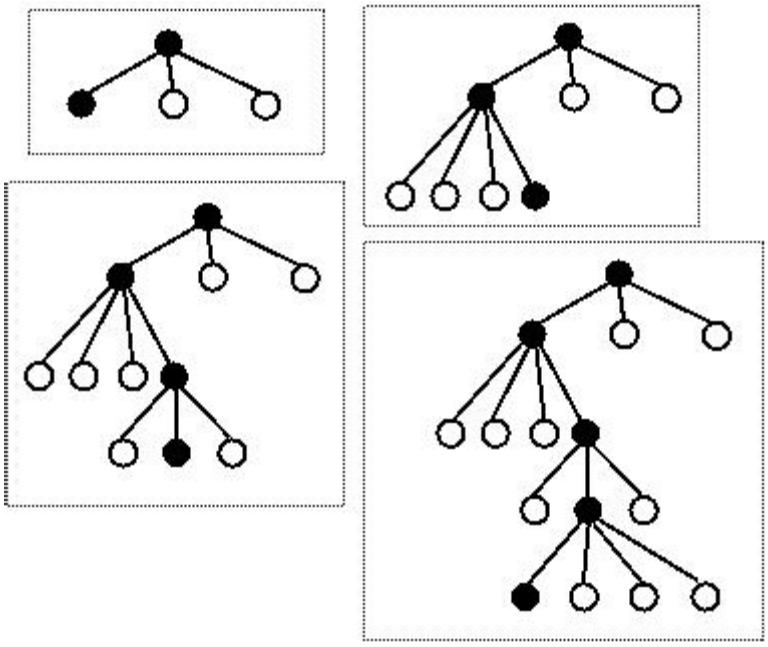
```
Procedure Deepth_first_Search ()
{
    Khởi tạo danh sách L chỉ chứa trạng thái ban đầu;
    do
    {
        if (danh sách L rỗng)
            {
                thông báo tìm kiếm thất bại;
                dừng;
            }
     Loại trạng thái u đầu danh sách L;
        if (u là trạng thái kết thúc)
        {
            thông báo tìm kiếm thành công;
            stop;
        }
        for (mỗi trạng thái v kề u)
        {
            Đặt v vào đầu danh sách L;
            father(v) ← u;
        }
   } //end do
}
```

Danh sách L là một *ngăn xếp* dùng để lưu các trạng thái đã được sinh ra và chờ được phát triển. Hàm *father* lưu lại cha của mỗi đỉnh trên đường đi, *father(v) = u* nếu cha của đỉnh *v* là *u*.

Nếu bài toán có nghiệm (tồn tại đường đi từ trạng thái ban đầu tới trạng thái đích), thì thuật toán tìm kiếm theo chiều rộng sẽ tìm ra nghiệm, đồng thời đường đi tìm được sẽ là ngắn nhất. Trong trường hợp bài toán vô nghiệm và không gian trạng thái hữu hạn, thuật toán sẽ dừng và cho thông báo vô nghiệm.

Trong tìm kiếm theo chiều sâu, tại trạng thái (đỉnh) hiện hành, chúng ta chọn một trạng thái kế tiếp (trong tập các trạng thái có thể biến đổi thành từ trạng thái hiện tại) làm trạng thái hiện hành cho đến lúc trạng thái hiện hành là trạng thái đích.

Trong trường hợp tại trạng thái hiện hành, ta không thể biến đổi thành trạng thái kế tiếp thì ta sẽ quay lui (back-tracking) lại trạng thái trước trạng thái hiện hành (trạng thái biến đổi thành trạng thái hiện hành) để chọn đường khác. Nếu ở trạng thái trước này mà cũng không thể biến đổi được nữa thì ta quay lui lại trạng thái trước nữa và cứ thế. Nếu đã quay lui đến trạng thái khởi đầu mà vẫn thất bại thì kết luận là không có lời giải.



Hình 3.4: Tìm kiếm theo chiều sâu

3.1.4.2 Tìm kiếm chiều rộng (Breath-First Search)

Ý tưởng: Bắt đầu mở rộng từ nút gốc, sau đó lần lượt mở rộng các nút được sinh ra từ nút gốc, tiếp đến những nút kế tiếp của các nút này, và cứ như vậy cho đến khi tìm thấy một nút đích nào đó hoặc không còn nút nào được sinh ra.

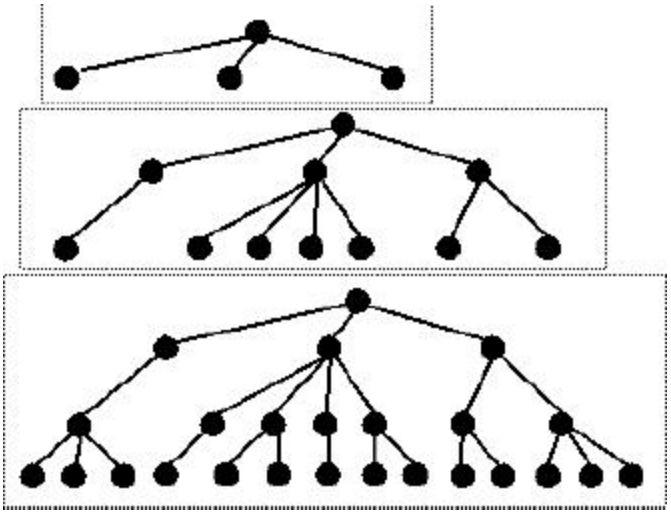
Thuật toán:

```
Procedure Breath_first_Search ()
{
    Khởi tạo danh sách L chỉ chứa trạng thái ban đầu;
    do
    {
        if (danh sách L rỗng)
            {
                thông báo tìm kiếm thất bại;
```

```
                    dừng;
                }
        Loại trạng thái u đầu danh sách L;
            if (u là trạng thái kết thúc)
            {
                thông báo tìm kiếm thành công;
                stop;
            }
            for (mỗi trạng thái v kề u)
            {
                Đặt v vào đầu danh sách L;
                father(v) ← u;
            }
    } //end do
}
```

Danh sách L là một *hàng đợi* dùng để lưu các trạng thái đã được sinh ra và chờ được phát triển. Hàm *father* lưu lại cha của mỗi đỉnh trên đường đi, father(v) = u nếu cha của đỉnh v là u.

Ngược lại với tìm kiếm theo kiểu chiều sâu, tìm kiếm chiều rộng mang hình ảnh của vết dầu loang. Từ trạng thái ban đầu, ta xây dựng tập hợp S bao gồm các trạng thái kế tiếp (mà từ trạng thái ban đầu có thể biến đổi thành). Sau đó, *ứng với mỗi* trạng thái T_k trong tập S, ta xây dựng tập S_k bao gồm các trạng thái kế tiếp của T_k rồi lần lượt bổ sung các S_k vào S. Quá trình này cứ lặp lại cho đến lúc S có chứa trạng thái kết thúc hoặc S không thay đổi sau khi đã bổ sung tất cả S_k.



Hình 3.5: Tìm kiếm theo chiều rộng

Tìm kiếm chiều sâu và tìm kiếm chiều rộng đều là các phương pháp tìm kiếm có hệ thống và chắc chắn tìm ra lời giải. Tuy nhiên, do bản chất là vét cạn nên với những bài toán có không gian lớn thì ta không thể dùng hai chiến lược này được. Hơn nữa, hai chiến lược này đều có tính chất "mù quáng" vì chúng không chú ý đến những thông tin (tri thức) ở trạng thái hiện thời và thông tin về đích cần đạt tới cùng mối quan hệ giữa chúng.

3.2 TÌM KIẾM THEO KINH NGHIỆM (HEURISTIC)

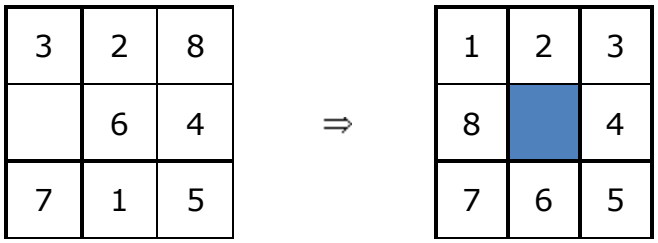
Các kỹ thuật tìm kiếm mù rất kém hiệu quả và trong nhiều trường hợp không thể áp dụng được. Trong phần này, chúng ta sẽ nghiên cứu các phương pháp tìm kiếm kinh nghiệm (tìm kiếm heuristic). Thuật toán heuristic gồm hai phần: Hàm đánh giá heuristic và thuật toán để sử dụng nó trong tìm kiếm không gian trạng thái.

3.2.1 Hàm Heuristic là gì?

Hàm Heuristic (hay hàm đánh giá, ký hiệu là h) đơn giản chỉ là một ***ước lượng về khả năng dẫn đến lời giải*** tính từ trạng thái đó (*khoảng cách* giữa trạng thái hiện tại và trạng thái đích). Thông thường, giá trị này là không thể tính toán được mà ta chỉ dùng nó như một cơ sở để suy luận về mặt lý thuyết. Hàm ***h***, luôn trả ra kết quả là một số không âm.

Ví dụ 3.4: Bài toán 8 số, chúng ta có thể đưa ra hai cách xây dựng hàm đánh giá, một số hàm đánh giá có thể được xác định như sau:

Hàm $h_1(u)$: Với mỗi trạng thái u thì $h_1(u)$ là giá trị không nằm đúng vị trí của nó trong trạng thái đích. Theo trạng thái đích thì $h_1(u) = 4$, vì giá trị không đúng vị trí là 3, 8, 6 và 1.



Hình 3.6: Các trạng thái u

Hàm $h_2(u)$ là tổng khoảng cách giữa vị trí của các giá trị trong trạng thái u và vị trí của nó trong trạng thái đích. Khoảng cách ở đây được hiểu là số ít nhất các dịch chuyển theo hàng hoặc cột để đưa một giá trị tới vị trí của nó trong trạng thái đích. Với trạng thái u và trạng thái đích như Hình 3.6, ta có: $\boldsymbol{h_2(u) = 2 + 3 + 1 + 3 = 9}$

Vì giá trị 3 cần ít nhất 2 dịch chuyển, giá trị 8 cần ít nhất 3 dịch chuyển, giá trị 6 cần ít nhất 1 dịch chuyển và giá trị 1 cần ít nhất 3 dịch chuyển.

Hai chiến lược tìm kiếm kinh nghiệm quan trọng nhất là tìm kiếm tốt nhất - đầu tiên (best-first search) và tìm kiếm leo đồi (hill-climbing search). Có thể xác định các chiến lược này như sau:

Tìm kiếm tốt nhất đầu tiên = Tìm kiếm theo bề rộng + Hàm đánh giá

Tìm kiếm leo đồi = Tìm kiếm theo độ sâu + Hàm đánh giá

3.2.2 Tìm kiếm leo đồi (Hill climbing – Pearl 1984)

Tìm kiếm leo đồi thực chất chỉ là một trường hợp đặc biệt của tìm kiếm theo chiều sâu nhưng không thể quay lui. Trong tìm kiếm leo đồi, việc lựa chọn trạng thái tiếp theo được quyết định dựa trên một hàm Heuristic.

Chiến lược leo đồi phát triển trạng thái con tốt nhất sẽ được chọn cho bước tiếp theo, không lưu giữ lại bất kỳ thông tin nào về các nút anh em lẫn cha mẹ của nó. Quá trình tìm kiếm sẽ dừng lại khi tiếp cận trạng thái tốt hơn so với mọi trạng thái con của nó. Vì không ghi lại thông tin của quá trình đã xảy ra nên thuật toán này không thể phục hồi lại từ những thất bại trong chiến lược của nó.

Thuật giải Leo đồi (Hill_Climbing_Search)

1. Nếu trạng thái bắt đầu cũng là trạng thái đích thì thoát và báo là đã tìm được lời giải. Ngược lại, đặt trạng thái hiện hành T_i là trạng thái khởi đầu T_0

2. Lặp lại cho đến khi đạt đến trạng thái kết thúc hoặc cho đến khi không tồn tại một trạng thái tiếp theo hợp lệ T_k của trạng thái hiện hành:

2a. Đặt T_k là một trạng thái tiếp theo hợp lệ của trạng thái hiện hành T_i.

2b. Đánh giá trạng thái T_k mới:

2b.1. Nếu là trạng thái kết thúc thì trả về trị này và thoát.

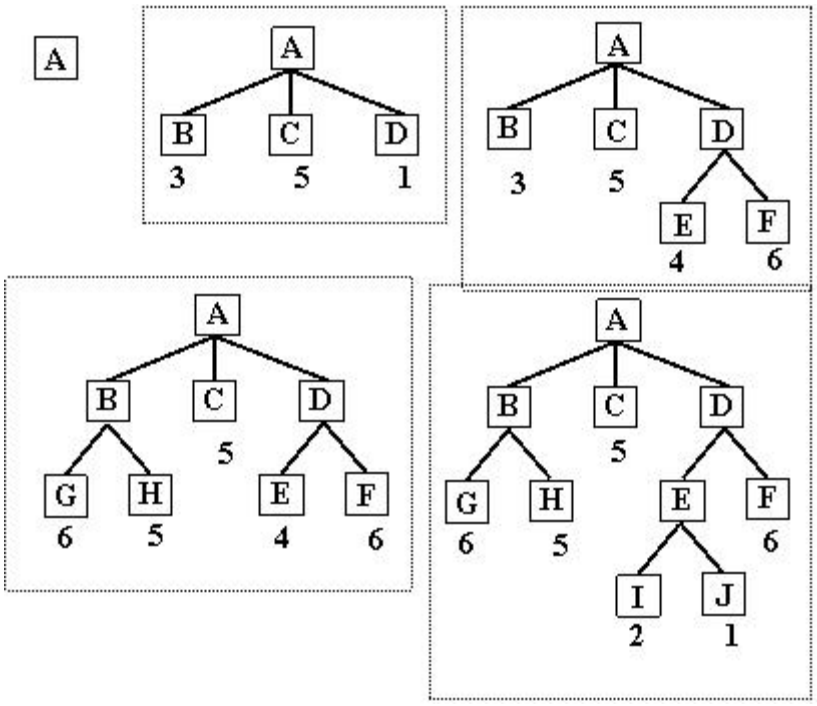
2b.2. Nếu không phải là trạng thái kết thúc nhưng **tốt** hơn trạng thái hiện hành thì cập nhật nó thành trạng thái hiện hành.

2b.3. Nếu nó không tốt hơn trạng thái hiện hành thì tiếp tục vòng lặp.

Hạn chế chủ yếu của chiến lược leo đồi là có xu hướng rơi vào "một cực đại cục bộ". Khi đến được một trạng thái tốt hơn so với mọi trạng thái con của nó, thuật toán dừng lại. Nếu trạng thái này không phải là đích mà chỉ là một điểm cực đại cục bộ, thuật toán sẽ thất bại trong việc tìm lời giải. Như vậy hiệu quả hoạt động chỉ có thể được cải thiện trong một phạm vi giới hạn nào đó, nhưng trong toàn bộ không gian có thể không bao giờ đạt được sự tối ưu tổng thể.

3.2.3 Tìm kiếm tốt nhất đầu tiên (Best – first – search)

Tìm kiếm tốt nhất đầu tiên (best first search) là sự kết hợp của tìm kiếm theo chiều rộng với hàm đánh giá. Khác với phương pháp tìm kiếm theo chiều rộng, các nút không được phát triển lần lượt mà được lựa chọn dựa trên hàm đánh giá; đỉnh này có thể ở mức hiện tại hoặc ở các mức trên. Tại mỗi bước của chiến lược tìm kiếm BFS, tiến hành chọn trạng thái có khả năng cao nhất trong số các trạng thái đã được xét *cho đến thời điểm hiện tại*. Cách tiếp cận này sẽ ưu tiên đi vào những nhánh tìm kiếm có khả năng nhất, nếu càng đi sâu vào một hướng và phát hiện ra rằng hướng này xấu hơn cả những hướng mà ta chưa đi, thì ta sẽ không đi tiếp hướng hiện tại nữa mà chọn đi theo một hướng tốt nhất trong số những hướng chưa đi.



Hình 3.7: Minh họa thuật giải Best-**First Search**

Ví dụ 3.5: Khởi đầu, chỉ có một nút (trạng thái) A nên nó sẽ được mở rộng tạo ra 3 nút mới B, C và D. Các giá trị dưới nút là cho biết độ tốt của nút. Giá trị càng nhỏ, nút càng tốt. Do D là nút có khả năng nhất nên nó sẽ được mở rộng tiếp sau nút A và sinh ra 2 nút kế tiếp là E và F. Đến đây, ta lại thấy nút B có vẻ có khả năng nhất (trong các nút B, C, E, F) nên ta sẽ chọn mở rộng nút B và tạo ra 2 nút G và H. Nhưng lại một lần nữa, hai nút G, H này được đánh giá ít khả năng hơn **E**, vì thế sự chú ý lại trở về E. E được mở rộng và các nút được sinh ra từ E là I và J. Ở bước kế tiếp, J sẽ được mở rộng vì nó có khả năng nhất. Quá trình này tiếp tục cho đến khi tìm thấy một lời giải.

Trong BFS, tại một bước, cũng có một di chuyển được chọn nhưng những cái khác vẫn được giữ lại, để ta có thể trở lại xét sau đó khi trạng thái hiện tại trở nên kém khả năng hơn những trạng thái đã được lưu trữ. Hơn nữa, ta chọn trạng thái tốt nhất mà không quan tâm đến nó có *tốt hơn* hay không các trạng thái trước đó. Điều này tương phản với leo đồi vì leo đồi sẽ dừng nếu không có trạng thái tiếp theo nào tốt hơn trạng thái hiện hành.

Để cài đặt các thuật giải theo kiểu tìm kiếm BFS, người ta thường dùng 2 danh sách sau: **OPEN:** tập chứa các trạng thái đã được sinh ra nhưng chưa được xét đến, **OPEN** là một loại hàng đợi ưu tiên (priority queue) mà trong đó, phần tử có độ ưu tiên cao nhất là phần tử *tốt nhất*.

CLOSE: tập chứa các trạng thái đã được xét đến. Chúng ta cần lưu trữ những trạng thái này trong bộ nhớ để đề phòng trường hợp khi một trạng thái mới được tạo ra lại trùng với một trạng thái mà ta đã xét đến trước đó. Trong trường hợp không gian tìm kiếm có dạng cây thì không cần dùng danh sách này.

Thuật giải BFS (Best_First_Search)

1. Đặt **OPEN** chứa trạng thái khởi đầu.

2. Cho đến khi tìm được trạng thái đích hoặc không còn nút nào trong OPEN, thực hiện:

2.a. Chọn trạng thái **tốt nhất** T_{max} trong OPEN và xóa T_{max} khỏi OPEN

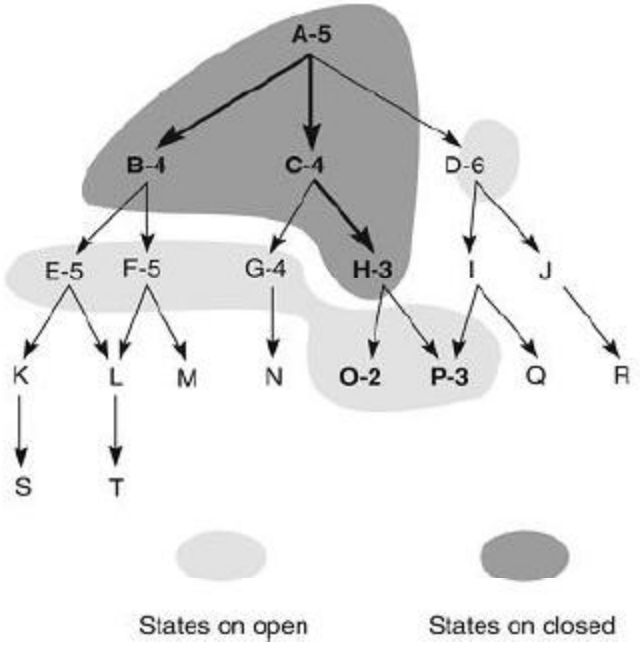
2.b. Nếu T_{max} là trạng thái kết thúc thì thoát.

2.c. Ngược lại, tạo ra các trạng thái kế tiếp T_k có thể có từ trạng thái T_{max}. Đối với mỗi trạng thái kế tiếp T_k thực hiện:

Tính f(T_k);

Thêm T_k vào OPEN

Xét đồ thị không gian tìm kiếm như Hình 3.8 (số cạnh mỗi nút cho biết giá trị ước lượng độ tốt của nút đó trong không gian, giá trị thấp nhất là tốt nhất). Giả sử nút đích cần tìm kiếm là P.



Hình 3.8: Đồ thị cho giải thuật tìm kiếm tốt nhất đầu tiên

Giống như các thuật toán tìm kiếm sâu và rộng, tìm kiếm tốt nhất cũng dùng các danh sách để lưu giữ trạng thái: danh sách *Open* chứa các nút được triển khai trong quá trình tìm kiếm và danh sách *Closed* chứa các nút đã xét. Một bước mới được bổ sung vào thuật toán là sắp xếp các trạng thái trong danh sách *Open* phù hợp với giá trị heuristic ước lượng "độ tốt" của chúng so với đích. Như vậy mỗi bước lặp của vòng lặp sẽ xem xét trạng thái "có hứa hẹn nhất" trong danh sách *Open* và loại bỏ trạng thái này ra khỏi *Open*. Nếu gặp trạng thái đích, thuật toán này sẽ cung cấp con đường lời giải đã dẫn đến đích đó. Nếu ngược lại, phần tử đầu tiên của *Open* không phải là đích, thuật toán sẽ áp dụng các luật phù hợp để phát sinh con cháu. Trường hợp một trạng thái con nào đó đã có sẵn trong *Open* hoặc *Closed*, thuật toán cũng sẽ kiểm tra

để chắc chắn rằng sẽ chọn được nút cung cấp con đường lời giải ngắn hơn. Các trạng thái lặp hai lần sẽ không được giữ lại. Nhờ cập nhật kịp thời nguồn gốc của các nút trong *Open* và *Closed* nên thuật toán này có nhiều khả năng tìm được đường đi ngắn nhất dẫn đến đích. Khi *Open* được duy trì dưới dạng một danh sách có sắp xếp, thường được tổ chức như là một hàng ưu tiên (Priority queue).

Dưới đây trình bày các bước áp dụng thuật toán tìm kiếm cho đồ thị trong Hình 3.8.

1. open = [A5]; closed = []

2. Đánh giá A5;

   open = [B4, C4, D6]; closed = [A5]

3. Đánh giá B4;

   open = [C4, E5, F5, D6]; closed = [B4, A5]

4. Đánh giá C4;

   open = [H3, G4, E5, F5, D6]; closed = [C4, B4, A5]

5. Đánh giá H3;

   open = [O2, P3, G4, E5, F5, D6]; closed = [H3, C4, B4, A5]

6. Đánh giá O2;

   open = [P3, G4, E5, F5, D6]; closed = [O2, H3, C4, B4, A5]

7. Đánh giá P3;

   Tìm được lời giải!

3.3 TÌM KIẾM VÀ TỐI ƯU HÓA

Vấn đề tìm kiếm là tìm đường đi từ trạng thái xuất phát đến trạng thái đích trong không gian trạng thái. Trong thực tế, thường người ta phải tính đến chi phí di chuyển các trạng thái, từ trạng thái u đến trạng thái *v*, biểu diễn bởi một số không âm. Giá trị này được gọi là trọng số của cung (*u*, *v*) trong đồ thị không gian trạng thái. Trọng số này được xác định tùy thuộc từng bài toán cụ thể.

Trong bài toán tìm đường đi trên bản đồ, trọng số của cung (*u*, *v*) là độ dài của đoạn đường từ địa điểm *u* đến địa điểm *v*. Độ dài đường đi được xác định bằng tổng

độ dài các cung trên đường đi. Mục tiêu của chúng ta là tìm đường đi ngắn nhất từ trạng thái xuất phát đến trạng thái đích. Không gian tìm kiếm là tất cả các đường đi từ trạng thái xuất phát đến trạng thái đích, hàm mục tiêu là độ dài đường đi. Người ta có thể giải quyết bài toán bằng các phương pháp tìm kiếm mù (tìm tất cả các đường đi từ trạng thái xuất phát đến trạng thái đích), sau đó so sánh độ dài của chúng và tìm ra đường đi ngắn nhất. Tuy nhiên, trong thực tế không thể áp dụng kỹ thuật này vì với bài toán có không gian tìm kiếm lớn thì đòi hỏi chi phí về thời gian rất cao. Mặt khác, các phương pháp heuristic như tìm kiếm tốt nhất - đầu tiên cho kết quả là đường đi ngắn nhất nhưng có thể kém hiệu quả, hay tìm kiếm leo đồi cũng chỉ cho kết quả là đường đi «tương đối tốt » chứ không đảm bảo đường đi là ngắn nhất. Do đó, để tăng hiệu quả tìm kiếm, chúng ta cần các phương pháp tìm kiếm heuristic sử dụng hàm đánh giá kết hợp.

Tìm kiếm tối ưu là tối thiểu chi phí ước lượng để đi đến mục tiêu. Một cách tổng quát, trong không gian tìm kiếm, mỗi đối tượng x được gán một giá trị hàm giá $f(x)$, ta cần tìm đối tượng x sao cho hàm mục tiêu $f(x)$ là lớn nhất (hoặc nhỏ nhất). Trong phần này, chúng tôi trình bày một số thuật toán trong bài toán tìm đường đi ngắn nhất.

a. Thuật giải A^T (Algorithm for Tree Search)

Thuật giải A^T là một phương pháp tìm kiếm theo kiểu BFS với độ tốt của nút là giá trị hàm ***g*** – tổng chiều dài con đường đã đi từ trạng thái bắt đầu đến trạng thái hiện tại.

Thuật giải A^T

1. Đặt **OPEN** chứa trạng thái khởi đầu.

2. Cho đến khi tìm được trạng thái đích hoặc không còn nút nào trong OPEN, thực hiện:

2a. Chọn trạng thái T_{max} có **giá trị *g* nhỏ nhất** trong OPEN và xóa T_{max} khỏi OPEN

2b. Nếu T_{max} là trạng thái kết thúc thì thoát.

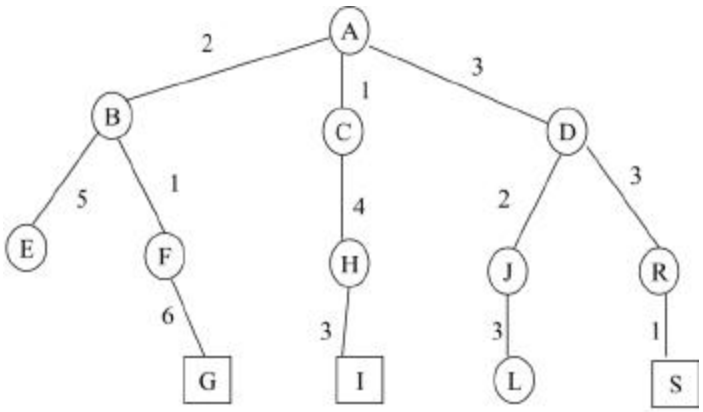
2c. Ngược lại, tạo ra các trạng thái kế tiếp T_k có thể có từ trạng thái T_{max}, với mỗi trạng thái kế tiếp T_k thực hiện:

$$g(T_k) = g(T_{max}) + \text{cost}\ (T_{max}, T_k);$$

Thêm T_k vào OPEN.

Vì chỉ sử dụng hàm g (mà không dùng hàm ước lượng h') để đánh giá độ tốt của một trạng thái.

Ví dụ 3.6: Cho cây tìm kiếm sau, tìm đường đi ngắn nhất từ A đến đích.



B1: mở A, g(A) =0;

B2: n = A;

B3: đóng A mở B, C, D. g(B) = g(A) + 2 = 2; g(C) =1; g(D) = 3; ⇒ PQ = {C, B, D}

B4 → B2: n =C;

B3: đóng C mở H; g(H) = 1+4 = 5

B3: đóng B mở E, F: g(E) = 2+5 =7; g(F) = 3;

B4: u = F;

B3: đóng F mở G; g(G) = g(F) +6 = 9

B4 → B2: u =D

B3: đóng D mở J, K; g(J)= 5; g(K) =6;

B4 → B2: u =H

B3: Đóng H mở I; g(I) = 8;

B4 → B2: u =J

B3: Đóng J mở L; g(L) = 8;

B4 → B2: u =K

B3: Đóng K mở S; g(S) = 7;

Vậy đường đi từ A → đích là: A → D → K → S với chi phí là 7

b. Thuật giải A^{KT} (Algorithm for Knowlegeable Tree Search)

Thuật giải A^{KT} mở rộng A^T bằng cách sử dụng thêm thông tin ước lượng h'. Độ tốt của một trạng thái f là tổng của hai hàm g và h'.

Thuật giải A^{KT}

1. Đặt **OPEN** chứa trạng thái khởi đầu.

2. Cho đến khi tìm được trạng thái đích hoặc không còn nút nào trong OPEN, thực hiện:

2a. Chọn trạng thái T_{max} có **giá trị *f* nhỏ nhất** trong OPEN và xóa T_{max} khỏi OPEN

2b. Nếu T_{max} là trạng thái kết thúc thì thoát.

2c. Ngược lại, tạo ra các trạng thái kế tiếp T_k có thể có từ trạng thái T_{max}, với mỗi trạng thái kế tiếp T_k thực hiện:

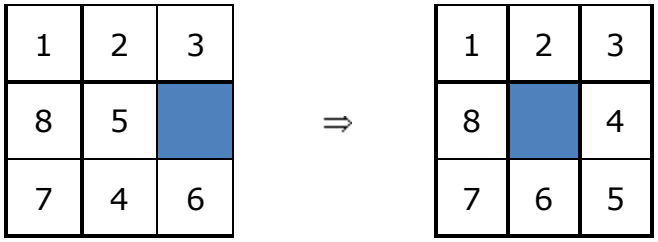
$$g(T_k) = g(T_{max}) + \text{cost}\ (T_{max}, T_k);$$

Tính $h'(T_k)$

$f(T_k) = g(T_k) + h'(T_k)$;

Thêm T_k vào OPEN.

Ví dụ 3.7: Bài toán 8 số (Taci)



a - Start b - Goal

$h_1(a, b)$ = Số vị trí có giá trị khác nhau giữa hai trạng thái a-Start và trạng thái b-Goal.

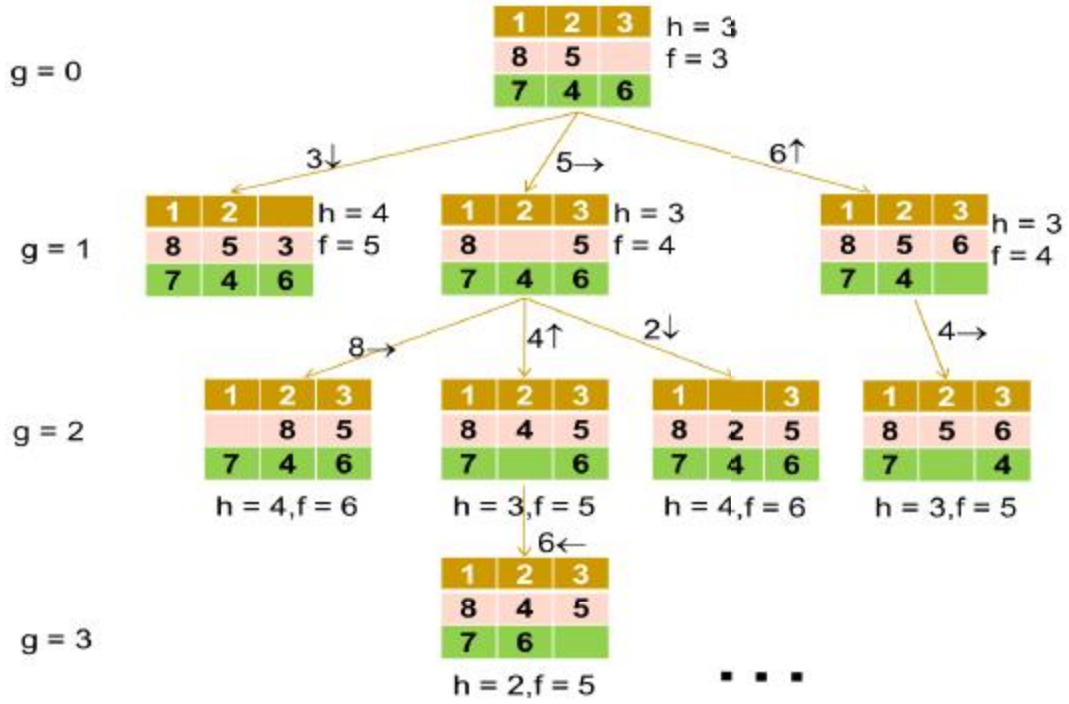
$h_2(a, b)$ = $\sum_{i\in[1,8]} \partial(i_a, i_b)$

Trong đó $\partial(i_a, i_b)$ là số lần ít nhất phải di chuyển giá trị *i* ở trạng thái a theo chiều ngang/ dọc về đúng vị trí của giá trị *i* ở trạng thái b.

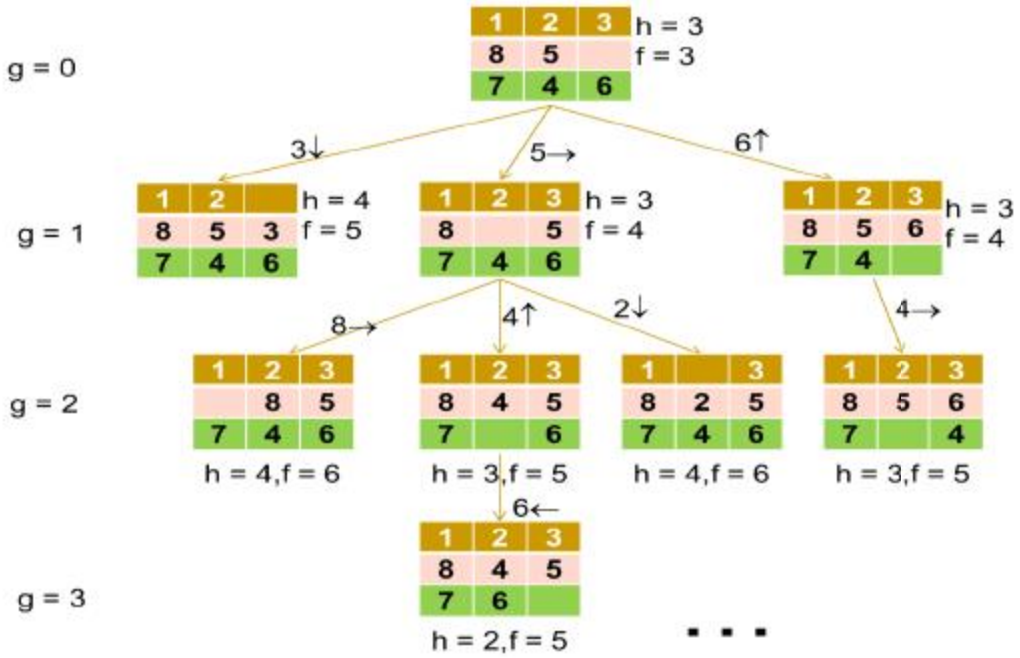
$h_1(S, G) = 0 + 0 + 0 + 1 + 1 + 1 + 0 + 0 = 3$

$h_2(S, G) = 0 + 0 + 0 + 2 + 2 + 1 + 0 + 0 = 5$

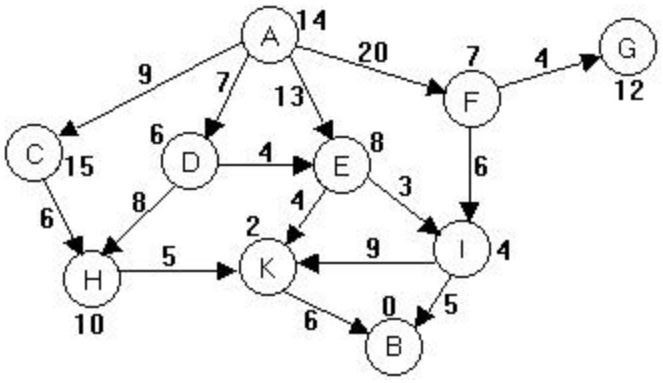
Sử dụng H1



Sử dụng H2



c. Thuật giải A*



Hình 3.9: Đồ thị không gian trạng thái với hàm đánh giá

A^* là một phiên bản đặc biệt của AKT áp dụng cho trường hợp đồ thị. Thuật giải A^* có sử dụng thêm tập hợp **CLOSE** để lưu trữ những trường hợp đã được xét đến. A^* mở rộng A^{KT} bằng cách bổ sung cách giải quyết trường hợp khi "mở" một nút mà nút này đã có sẵn trong OPEN hoặc CLOSE. Khi xét đến một trạng thái T_i bên cạnh việc lưu trữ 3 giá trị cơ bản g, h', f' để phản ánh độ tốt của trạng thái đó, A^* còn lưu trữ thêm hai thông số sau:

1. *Trạng thái cha của trạng thái T_i (ký hiệu là $P(T_i)$)*: cho biết trạng thái dẫn đến trạng thái T_i. Trong trường hợp có nhiều trạng thái dẫn đến T_i thì chọn $P(T_i)$ sao cho chi phí đi từ trạng thái khởi đầu đến T_i là thấp nhất, nghĩa là: $g(T_i) = g(T_P)$ + cost (T_p, T_i) là thấp nhất.

2. *Danh sách các trạng thái kế tiếp của T_i:* danh sách này lưu trữ các trạng thái kế tiếp T_k của T_i sao cho chi phí đến T_k thông qua T_i từ trạng thái ban đầu là thấp nhất.

Thuật giải *A

1. Đặt OPEN chỉ chứa T_0. Đặt $g(T_0) = 0$, $h'(T_0) = 0$ và $f'(T_0) = 0$. Đặt CLOSE là rỗng.

2. Lặp lại các bước sau cho đến khi gặp điều kiện dừng.

2a. Nếu OPEN rỗng: bài toán vô nghiệm, thoát.

2b. Ngược lại, chọn T_{max} trong OPEN sao cho $f'(T_{max})$ là nhỏ nhất

2b.1. Lấy T_{max} ra khỏi **OPEN** và đưa T_{max} vào **CLOSE.**

2b.2. Nếu T_{max} chính là trạng thái đích (G) thì thoát và thông báo lời giải là T_{max}.

2b.3. Nếu T_{max} không phải là G. Tạo ra danh sách *tất cả* các trạng thái kế tiếp của T_{max}. Gọi một trạng thái này là T_k. Với mỗi T_k, thực hiện các bước sau:

2b.3.1. Tính $g(T_k) = g(T_{max}) + \text{cost}(T_{max}, T_k)$.

2b.3.2. Nếu tồn tại $T_{k'}$ trong OPEN trùng với T_k.

Nếu $g(T_k) < g(T_{k'})$ thì

Đặt $g(T_{k'}) = g(T_k)$

Tính lại $f'(T_{k'})$

Đặt $P(T_{k'}) = T_{max}$

2b.3.3. Nếu tồn tại $T_{k'}$ trong CLOSE trùng với T_k.

Nếu $g(T_k) < g(T_{k'})$ thì

Đặt $g(T_{k'}) = g(T_k)$

Tính lại $f'(T_{k'})$

Đặt $P(T_{k'}) = T_{max}$

Lan truyền sự thay đổi giá trị g, f'cho tất cả các trạng thái kế tiếp của T_i đã được lưu trữ trong CLOSE và OPEN.

2b.3.4. Nếu T_k chưa xuất hiện trong cả **OPEN** lẫn **CLOSE** thì:

Thêm T_k vào OPEN

Tính: $f'(T_k) = g(T_k) + h'(T_k)$.

Ví dụ 3.8: Minh hoạ giải thuật A*

Thuật toán *A** là thuật toán sử dụng kỹ thuật tìm kiếm tốt nhất đầu tiên với hàm đánh giá $f(u)$.

Xét đồ thị không gian trạng thái trong Hình 3.9. Trong đó, trạng thái ban đầu là A, trạng thái đích là B, các giá trị ở cạnh là độ dài đường đi, các giá trị ở đỉnh là giá trị của hàm h. Đầu tiên, phát triển đỉnh A sinh ra các đỉnh con C, D, E và F. Tính giá trị của hàm f tại các đỉnh này ta có:

g(C) =9, f(C) = 9 + 15 = 24, g(D) = 7, f(D) = 7 + 6 = 13,

g(E) = 13, f(E) = 13 + 8 = 21, g(F) = 20, f(F) = 20 +7 = 27

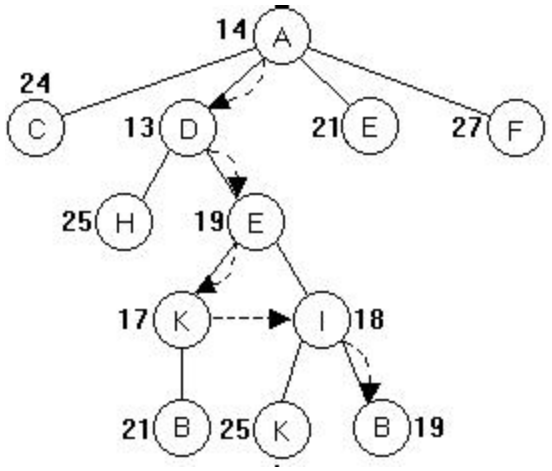
Như vậy đỉnh tốt nhất là D (f(D) = 13 là nhỏ nhất), phát triển D ta nhận được các đỉnh con H và E. Đánh giá H và E (mới):

g(H) = g(D) + Độ dài cung (D, H) = 7 + 8 = 15, f(H) = 15 + 10 = 25.

Đường đi tới E qua D có độ dài:

g(E) = g(D) + Độ dài cung (D, E) = 7 + 4 = 11.

Vậy đỉnh E mới có đánh giá là f(E) = g(E) + h(E) = 11 + 8 = 19. Trong số các đỉnh cho phát triển, thì đỉnh E với đánh giá f(E) = 19 là đỉnh tốt nhất. Phát triển đỉnh này, ta nhận được các đỉnh con của nó là K và I. Chúng ta tiếp tục quá trình trên cho tới khi đỉnh được chọn để phát triển là đỉnh kết thúc B, độ dài đường đi ngắn nhất tới B là g(B) = 19. Quá trình tìm kiếm trên được mô tả bởi cây tìm kiếm trong Hình 3.10, trong đó các số cạnh các đỉnh là các giá trị của hàm đánh giá $f(u)$.

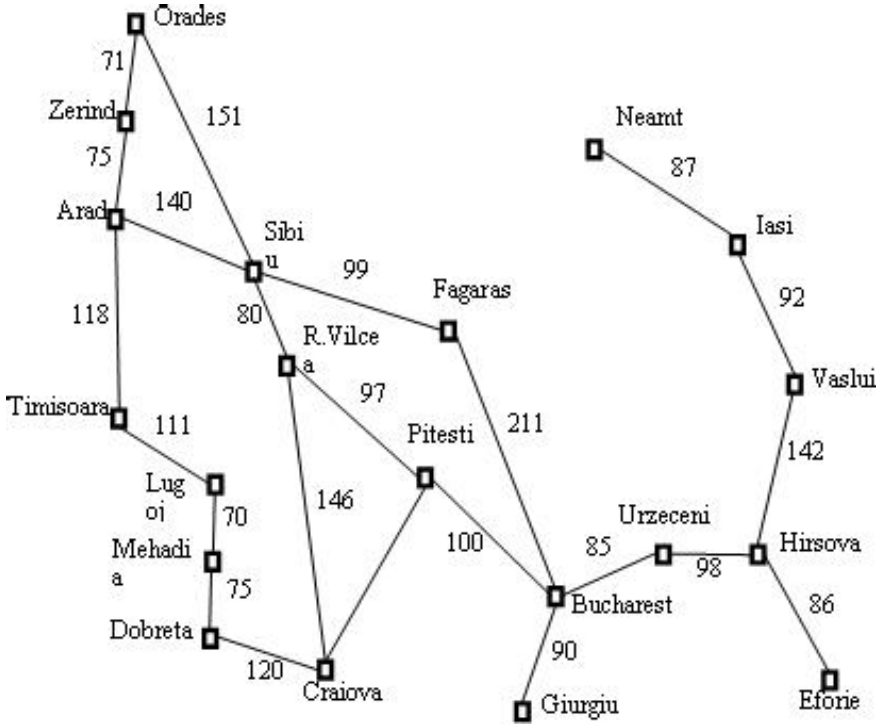


Hình 3.10: Cây tìm kiếm theo thuật giải A*

Ví dụ 3.9: Áp dụng thuật giải *A** tìm kiếm đường đi ngắn nhất từ thành phố *Arad* đến thành phố *Bucharest* của Romania. Bản đồ các thành phố của Romania được cho trong đồ thị như Hình 3.10. Trong đó mỗi đỉnh của đồ thị của là một thành phố, giữa hai đỉnh có cung nối nghĩa là có đường đi giữa hai thành phố tương ứng. Trọng số của cung chính là chiều dài (tính bằng km) của đường đi nối hai thành phố tương ứng, chiều dài theo đường chim bay một thành phố đến Bucharest được cho trong bảng kèm theo.

Bảng 3.1: Khoảng cách đường chim bay từ một thành phố đến Bucharest.

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Dobreta	242	Pitesti	98
Eforie	161	R.Vilcea	193
Fagaras	178	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



Hình 3.11: Bảng đồ của Romania với khoảng cách đường tính theo km

Chúng ta sẽ chọn hàm **h'** chính là khoảng cách đường chim bay cho trong bảng trên và hàm chi phí **cost(T_i, T_{i+1})** chính là chiều dài con đường nối từ thành phố T_i và T_{i+1}.

Sau đây là từng bước hoạt động của thuật toán A^* trong việc tìm đường đi ngắn nhất từ Arad đến Bucharest.

Ban đầu:

OPEN = {(Arad, g= 0, h'= 0, f'= 0)}, CLOSE = {}

Do trong OPEN chỉ chứa một thành phố duy nhất nên thành phố này sẽ là thành phố tốt nhất. Nghĩa là T_{max} = Arad, lấy Arad ra khỏi OPEN và đưa vào CLOSE.

OPEN = {}, CLOSE = {(Arad, g = 0, h'= 0, f'= 0)}

Từ Arad có thể đi đến được 3 thành phố là Sibiu, Timisoara và Zerind. Lần lượt tính giá trị *f', g* và *h'* của 3 thành phố này. Do cả 3 nút mới tạo ra này chưa có nút cha nên ban đầu nút cha của chúng đều là Arad.

h'(Sibiu) = 253

g(Sibiu) = g(Arad)+cost (Arad, Sibiu) = 0+140 = 140

f'(Sibiu) = g(Sibiu)+h'(Sibiu) = 140+253 = 393

P(Sibiu) = Arad

h'(Timisoara) = 329

g(Timisoara) = g(Arad)+cost (Arad, Timisoara) = 0+118 = 118

f'(Timisoara) = g(Timisoara)+ h'(Timisoara) = 118+329 = 447

P(Timisoara) = Arad

h'(Zerind) = 374

g(Zerind) = g(Arad)+cost (Arad, Zerind) = 0+75= 75

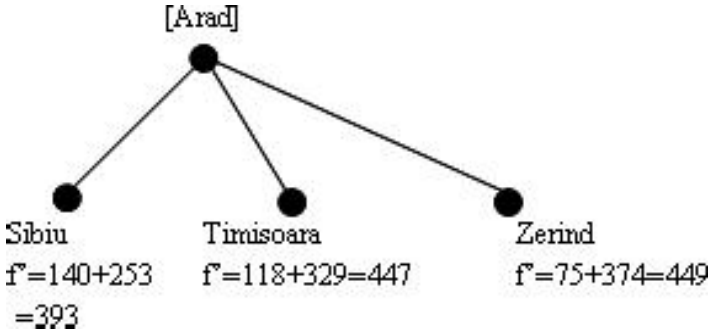
f'(Zerind) = g(Zerind)+h'(Zerind) = 75+374 = 449

P(Zerind) = Arad

Do cả 3 nút Sibiu, Timisoara, Zerind đều không có trong cả OPEN và CLOSE nên ta bổ sung 3 nút này vào OPEN.

OPEN = {(Sibiu, g=140, h'= 253, f' = 393, P = Arad), (Timisoara, g = 118, h'= 329, f'= 447, P = Arad), (Zerind, g = 75, h' = 374, f' = 449, P = Arad)}

CLOSE = {(Arad, g = 0, h' = 0, f' = 0)}



Nút được đóng ngoặc vuông [Arad] là nút trong tập CLOSE, ngược lại là trong tập OPEN. Trong tập OPEN, nút Sibiu là nút có giá trị f' nhỏ nhất nên ta sẽ chọn T_{max} = Sibiu, lấy Sibiu ra khỏi OPEN và đưa vào CLOSE.

OPEN = {(Timisoara, g = 118, h' = 329, f' = 447, P = Arad), (Zerind, g = 75, h' = 374, f' = 449, P = Arad)}

CLOSE = {(Arad, g = 0, h' = 0, f' = 0), (Sibiu, g=140, h'= 253, f' = 393, P = Arad)}

Từ Sibiu có thể đi đến được 4 thành phố là: Arad, Fagaras, Oradea, Rimnicu. Lần lượt tính các giá trị g, h', f' cho các nút này.

h'(Arad) = 366

g(Arad) = g(Sibiu) + cost (Sibiu, Arad) = 140+140 = 280

f'(Arad) = g(Arad) + h'(Arad) = 280+366 = 646

h'(Fagaras) = 178

g(Fagaras) = g(Sibiu) + cost (Sibiu, Fagaras) = 140+99 = 239

f'(Fagaras) = g(Fagaras) + h'(Fagaras) = 239+178 = 417

h'(Oradea) = 380

g(Oradea) = g(Sibiu) + cost (Sibiu, Oradea) = 140+151 = 291

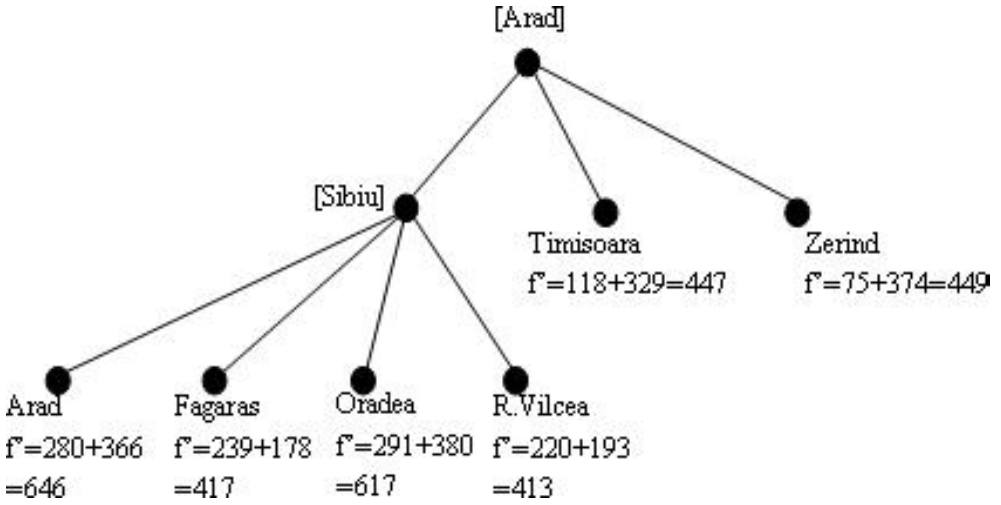
f'(Oradea) = g(Oradea) + h'(Oradea) = 291+380 = 671

h'(R.Vilcea) = 193

g(R.Vilcea) = g(Sibiu) + cost(Sibiu, R.Vilcea) = 140+80 = 220

f'(R.Vilcea) = g(R.Vilcea) + h'(R.Vilcea) = 220+193 = 413

Nút Arad đã có trong CLOSE. Tuy nhiên, do g(Arad) mới được tạo ra (có giá trị 280) lớn hơn g(Arad) lưu trong CLOSE (có giá trị 0) nên ta sẽ không cập nhật lại giá trị g và f' của Arad lưu trong CLOSE. 3 nút còn lại: Fagaras, Oradea, Rimnicu đều không có trong cả OPEN và CLOSE nên ta sẽ đưa 3 nút này vào OPEN, đặt cha của chúng là Sibiu. Như vậy, đến bước này OPEN đã chứa tổng cộng 5 thành phố.



OPEN = {(Timisoara, g = 118, h' = 329, f' = **447**, P = Arad), (Zerind, g = 75, h' = 374, f' = **449**, P = Arad), (Fagaras, g = 239, h' = 178, f' = **417**, P = Sibiu), (Oradea, g = 291, h'= 380, f' = **617**, P = Sibiu), (R.Vilcea, g = 220, h' = 193, f' = **413**, P = Sibiu)}

CLOSE = {(Arad, g = 0, h' = 0, f' = 0), (Sibiu, g = 140, h' = 253, f' = 393, P = Arad)}

Trong tập OPEN, nút R.Vilcea là nút có giá trị f' nhỏ nhất, chọn T_{max} = R.Vilcea. Chuyển R.Vilcea từ OPEN sang CLOSE. Từ R.Vilcea có thể đi đến được 3 thành phố là Craiova, Pitesti và Sibiu. Lần lượt tính giá trị f', g và h' của 3 thành phố này.

h'(Sibiu) = 253

g(Sibiu) = g(R.Vilcea) + cost(R.Vilcea,Sibiu) = 220 + 80 = 300

f'(Sibiu) = g(Sibiu) + h'(Sibiu) = 300 + 253 = 553

h'(Craiova) = 160

g(Craiova) = g(R.Vilcea) + cost(R.Vilcea, Craiova) = 220 + 146 = 366

f'(Craiova) = g(Fagaras) + h'(Fagaras) = 366 + 160 = 526

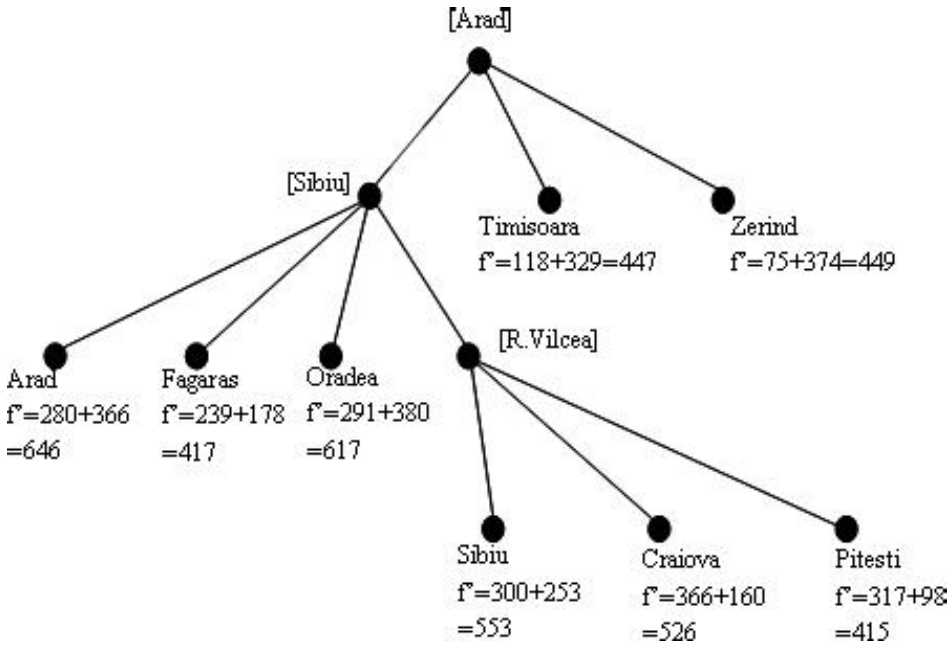
h'(Pitesti) = 98

g(Pitesti) = g(R.Vilcea) + cost(R.Vilcea, Pitesti) = 220 + 97 = 317

f'(Pitesti) = g(Oradea) + h'(Oradea) = 317 + 98 = 415

Sibiu đã có trong tập CLOSE. Tuy nhiên, do g'(Sibiu) mới (có giá trị là 553) lớn hơn g'(Sibiu) (có giá trị là 393) nên ta sẽ không cập nhật lại các giá trị của Sibiu được lưu

trong CLOSE. Còn lại 2 thành phố là Pitesti và Craiova đều không có trong cả OPEN và CLOSE nên ta sẽ đưa nó vào OPEN và đặt cha của chúng là R.Vilcea.



OPEN = {(Timisoara, g = 118, h' = 329, f' = **447**, P = Arad), (Zerind, g = 75, h' = 374, f' = **449**, P = Arad), (Fagaras, g = 239, h' = 178, f' = **417**, P = Sibiu), (Oradea, g = 291, h'= 380, f' = **617**, P = Sibiu), (Craiova, g = 366, h' = 160, f'= **526**, P = R.Vilcea), (Pitesti, g = 317, h' = 98, f' = **415**, P = R.Vilcea)}

CLOSE = {(Arad, g = 0, h' = 0, f' = 0), (Sibiu, g = 140, h' = 253, f' = 393, P = Arad), (R.Vilcea, g = 220, h' = 193, f' = **413**, P = Sibiu) }

Trong tập OPEN, nút tốt nhất là Pitesti, từ Pitesti ta có thể đi đến được R.Vilcea, Bucharest và Craiova. Lấy Pitesti ra khỏi OPEN và đặt nó vào CLOSE. Thực hiện tiếp theo tương tự như trên, ta sẽ không cập nhật giá trị f', g của R.Vilcea và Craiova lưu trong CLOSE. Sau khi tính toán f', g của Bucharest, ta sẽ đưa Bucharest vào tập OPEN, đặt P(Bucharest) = Pitesti.

h'(Bucharest) = 0

g(Bucharest) = g(Pitesti) + cost(Pitesti, Bucharest) = 317 + 100 = 418

f'(Bucharest) = g(Fagaras) + h'(Fagaras) = 417 + 0 = 417

Ở bước kế tiếp, T_{max} = Bucharest. Và như vậy thuật toán kết thúc (tại bước này, có hai ứng cử viên là Bucharest và Fagaras vì đều cùng có f'= 417, nhưng vì Bucharest là đích nên ta sẽ ưu tiên chọn hơn).

Để xây dựng lại con đường đi từ Arad đến Bucharest ta lần theo giá trị P được lưu trữ kèm với *f', g* và *h'* cho đến lúc đến Arad, với P(Bucharest) = Pitesti, P(R.Vilcea) = Sibiu, P(Sibiu) = Arad. Vậy con đường đi ngắn nhất từ Arad đến Bucharest là Arad, Sibiu, R.Vilcea, Pitesti, Bucharest.

Trong ví dụ này, hàm *h'* có chất lượng khá tốt và cấu trúc đồ thị khá đơn giản nên ta gần như đi thẳng đến đích mà ít phải khảo sát các con đường khác.

d. Bài toán người bán hàng (TSP – Travelling Salesman Problem)

Bài toán: Có *n* thành phố (được đánh số từ 1 đến *n*), một người bán hàng xuất phát từ một thành phố, muốn đi qua tất cả các thành phố khác, mỗi thành phố một lần rồi quay về thành phố xuất phát. Giả thiết biết được chi phí đi từ thành phố *i* đến thành phố *j* là c_{ij}, hãy tìm một hành trình cho người bán hàng sao cho tổng chi phí theo hành trình này là thấp nhất. Bài toán này được gọi với tên gọi quen thuộc là bài toán người bán hàng (Traveling Saleman Problem-TSP).

Giải quyết: Bài toán chính là tìm chu trình Hamilton với chi phí thấp nhất, có thể sử dụng đệ qui phi tuyến để giải quyết, kết quả tối ưu nhưng độ phức tạp cao. Một cách giải quyết gần đúng là sử dụng phương pháp tham lam, cụ thể là thuật giải GTS (Greedy Travelling Salesman)

3.3.1.1 Thuật giải GTS1

Bước 1: [Khởi đầu]

COST = 0, TOUR = ∅, v = u (u là thành phố xuất phát)

Bước 2: [Thăm tất cả các thành phố]

Cho k chạy từ 1 đến n – 1 qua bước 3

Bước 3: [Tìm cạnh có chi phí thấp nhất]

Tìm (v, w) là cạnh có chi phí thấp nhất từ v đến các đỉnh chưa đi qua w

COST = COST + C[v, w]

TOUR = TOUR + (v, w)

v = w

Bước 4: [Quay về thành phố xuất phát]

COST = COST + C[v, u]

TOUR = TOUR + (v, u)

Ví dụ 3.10: Cho đồ thị như sau:

	A	B	C	D	E	F
A	**0**	**100**	**50**	**30**	**200**	**150**
B	**120**	**0**	**30**	**80**	**120**	**50**
C	**80**	**20**	**0**	**120**	**100**	**30**
D	**40**	**70**	**130**	**0**	**50**	**120**
E	**180**	**150**	**130**	**70**	**0**	**200**
F	**200**	**80**	**60**	**100**	**150**	**0**

a. Tìm hành trình tốt nhất và chi phí tương ứng theo thuật giải GTS1 với thành phố xuất phát là A.

b. Câu hỏi tương tự câu a nhưng thành phố xuất phát là C. Có nhận xét gì về hai kết quả trên?

Hướng dẫn

Bước 1: COST = 0, TOUR = ∅, v = A

Bước 2: k = 1... 5

Bước 3:

k = 1:

Từ A có thể đi đến B, C, D, E, F: {AB =100, AC =50, AD =30, AE =200, AF = 150}

- w = D, COST = 0 + 30 =30, TOUR = {(AD)}, v = D

k = 2:

Từ D có thể đi đến B, C, E, F: {DB = 70, DC =130, DE =50, DF = 120}

- w = E, COST = 30 + 50 =80, TOUR = {(AD), (DE)}, v = E

k = 3:

- Từ E có thể đi đến B, C, F: {EB = 150, EC = 130, EF = 200}

w = C, COST = 80 + 130 = 210, TOUR ={(AD), (DE), (EC)}, v = C

k = 4:

Từ C đi đến B, F: {CB = 20, CF = 30}

- w = B, COST = 230, TOUR = {(AD), (DE), (EC), (CB)}, v = B

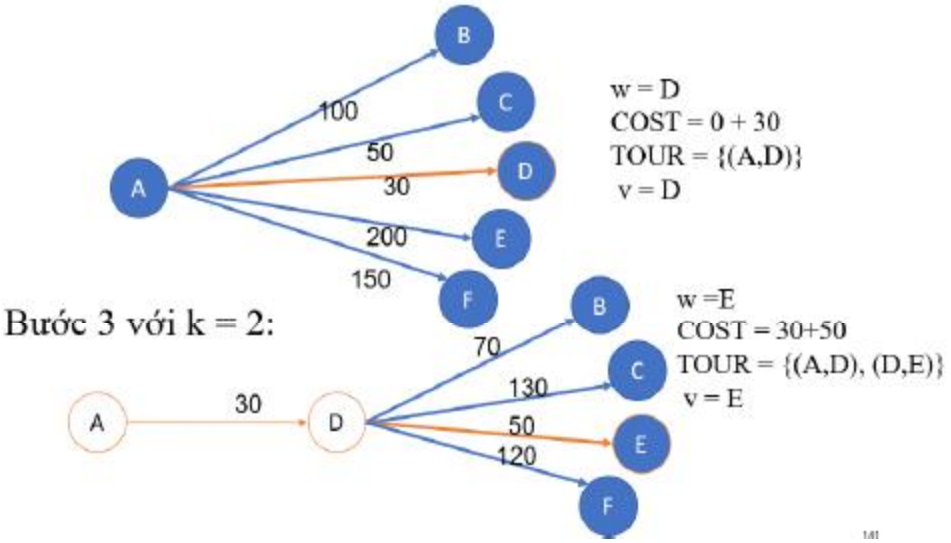
k = 5:

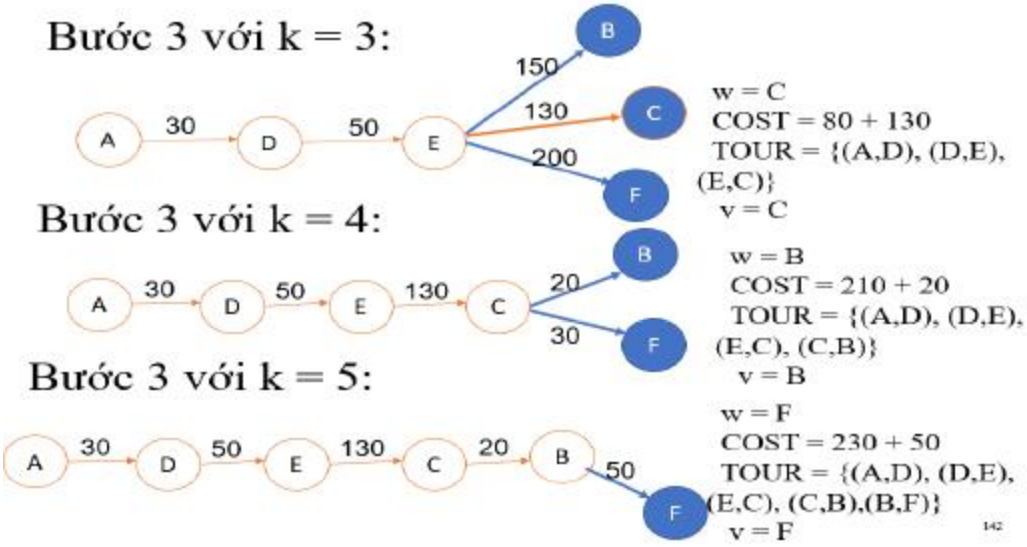
Từ B đi đến F: {BF = 50}

- w = F, COST = 280, TOUR = {(AD), (DE), (EC), (CB), (BF)}, v = F

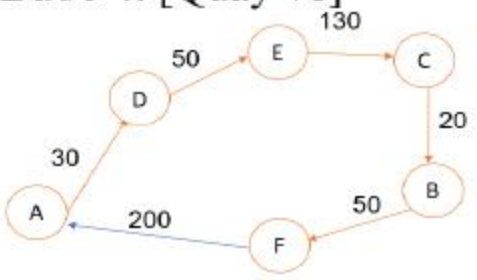
Bước 4: Quay về A {FA = 200}

COST = 480, TOUR = {(AD), (DE), (EC), (CB), (BF), (FA)}





Bước 4: [Quay về]



COST = 280 + 200 = 480
TOUR = {(A,D), (D,E), (E,C), (C,B),(B,F), (F,A)}

3.3.1.2 Thuật giải GTS2

Bước 1: COST = ∞, BEST = ∅, k = 0

Bước 2: Nếu k < p qua Bước 3, ngược lại dừng

Bước 3: k = k + 1

Gọi GTS1 với thành phố xuất phát là v_k ta được chi phí C_k và hành trình T_k.

Bước 4: Nếu C_k < COST thì

COST = C_k, BEST = T_k.

Bước 5: Quay lại Bước 2.

Ví dụ 3.11: Cho đồ thị như sau:

	A	B	C	D	E	F
A	∞	8	10	20	15	6
B	6	∞	12	15	8	10
C	8	9	∞	10	20	15
D	15	6	8	∞	10	20
E	20	15	20	25	∞	8
F	30	40	20	10	30	∞

P =3, V = {A, C, E}

Hướng dẫn

Bước 1: cost = ∞, best = ∅, k =0;

Bước 2: k =0 < p qua B3

Bước 3:

k =1:

Gọi GTS1 với thành phố xuất phát là A

$T_1 = A \to F \to D \to B \to E \to C \to A$

$C_1 = 6 + 10 + 6 + 8 + 20 + 8 = 58$

Bước 4:

C_1 < Cost nên Cost = C_1 = 58

Best = $T_1 = A \to F \to D \to B \to E \to C \to A$

Bước 2: k =1 < p

Bước 3:

k =2:

Gọi GTS1 với thành phố xuất phát là C

$T_2 = C \to A \to F \to D \to B \to E \to C$

$C_2 = 8 + 6 + 10 + 6 + 8 + 20 = 58$

Bước 4:

C_2 = cost nên cost = C_1; best = T_1

Bước 2: k = 2 < p

Bước 3:

k =3:

Gọi GTS1 với thành phố xuất phát là E.

$T_3 = E \to F \to D \to B \to A \to C \to E$

$C_3 = 8 + 10 + 6 + 6 + 10 + 20 = 60$

Bước 4:

C_3 > cost nên cost = C_1; best = T_1

Bước 2: k = 3 = p dừng.

Vậy hành trình tốt nhất với chi phí tương ứng là: T = A → F → D → B → E → C → A

C = 6 + 10 + 6 + 8 + 20 + 8 = 58

3.4 BÀI TOÁN PHÂN CÔNG CÔNG VIỆC

Bài toán: Cho *n* công việc $\{J_1, J_2, \ldots, J_n\}$ với thời gian thực hiện là $T = \{t_1, t_2, \ldots, t_n\}$ và *m* máy. Hãy phân công các công việc vào các máy sao cho thời gian hoàn tất các công việc là thấp nhất. Đây là bài toán lập lịch với độ phức tạp cao, dùng heuristic để giải quyết.

Giải quyết

- *Lập một thứ tự L các công việc cần được thực hiện.*
- *Lặp lại các công việc sau cho đến khi nào các công việc đều được phân công:*
- *Nếu có máy nào rảnh thì nạp công việc kế tiếp trong danh sách L vào (nếu có 2 hay nhiều máy cùng rảnh tại một thời điểm thì máy với chỉ số thấp sẽ được phân cho công việc).*

Giả sử ban đầu các máy đều chưa thực hiện công việc nào và có cùng công suất.

Bước 1: i = 0, sắp xếp các công việc theo chiều giảm dần của thời gian thực hiện.

Bước 2: i = i + 1

Phân công việc i cho máy có thời gian hoàn tất các công việc hiện hành là thấp nhất.

Bước 3: lặp lại Bước 2 cho đến khi i = n

Ví dụ 3.12: cho n = 8 và T = {10, 6, 16, 12, 2, 4, 2, 8}, m=3. Tính thời gian hoàn tất các công việc?

Hướng dẫn: t(M1) = t(M2) = t(M3) = 0

Bước 1: sắp xếp T = {16, 12, 10, 8, 6, 4, 2, 2}, i = 0

Bước 2 (i = 1): phân công việc J3 có thời gian thực hiện là 16 cho máy 1 ⇒ t(M1) = 16

Bước 3: Lặp Bước 2

i = 2: phân công việc J4 có thời gian thực hiện là 12 cho máy 2 ⇒ t(M2) = 12

i = 3: phân công việc J1 có thời gian thực hiện là 10 cho máy 3 ⇒ t(M3) = 10

i = 4: phân công việc J8 có thời gian thực hiện là 8 cho máy 3 ⇒ t(M3) = 10 + 8 = 18

i = 5: phân công việc J2 có thời gian thực hiện là 6 cho máy 2 ⇒ t(M2) = 12 + 6 = 18

i = 6: phân công việc J6 có thời gian thực hiện là 4 cho máy 1 ⇒ t(M1) = 16 + 4 = 20

i = 7: phân công việc J5 có thời gian thực hiện là 2 cho máy 2 ⇒ t(M2) = 18 + 2 = 2

i = 8: phân công việc J7 có thời gian thực hiện là 2 cho máy 3 ⇒ t(M3) = 20.

BÀI 4: BIỂU DIỄN VÀ XỬ LÝ TRI THỨC

Học xong bài này sinh viên có thể nắm được:

- *Các khái niệm liên quan đến xử lý và biểu diễn tri thức.*
- *Mệnh đề, logic vị từ.*
- *Các thuật toán liên quan Robinson, Vương Hạo.*
- *Mạng ngữ nghĩa.*

4.1 LOGIC MỆNH ĐỀ

Logic mệnh đề là kiểu biểu diễn tri thức đơn giản và gần gũi nhất đối với con người, là phương pháp biểu diễn tri thức cổ điển nhất trong máy tính, nó cho phép xây dựng nên các công thức. Mệnh đề là một khẳng định, một phát biểu mà giá trị của nó chỉ có thể hoặc là đúng hoặc là sai.

Ví dụ 4.1: "1+1=2" có giá trị đúng hay "Mọi loại cá có thể sống trên bờ" có giá trị sai.

Logic mệnh đề dùng kí hiệu để biểu diễn tri thức và các toán tử áp lên các kí hiệu để suy luận logic.

Các ký hiệu:

- Hai hằng logic True và False.
- Các ký hiệu mệnh đề (còn được gọi là các biến mệnh đề): P, Q, R, ...
- Các toán tử logic: phép hội ($\wedge$), phép tuyển ($\vee$), phép phủ định ($\neg$), kéo theo ($\rightarrow$), kéo theo kép ($\leftrightarrow$).
- **Modus Ponens**: Nếu mệnh đề P là đúng và mệnh đề P $\rightarrow$ Q là đúng thì giá trị của Q sẽ là đúng.
- **Modus Tollens**: Nếu mệnh đề P $\rightarrow$ Q là đúng và mệnh đề Q là sai thì giá trị của P sẽ là sai.

Cú pháp:

Cú pháp bao gồm các ký hiệu và các quy tắc liên kết các ký hiệu để tạo thành các công thức trong logic mệnh đề. Các công thức là các ký hiệu mệnh đề sẽ được gọi là các *câu đơn* hoặc *câu phân tử*. Các công thức không phải là câu đơn sẽ được gọi là câu phức hợp. Nếu P là mệnh đề thì P và TP được gọi là *literal*, P là *literal dương*, còn TP là *literal âm.*

Ngữ nghĩa của logic mệnh đề cho phép *xác định* ý nghĩa của các công thức trong thế giới thực nào đó và được xác định thông qua bảng chân trị True (1) hoặc False (0).

Bảng 4.1: Bảng chân trị của các phép toán logic

P	¬P	Q	P ∧ Q	P ∨ Q	P → Q	P ↔ Q
0	1	0	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	0	1	1	1	1	1

4.1.1 Các qui luật logic cơ bản

Hai công thức P và Q được xem là tương đương nếu chúng có cùng một giá trị chân lý với mọi giá trị chân lý của các mệnh đề thành phần. Khi P tương đương với Q, ta viết P ≡ Q hay P ⇔ Q. Bằng bảng chân lý, dễ dàng chứng minh được sự tương đương của các công thức sau đây:

1. Luật phủ định kép: ¬(¬P) ≡ P

2. Luật về phần tử bù:

a. P ∨ ¬P ≡ 1

b. P ∧ ¬P ≡ 0

3. Luật về phần tử trung hòa:

a. P ∨ 0 ≡ P

b. P ∧ 1 ≡ P

4. Luật về phép kéo theo:

a. P→Q ≡ ¬P ∨ Q

b. $P \rightarrow Q \equiv \neg Q \rightarrow \neg P$

5. Luật về phép kéo theo kép: $P \leftrightarrow Q \equiv (P \rightarrow Q) \wedge (Q \rightarrow P)$.

6. Luật DeMorgan:

a. $\neg(P \vee Q) \equiv \neg P \wedge \neg Q$

b. $\neg(P \wedge Q) \equiv \neg P \vee \neg Q$

7. Luật phân phối:

a. $R \vee (P \wedge Q) \equiv (R \vee P) \wedge (R \vee Q)$

b. $R \wedge (p \vee Q) \equiv (R \wedge p) \vee (R \wedge Q)$

8. Luật giao hoán:

a. $P \wedge Q \equiv Q \wedge P$

b. $P \vee Q \equiv Q \vee P$

9. Luật kết hợp:

a. $(P \wedge Q) \wedge R \equiv P \wedge (Q \wedge R)$

b. $(P \vee Q) \vee R \equiv P \vee (Q \vee R)$

10. Luật lũy đẳng:

a. $P \wedge P \equiv P$

b. $P \vee P \equiv P$

11. Luật thống trị:

a. $P \wedge 0 \equiv 0$

b. $P \vee 1 \equiv 1$

12. Luật hấp thu:

a. $P \vee (P \wedge Q) \equiv P$

b. $P \wedge (P \vee Q) \equiv P$

4.1.2 Dạng chuẩn hội (Conjunctive normal form – CNF)

Một công thức P được gọi là có dạng tuyển chuẩn nếu $P = P_1 \wedge P_2 \wedge \ldots \wedge P_n$ trong đó các $P_i = P_{i1} \wedge P_{i2} \wedge \ldots \wedge P_{imi}$ với P_{ij} là literal với $\forall i = 1 \ldots n, \forall j = 1 \ldots m_i$.

Dễ thấy rằng mọi công thức trong logic mệnh đề đều có thể đưa về dạng tuyển chuẩn.

Ví dụ 4.2: $P \rightarrow (Q \wedge \neg R) \vee (\neg T \wedge S)$

$\equiv \neg P \vee (Q \wedge \neg R) \vee (\neg T \wedge S)$ (luật kéo theo)

$\equiv [(\neg P \vee Q) \wedge (\neg P \vee \neg R)] \vee (\neg T \wedge S)$ (luật phân phối)

$\equiv ([(\neg P \vee Q) \wedge (\neg P \vee \neg R)] \vee \neg T) \wedge ([(\neg P \vee Q) \wedge (\neg P \vee \neg R)] \vee S)$ (luật phân phối)

$\equiv (\neg P \vee Q \vee \neg T) \vee (\neg P \vee \neg R \vee \neg T) \wedge (\neg P \vee Q \vee S) \wedge (\neg P \vee \neg R \vee S)$ (luật phân phối)

4.1.3 Câu dạng Horn

Một công thức dạng $P_1 \wedge P_2 \wedge ... \wedge P_n \rightarrow Q$ trong đó P_i và Q là các literal dương, $\forall i = 1 ... n$ được gọi là một câu dạng Horn.

Câu dạng Horn trên có thể được viết lại dưới dạng luật ***If-then*** như sau:

If $P_1 \wedge P_2 \wedge ... \wedge P_n$ ***then*** Q hay Nếu $P_1 \wedge P_2 \wedge ... \wedge P_n$ thì Q.

4.1.4 Luật Suy Diễn

Một công thức H được gọi hệ quả logic của tập các công thức $G = \{G_1, G_2, ..., G_n\}$ nếu trong mọi trường hợp của các mệnh đề thành phần của các G_i làm các tất cả G_i đúng thì cũng làm cho H đúng.

Các luật suy diễn có thể được dùng để suy ra các tri thức mới từ một cơ sở tri thức đã có. Tri thức mới chính là hệ quả logic của các tri thức trong cơ sở tri thức. Luật suy diễn giống như một thủ tục mà ta sử dụng để sinh ra một công thức mới từ các công thức đã có.

Một luật suy diễn gồm hai phần: một tập các điều kiện (giả thiết) và một kết luận, trong đó kết luận chính là hệ quả logic của tập các điều kiện. Nói cách khác, mọi trường hợp của các mệnh đề thành phần của giả thiết làm giả thiết đúng cũng làm kết luận đúng. Lúc đó người ta bảo luật suy diễn có tính đúng đắn.

Sau đây là số luật suy diễn cơ bản trong đó P, P_i, Q, R là các công thức

1. Luật khẳng định (Modus Ponens)

$(P \rightarrow Q) \wedge P \Rightarrow Q$

2. Luật phủ định **(Modus Tollens)**

$(P \rightarrow Q) \wedge \neg Q \Rightarrow \neg P$

3. 3. Luật bắc cầu (tam đoạn luận)

$(P \rightarrow Q) \wedge (Q \rightarrow R) \Rightarrow (P \rightarrow R)$

4. Luật loại bỏ hội **(And-Elimination)**

$P_1 \wedge P_2 \wedge ... \wedge P_n \Rightarrow P_i$

5. Luật đưa vào hội **(And-Introduction)**

$P_1, P_2, ..., P_n \Rightarrow P_1 \wedge P_2 \wedge ... \wedge P_n$

6. Luật đưa vào tuyển **(Or-Introduction)**

$P_i \Rightarrow P_1 \vee P_2 \vee ... \vee P_n$

7. Luật loại bỏ phủ định hai lần **(Elimination of Double Negatioon)**

$\neg(\neg P) \Rightarrow P$

8. Luật hợp giải **(Resolution)**

$(P \vee Q) \wedge (\neg Q \vee R) \Rightarrow P \vee R$

9. Luật hợp giải đơn **(Unit Resolution)**

$(P \vee Q) \wedge \neg Q \Rightarrow P$

4.1.5 Các giải thuật liên quan đến Logic mệnh đề

Một trong những vấn đề khá quan trọng của logic mệnh đề là chứng minh tính đúng đắn của phép suy diễn $(a \rightarrow b)$. Đây cũng chính là bài toán chứng minh thường gặp trong toán học.

Với hai phép suy luận cơ bản của logic mệnh đề (Modus Ponens, Modus Tollens) cùng với các phép biến đổi hình thức, ta cũng có thể chứng minh được phép suy diễn. Tuy nhiên, thao tác biến đổi hình thức là rất khó cài đặt được trên máy tính. Với công cụ máy tính, bạn có thể cho rằng ta sẽ dễ dàng chứng minh được mọi bài toán bằng một phương pháp "thô bạo" là lập bảng chân trị. Tuy về lý thuyết, phương pháp lập bảng chân trị luôn cho được kết quả cuối cùng nhưng độ phức tạp của phương pháp

này là quá lớn, $O(2^n)$ với n là số biến mệnh đề. Sau đây chúng ta sẽ nghiên cứu hai phương pháp chứng minh mệnh đề với độ phức tạp chỉ có O(n).

4.2 BIỂU DIỄN TRI THỨC

4.2.1 Định nghĩa

Tri thức là những điều hiểu biết có hệ thống về sự vật, hiện tượng tự nhiên hoặc xã hội. Tri thức có được thông qua quá trình thu nhận dữ liệu, xử lí, lưu trữ, sau đó được đưa vào mạng tri thức đã có. Trên cơ sở đó, thực hiện các liên kết, suy diễn, kiểm chứng để sinh ra tri thức mới. Tri thức là điều kiện tiên quyết của các hành xử thông minh.

Trong ngữ cảnh của ngành khoa học máy tính, người ta quan niệm rằng **dữ liệu** là các con số, chữ cái, hình ảnh, âm thanh... mà máy tính có thể tiếp nhận và xử lý. Bản thân dữ liệu thường không có ý nghĩa đối với con người. Còn thông tin là tất cả những gì mà con người có thể cảm nhận được một cách trực tiếp thông qua các giác quan của mình hoặc gián tiếp thông qua các phương tiện kỹ thuật như internet, tivi, ... Với phương tiện máy tính, con người sẽ tiếp thu được *một phần* dữ liệu có ý nghĩa đối với mình.

4.2.2 Phân loại

Người ta thường phân loại tri thức ra làm các dạng như sau:

Tri thức sự kiện: là các khẳng định về một sự kiện, khái niệm nào đó trong một phạm vi xác định. Các định luật vật lý, toán học, ... thường được xếp vào loại này (mặt trời mọc ở đằng đông, tam giác đều có 3 góc 60^0, ...)

Tri thức thủ tục: mô tả cách thức giải quyết một vấn đề. Loại tri thức này cung cấp một tập hợp các chỉ dẫn về cách thực hiện các tác vụ nhất định, đưa ra giải pháp để thực hiện một công việc nào đó. Thuật toán, thuật giải là một dạng của tri thức thủ tục.

Tri thức mô tả: cho biết một đối tượng, sự kiện, vấn đề, khái niệm, ... được thấy, cảm nhận, cấu tạo như thế nào. Loại tri thức này thường bao gồm các phát biểu đơn giản, dưới dạng các khẳng định logic đúng hoặc sai (một cái bàn thường có 4 chân, con người có 2 tay, 2 mắt, ...)

Tri thức Heuristic: là một dạng tri thức cảm tính. Các tri thức thuộc loại này thường có dạng ước lượng, phỏng đoán, và thường được hình thành thông qua kinh

nghiệm. Tri thức Heuristic đôi khi còn được gọi là kiến thức nông cạn (nếu thấy đám khói lớn cuồn cuộn bốc lên ở đằng xa, thì biết ở đó có cháy nhà).

Siêu tri thức: giúp lựa chọn tri thức thích hợp nhất trong số các tri thức khi giải quyết một vấn đề. (tri thức về huyết áp là quan trọng để chẩn đoán bệnh hơn là các tri thức về màu mắt)

Tri thức có cấu trúc: mô tả mô hình tổng quan của một hệ thống, bao gồm các khái niệm, các khái niệm con, các đối tượng, diễn tả các chức năng và mối liên hệ giữa các chúng dựa trên một cấu trúc xác định.

Tri thức không quyết định sự thông minh nhưng nó là một yếu tố cơ bản cấu thành trí thông minh. Vì vậy, để xây dựng một trí thông minh nhân tạo, ta cần phải có yếu tố cơ bản này. Các phương pháp đưa tri thức vào máy tính được gọi là biểu diễn tri thức.

4.2.3 Biểu diễn tri thức

Biểu diễn tri thức và lập luận là lĩnh vực trí tuệ nhân tạo trong việc biểu diễn thông tin về thế giới thực dưới một hình thức mà hệ thống máy tính có thể sử dụng để giải quyết các nhiệm vụ phức tạp như chẩn đoán sức khoẻ hoặc đối thoại bằng ngôn ngữ tự nhiên. Biểu diễn tri thức kết hợp các phát hiện từ tâm lý học về cách con người giải quyết các vấn đề và thể hiện kiến thức để thiết kế dưới các hình thức làm cho các hệ thống phức tạp trở nên dễ dàng thiết kế và xây dựng hơn. Biểu diễn tri thức và lập luận cũng kết hợp các phát hiện từ logic để tự động hóa các loại lập luận khác nhau, chẳng hạn như áp dụng các luật hoặc các mối quan hệ giữa tập hợp và tập con.

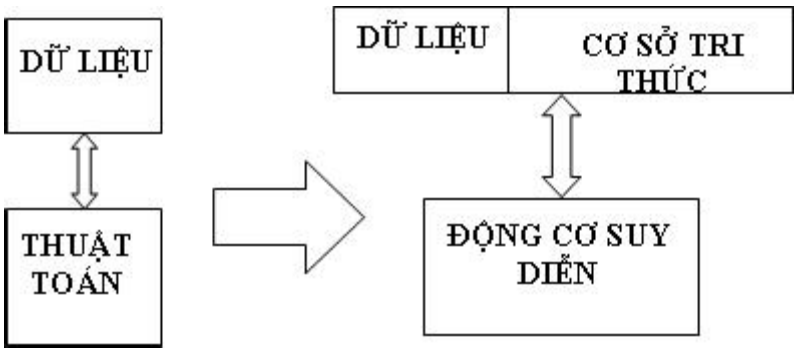
Tri thức được mô tả dưới dạng các câu trong *ngôn ngữ biểu diễn tri thức*. Mỗi câu có thể được xem như sự mã hóa của một sự hiểu biết về thế giới hiện thực. Ngôn ngữ biểu diễn tri thức gồm hai thành phần cơ bản là *cú pháp* và *ngữ nghĩa*.

- Cú pháp của một ngôn ngữ bao gồm các ký hiệu về các quy tắc liên kết các ký hiệu (các luật cú pháp) để tạo thành các câu (công thức) trong ngôn ngữ.
- Ngữ nghĩa của ngôn ngữ cho phép xác định ý nghĩa của các câu trong một miền nào đó của thế giới hiện thực.

Ngoài hai thành phần cú pháp và ngữ nghĩa, ngôn ngữ biểu diễn tri thức cần được cung cấp *cơ chế suy diễn*. Một luật suy diễn cho phép suy ra một công thức từ một tập các công thức.

Chương trình trí tuệ nhân tạo được cấu tạo từ hai thành phần là ***cơ sở tri thức*** (knowledge base) và ***động cơ suy diễn*** *(inference engine).*

- **Cơ sở tri thức**: là tập hợp các tri thức liên quan đến vấn đề mà chương trình quan tâm giải quyết.
- **Động cơ suy diễn**: là phương pháp vận dụng tri thức trong cơ sở tri thức để giải quyết vấn đề.



Hình 4.2: Mô hình biểu diễn tri thức

Cơ sở tri thức cũng gặp phải những vấn đề như sự trùng lắp, thừa, mâu thuẫn. Vì vậy, bên cạnh việc biểu diễn tri thức, ta còn phải có các kỹ thuật để loại bỏ những tri thức trùng lắp, thừa hoặc mâu thuẫn.

4.2.4 Logic vị từ

Biểu diễn tri thức bằng mệnh đề gặp phải một trở ngại cơ bản là ta không thể can thiệp vào cấu trúc của một mệnh đề. Hay nói một cách khác là mệnh đề *không có cấu trúc*. Điều này làm hạn chế rất nhiều thao tác suy luận. Do đó, người ta đã đưa vào khái niệm vị từ và lượng từ (∀ - với mọi, ∃ - tồn tại) để tăng cường tính cấu trúc của một mệnh đề.

Trong logic vị từ, một mệnh đề *được cấu tạo bởi* hai thành phần là *các đối tượng tri thức* và *mối liên hệ giữa chúng (gọi là vị từ).* Các mệnh đề sẽ được biểu diễn dưới dạng:

Vị từ (⟨đối tượng 1⟩, ⟨đối tượng 2⟩, ..., ⟨đối tượng n⟩)

Logic vị từ là sự mở rộng của logic mệnh đề bằng cách đưa vào các khái niệm vị từ và các lượng từ tồn tại và phổ dụng (∀ - với mọi, ∃ - tồn tại)

Vị từ là một khẳng định $p(x, y,...)$ trong đó có chứa một số biến x, y,... lấy giá trị trong những tập hợp cho trước A, B,... sao cho $p(x, y,...)$ không phải là mệnh đề nhưng nếu thay x, y,... bằng những phần tử cố định tùy ý $a \in A$, $b \in B$,... ta sẽ được một mệnh đề. Các biến x, y, ... được gọi là biến tự do của vị từ.

Ký hiệu

Logic vị từ sử dụng các loại ký hiệu sau đây.

- Hằng: a, b, c, A_n, B_m...
- Biến: x, y, z, u, v, w,...
- Vị từ: P, Q, R, S, ...
- Hàm: f, g, h, sin, cos, ...
- Toán tử kết nối logic: $\wedge$ (hội), $\vee$ (tuyển), $\neg$(phủ định), $\rightarrow$ (kéo theo), $\leftrightarrow$ (kéo theo hai chiều).
- Các lượng từ: $\forall$ (với mọi), $\exists$ (tồn tại).
- Các ký hiệu ngăn cách: dấu phẩy, dấu mở và đóng ngoặc.
- Các hạng thức (term) là các biểu thức mô tả các đối tượng được xác định như sau.
  - Các ký hiệu hằng và các ký hiệu biến là hạng thức.
  - Nếu $t_1, t_2, t_3, ..., t_n$ là n hạng thức và f là một ký hiệu hàm n biến thì $f(t_1, t_2, ..., t_n)$ là hạng thức. Một hạng thức không chứa biến được gọi là một *hạng thức cụ thể* (ground term).

Ví dụ 4.3: K_n là một ký hiệu hằng, f(x) là ký hiệu hàm một biến $f(K_n)$ thì K_n là một hạng thức cụ thể

- Các công thức phân tử (câu đơn) được xác định như sau.
  - Các ký hiệu vị từ không biến (các ký hiệu mệnh đề) là câu đơn.
  - Nếu $t_1, t_2, t_3, ..., t_n$ là n hạng thức và p là vị từ của n biến thì $p(t_1, t_2, ..., t_n)$ là câu đơn.

Ví dụ 4.4: Hà là một ký hiệu hằng, Love là một vị từ của hai biến, husband là hàm của một biến, thì Love (Hà, husband(Hà)) là một câu đơn

Các công thức :

Từ công thức phần tử, sử dụng các kết nối logic và các lượng từ để xây dựng nên các công thức (các câu). Các công thức được xác định như sau:

- Các công thức phân tử là công thức.
- Nếu P và Q là các công thức, thì các biểu thức $(P \wedge Q)$, $(P \vee Q)$, $(\neg P)$, $(P \rightarrow H)$, $(P \leftrightarrow Q)$ là công thức.
- Nếu P là một công thức và x là biến thì các biểu thức: $(\forall x, P)$, $(\exists x, P)$ là công thức.

Các công thức không phải là công thức phân tử sẽ được gọi là các câu phức hợp. Các công thức không chứa biến sẽ được gọi là *công thức cụ thể*. Khi viết các công thức ta sẽ bỏ đi các dấu ngoặc không cần thiết, chẳng hạn các dấu ngoặc ngoài cùng.

Một công thức là công thức phân tử hoặc phủ định của công thức phân tử được gọi là *literal*. Một công thức là tuyển của các literal sẽ được gọi là *câu tuyển*.

Trong công thức $(\forall x, P)$, hoặc $\exists x, P$ trong đó P là một công thức nào đó, thì mỗi xuất hiện của biến x trong công thức P được gọi là *xuất hiện buộc*. Một công thức mà tất cả các biến đều là xuất hiện buộc thì được gọi là *công thức đóng*.

4.3 THUẬT GIẢI VƯƠNG HẠO

Bước 1:

Phát biểu lại giả thiết và kết luận của vấn đề theo dạng chuẩn sau:

$GT_1, GT_2, ..., GTn \rightarrow KL_1, KL_2, ..., KLm$, *trong đó các GTi và KLi là các mệnh đề được xây dựng từ các biến mệnh đề và 3 phép nối cơ bản:* $\wedge, \vee, \neg$

Bước 2: *Chuyển về các GTi và KLi có dạng phủ định.*

Ví dụ 4.5: $p \vee q, \neg (r \wedge s), \neg g, p \vee r \rightarrow s, \neg p$

$\Rightarrow p \vee q, p \vee r, p \rightarrow (r \wedge s), g, s$

Bước 3:

- *Nếu GTi có phép* $\wedge$ *thì thay thế phép* $\wedge$ *bằng dấu ","*

- *Nếu KLi có phép $\vee$ thì thay thế phép $\vee$ bằng dấu ","*

  Ví dụ 4.6: $p \wedge q, r \wedge (\neg p \vee s) \rightarrow \neg q, \neg s$

  $\Rightarrow p, q, r, \neg p \vee s \rightarrow \neg q, \neg s$

  Bước 4:

- *Nếu GTi có phép $\vee$ thì tách thành hai dòng con.*
- *Nếu ở KLi có phép $\wedge$ thì tách thành hai dòng con.*

  Ví dụ 4.7: $p, \neg p \vee q \rightarrow q$

  $p, \neg p \rightarrow q$ $\qquad$ $p, q \rightarrow q$

  Bước 5:

- *Một dòng được chứng minh nếu tồn tại chung một mệnh đề ở cả hai phía.*

  Ví dụ 4.8: $p, \mathbf{q} \rightarrow \mathbf{q}$ được chứng minh

  $p, \neg p \rightarrow q \Rightarrow \mathbf{p} \rightarrow \mathbf{p}, q$

  Bước 6:

- *Nếu một dòng không còn phép nối $\wedge$ hoặc $\vee$ ở cả hai vế và ở 2 vế không có chung một biến mệnh đề thì dòng đó không được chứng minh.*
- *Một vấn đề được chứng minh nếu tất cả dòng dẫn xuất từ dạng chuẩn ban đầu đều được chứng minh.*

4.4 HỢP GIẢI

- Biến mệnh đề là một phát biểu đơn (câu đơn) có giá trị đúng hoặc sai và thường được kí hiệu bởi các kí tự thường p, q, ...
- Đơn tử (Literal): là 1 biến mệnh đề hoặc phủ định của biến mệnh đề
- Dạng tuyển chuẩn (CNF – Conjunctive Normal Form) là phát biểu dạng ($p_1 \vee p_2 \vee ... \vee p_m$) trong đó p_i là các đơn tử.
- Hợp giải chỉ làm việc trên các phát biểu (các mệnh đề) có dạng tuyển chuẩn.

4.4.1 Qui trình INDO

1. I (Implication out): loại bỏ phép kéo theo, có thể áp dụng tương đương logic ($(p \rightarrow q) \cong (\neg p \vee q)$) để loại bỏ.
2. N (Negative in): đưa dấu phủ định vào sâu bên trong, có thể sử dụng luật De Morgan.
3. D (Disjunctions out): dùng luật phân phối để đưa dấu tuyển $\vee$ vào sâu hơn dấu hội $\wedge$.
4. (Operator out): loại bỏ hội $\wedge$.

Ví dụ 4.9: Chuyển phát biểu sau về dạng tuyển chuẩn

$(p \rightarrow q) \wedge \neg[(q \rightarrow r) \vee (r \rightarrow t)]$

I: $(\neg p \vee q) \wedge \neg[(\neg q \vee r) \vee (\neg r \vee t)]$

N: $(\neg p \vee q) \wedge [\neg(\neg q \vee r) \wedge \neg (\neg r \vee t)]$

N: $(\neg p \vee q) \wedge [(q \wedge \neg r) \wedge (r \wedge \neg t)]$

$(\neg p \vee q) \wedge q \wedge \neg r \wedge r \wedge \neg t$

O: $\{\neg p \vee q, q, \neg r, r, \neg t\}$

Ví dụ 4.10: chuyển phát biểu sau về dạng tuyển chuẩn

$(p \wedge q \rightarrow r) \wedge \neg[(p \rightarrow r \vee t) \wedge (r \rightarrow q)]$

$[\neg(p \wedge q) \vee r] \wedge \neg[(\neg p \vee r \vee t) \wedge (\neg r \vee q)]$

$(\neg p \vee \neg q \vee r) \wedge [(p \wedge \neg r \wedge \neg t) \vee (r \wedge \neg q)]$

$(\neg p \vee \neg q \vee r) \wedge (p \vee r) \wedge (p \vee \neg q) \wedge (\neg r \vee r) \wedge (\neg r \vee \neg q) \wedge (\neg t \vee r) \wedge (\neg t \vee \neg q)$

Dạng tuyển chuẩn là $\{\neg p \vee \neg q \vee r, p \vee r, p \vee \neg q, \neg r \vee r, \neg r \vee \neg q, \neg t \vee r, \neg t \vee \neg q\}$

4.4.2 Qui tắc

$p \vee q$

$\neg q \vee r$

$\therefore p \vee r$

Hợp giải dựa vào phương pháp chứng minh phản chứng. Giả sử ta muốn chứng minh $P_1 \wedge P_2 \wedge .. \wedge P_n \rightarrow Q$ là đúng, ta cần chứng minh $P_1 \wedge P_2 \wedge .. \wedge P_n \wedge \neg Q$ là sai! Nghĩa là khi thêm $\neg Q$ vào vế trái của phát biểu sẽ tạo ra mâu thuẫn!

4.4.3 **4.4.3. Thuật toán**

Bước 1: Lấy phủ định kết luận chuyển qua vế trái, danh sách các mệnh đề có được được gọi là cơ sở tri thức (KB - Knowledge Base).

Bước 2: Chuyển các các mệnh đề về dạng CNF.

Bước 3: Lặp cho đến khi không thể áp dụng qui tắc hợp giải

- *Nếu trong KB có 2 mệnh đề phủ định nhau (p, ¬p) thì trả về thành công (True).*
- *Nếu trong KB có 2 mệnh đề chứa các đơn tử phủ định nhau thì áp dụng qui tắc hợp giải.*

Bước 4: Trả về thất bại (False)

Ví dụ 4.11: Cho biết suy luận sau đúng hay sai?

$[(p \rightarrow q) \wedge (q \rightarrow r) \wedge (r \rightarrow t)] \rightarrow (p \rightarrow t)$

$\{\neg p \vee q, \neg q \vee r, \neg r \vee t \rightarrow \neg p \vee t\}$

$\{\neg p \vee q, \neg q \vee r, \neg r \vee t, p , \neg t\}$

Ta thấy 2 mệnh đề $\neg p \vee q$ và $\neg q \vee r$ có q và $\neg q$ mâu thuẫn.

Áp dụng qui tắc ta được $\neg p \vee r$, ta được: $\{\neg p \vee q, \neg q \vee r, \neg r \vee t, p, \neg t, \neg p \vee r\}$

$\{\neg p \vee q, \neg q \vee r, \neg r \vee t, p, \neg t, \neg p \vee r\} \Rightarrow \{\neg p \vee q, \neg q \vee r, \neg r \vee t, p , \neg t, \neg p \vee r, q\}$

$\{\neg p \vee q, \neg q \vee r, \neg r \vee t, p, \neg t, \neg p \vee r, q\}$

$\Rightarrow \{\neg p \vee q, \neg q \vee r, \neg r \vee t, p , \neg t, \neg p \vee r, q, \neg q \vee t\}$

$\Rightarrow \{\neg p \vee q, \neg q \vee r, \neg r \vee t, p , \neg t, \neg p \vee r, q, \neg q \vee t\}$

$\Rightarrow \{\neg p \vee q, \neg q \vee r, \neg r \vee t, p , \neg t, \neg p \vee r, q, \neg q \vee t, t\}$

$\Rightarrow \{\neg p \vee q, \neg q \vee r, \neg r \vee t, p , \neg t, \neg p \vee r, q, \neg q \vee t, t\}$

$\Rightarrow \{\}$

Ví dụ 4.12: Cho biết phát biểu sau là đúng hay sai?

$[(p \vee q \rightarrow r) \wedge (r \rightarrow s) \wedge (s \rightarrow t) \wedge \neg t] \rightarrow [\neg(p \rightarrow q)]$

$\Rightarrow$ I: $[(\neg(p \vee q) \vee r) \wedge (\neg r \vee s) \wedge (\neg s \vee t) \wedge \neg t] \rightarrow \neg(\neg p \vee q)$

$\Rightarrow$ N: $[((\neg p \wedge \neg q) \vee r) \wedge (\neg r \vee s) \wedge (\neg s \vee t) \wedge \neg t] \rightarrow (p \wedge \neg q)$

$\Rightarrow$ D: $[(\neg p \vee r) \wedge (\neg q \vee r) \wedge (\neg r \vee s) \wedge (\neg s \vee t) \wedge \neg t] \rightarrow (p \wedge \neg q)$

$\Rightarrow$ O: $\{\neg p \vee r, \neg q \vee r, \neg r \vee s, \neg s \vee t, \neg t \rightarrow p \wedge \neg q\}$

Nhận xét

- Có thể thấy, hợp giải có khả năng bùng nổ tổ hợp khi số lượng các cặp mệnh đề có chứa mâu thuẫn lớn.
- Một cách tiếp cận đơn giản hơn về mặt áp dụng là hợp giải tuyến tính.

4.4.4 Hợp giải tuyến tính

Không mất tính tổng quát, giả sử cơ sở tri thức (KB - Knowledge Base) ban đầu không chứa mâu thuẫn. Khi đó, nếu phát biểu là đúng thì khi thêm phủ định của kết luận vào sẽ gây nên mâu thuẫn. Điều đó dẫn đến một cách thức áp dụng hiệu quả hơn đó là hợp giải tuyến tính.

Sự khác nhau chính giữa hợp giải thông thường và tuyến tính ở 2 vấn đề sau:

1. Hợp giải tuyến tính luôn bắt đầu từ các mệnh đề của phần kết luận chuyển sang trong cơ sở tri thức (KB - Knowledge Base).
2. HGTT sử dụng mệnh đề kết quả của bước 3b để "hợp giải" với các mệnh đề còn lại. Nếu con đường này thất bại thì quay lui áp dụng với các con đường gần nó nhất.

Vậy HGTT gần gũi với khái niệm đệ qui quay lui!

Ví dụ 4.13: Chứng minh mệnh đề sau

$[(p \vee q \rightarrow r) \wedge (r \rightarrow s) \wedge (s \rightarrow t) \wedge \neg t] \rightarrow [\neg(p \rightarrow q)]$

$[[(\neg p \wedge \neg q) \vee r] \wedge (\neg r \vee s) \wedge (\neg s \vee t) \wedge \neg t] \rightarrow (p \wedge \neg q)$

$[(\neg p \vee r) \wedge (\neg q \vee r) \wedge (\neg r \vee s) \wedge (\neg s \vee t) \wedge \neg t] \rightarrow (p \wedge \neg q)$

$\{\neg p \vee r, \neg q \vee r, \neg r \vee s, \neg s \vee t, \neg t, \neg p \vee q\}$

{¬p ∨ r, ¬q ∨ r, ¬r ∨ s, ¬s ∨ t, ¬t, ¬p ∨ q}

¬p ∨ q, ¬q ∨ r => ¬p ∨ r

¬p ∨ r, ¬r ∨ s => ¬p ∨ s

¬p ∨ s, ¬s ∨ t => ¬p ∨ t

¬p ∨ t, ¬t => ¬p

=> Thất bại

4.5 MẠNG NGỮ NGHĨA

4.5.1 Khái niệm

Mạng ngữ nghĩa là một phương pháp biểu diễn tri thức đầu tiên và cũng là phương pháp dễ hiểu nhất đối với chúng ta. Phương pháp này sẽ biểu diễn tri thức dưới dạng một đồ thị, trong đó đỉnh là các đối tượng (khái niệm) còn các cung cho biết mối quan hệ giữa các đối tượng (khái niệm) này.

Biểu diễn mạng ngữ nghĩa:

Mạng ngữ nghĩa là đồ thị có hướng mở rộng: G = (V, R), trong đó: V = {x_1, x_2, …, x_n} là tập các đỉnh, R = {r_1, r_2, …, r_m} là tập các quan hệ nhiều ngôi trên V.

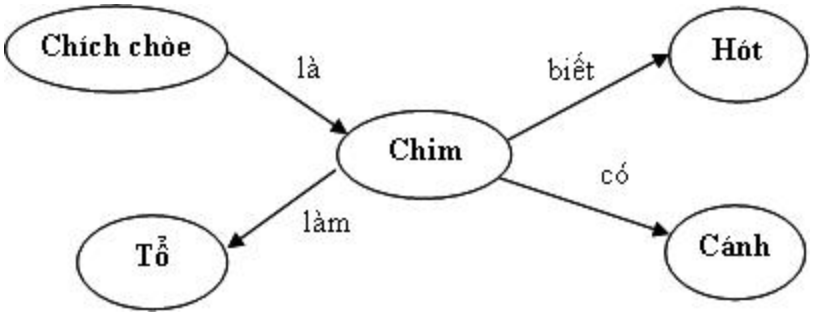
Đặc điểm của mạng ngữ nghĩa trực quan, có tính kế thừa và lan truyền thông tin; khó kiểm tra tính phi mâu thuẫn, …

Ứng dụng của mạng ngữ nghĩa: trong xử lý ngôn ngữ tự nhiên (dễ dàng hơn trong việc hiểu ngữ nghĩa của câu, do đó có thể tiến hành việc đọc và dịch các bài text), hệ chuyên gia, giải các bài toán thông minh (chẳng hạn, giải bài toán hệ thức lượng trong tam giác).

Ví dụ 4.14: Giữa các khái niệm *chích chòe*, *chim*, *hót*, *cánh*, *tổ* có một số mối quan hệ như sau:

- Chích chòe là một loài chim.
- Chim biết hót
- Chim có cánh
- Chim sống trong tổ

Các mối quan hệ này sẽ được biểu diễn trực quan bằng một đồ thị như sau:



Hình 4.1: Biểu diễn quan hệ giữa các đối tượng bằng đồ thị

Do mạng ngữ nghĩa là một loại đồ thị cho nên nó thừa hưởng được tất cả những mặt mạnh của đồ thị. Chúng ta có thể dùng những thuật toán của đồ thị trên mạng ngữ nghĩa như thuật toán tìm liên thông, tìm đường đi ngắn nhất, ... để thực hiện các cơ chế suy luận. Điểm đặc biệt của mạng ngữ nghĩa so với đồ thị thông thường chính là việc gán một ý nghĩa (*có, làm, là, biết, ...*) cho các cung. Trong đồ thị tiêu chuẩn, việc có một cung nối giữa hai đỉnh chỉ cho biết có sự *liên hệ* giữa hai đỉnh đó và tất cả các cung trong đồ thị đều biểu diễn cho cùng một loại liên hệ. Trong mạng ngữ nghĩa, cung nối giữa hai đỉnh còn cho biết giữa hai khái niệm tương ứng có sự liên hệ *như thế nào*. Việc gán ngữ nghĩa vào các cung của đồ thị đã giúp giảm bớt được số lượng đồ thị cần phải dùng để biểu diễn các mối liên hệ giữa các khái niệm.

4.5.2 Ưu điểm và nhược điểm của mạng ngữ nghĩa

Ưu điểm

- Mạng ngữ nghĩa rất linh động, ta có thể dễ dàng thêm vào mạng các đỉnh hoặc cung mới để bổ sung các tri thức cần thiết.
- Mạng ngữ nghĩa có tính trực quan cao nên rất dễ hiểu.
- Mạng ngữ nghĩa cho phép các đỉnh có thể thừa kế các tính chất từ các đỉnh khác thông qua các cung loại "là", từ đó, có thể tạo ra các liên kết "ngầm" giữa những đỉnh không có liên kết trực tiếp với nhau.
- Mạng ngữ nghĩa hoạt động khá tự nhiên theo cách thức con người ghi nhận thông tin.

Nhược điểm

- Chưa có một chuẩn nào quy định các giới hạn cho các đỉnh và cung của mạng. Nghĩa là có thể gán ghép bất kỳ khái niệm nào cho đỉnh hoặc cung!

- Tính thừa kế trên mạng sẽ có thể dẫn đến nguy cơ mâu thuẫn trong tri thức.

Hầu như không thể biển diễn các tri thức dạng thủ tục bằng mạng ngữ nghĩa vì các khái niệm về thời gian và trình tự không được thể hiện tường minh trên mạng ngữ nghĩa.

Ví dụ 4.15: Giải bài toán tam giác tổng quát

Một số bài toán thông thường về tam giác như "*Cho 3 cạnh của tam giác, tính chiều dài các đường cao*", "*Cho góc a, b và cạnh AC. Tính chiều dài trung tuyến*", ... Có một chương trình tổng quát có thể tự động giải được *tất cả* những bài toán tam giác thuộc loại này không? Có 22 yếu tố liên quan đến cạnh và góc của tam giác. Để xác định một tam giác hay để xây dựng một 1 tam giác ta cần có 3 yếu tố trong đó phải có yếu tố cạnh. Như vậy có khoảng C^3_{22} -1 cách để xây dựng hay xác định một tam giác.

Để giải bài toán tam giác bằng công cụ mạng ngữ nghĩa, ta phải sử dụng khoảng 200 đỉnh để chứa công thức và khoảng 22 đỉnh để chứa các yếu tố của tam giác. Mạng ngữ nghĩa cho bài toán này có cấu trúc như sau:

Đỉnh của đồ thị bao gồm hai loại:

- Đỉnh chứa công thức (ký hiệu bằng hình chữ nhật)
- Đỉnh chứa yếu tố của tam giác (ký hiệu bằng hình tròn)

Cung chỉ nối từ đỉnh hình tròn đến đỉnh hình chữ nhật cho biết yếu tố tam giác xuất hiện trong công thức nào *(không có trường hợp cung nối giữa hai đỉnh hình tròn hoặc cung nối giữa hai đỉnh hình chữ nhật).*

Trong một công thức liên hệ giữa n yếu tố của tam giác, ta giả định rằng nếu đã biết giá trị của n-1 yếu tố thì sẽ tính được giá trị của yếu tố còn lại. Chẳng hạn như trong công thức tổng 3 góc của tam giác bằng 180^0 thì khi biết được hai góc, ta sẽ tính được góc còn lại.

4.5.3 Cơ chế suy diễn

Bước 1: *Kích hoạt những* ***đỉnh hình tròn*** *đã cho ban đầu (những yếu tố đã có giá trị)*

Bước 2: *Lặp lại bước sau cho đến khi kích hoạt được tất cả những đỉnh ứng với những yếu tố cần tính hoặc không thể kích hoạt được bất kỳ đỉnh nào nữa.*

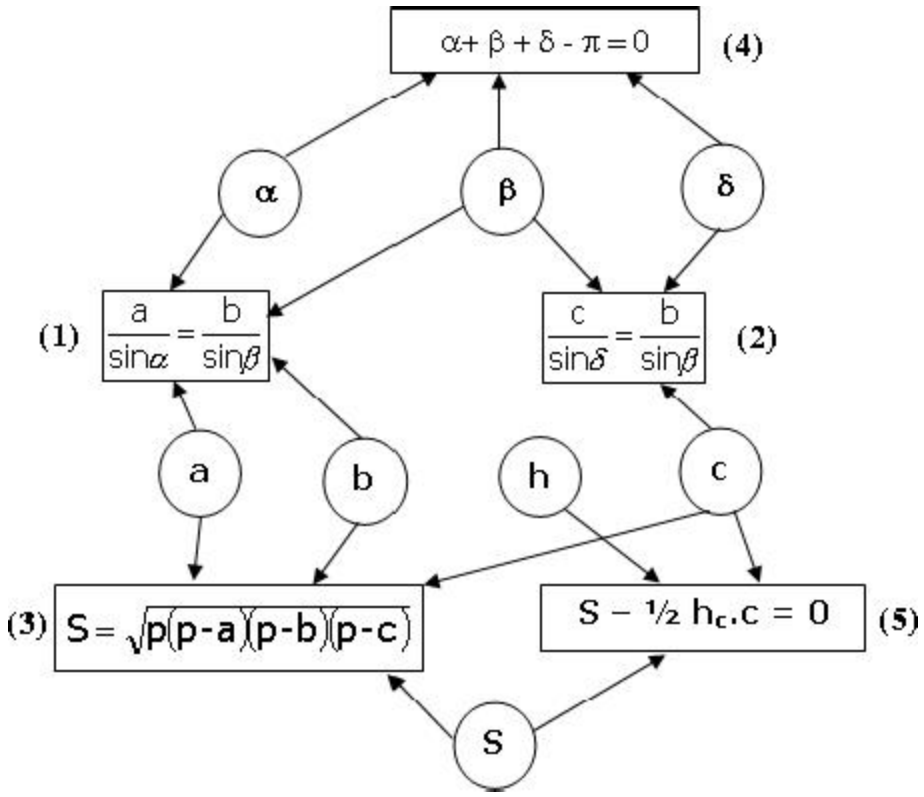
*Nếu một đỉnh hình chữ nhật có cung nối với **n** đỉnh hình tròn mà **n-1** đỉnh hình tròn đã được kích hoạt **thì** kích hoạt đỉnh hình tròn còn lại (và tính giá trị đỉnh còn lại này thông qua công thức ở đỉnh hình chữ nhật).*

Giả sử ta có mạng ngữ nghĩa để giải bài toán tam giác như Hình 4.3.

Mô hình tam giác có các công thức

1. $\alpha + \beta + \gamma = \pi$
2. $2p = a + b + c$
3. $h_a = b*\sin\gamma = c*\sin\beta$
4. $h_b = a*\sin\gamma = c*\sin\alpha$
5. $h_c = a*\sin\beta = b*\sin\alpha$
6. $S = \frac{1}{2} h_a*a = \frac{1}{2} h_b*b = \frac{1}{2} h_c*c$

Trong đó: α, β, γ là 3 góc trong của tam giác; a, b, c là 3 cạnh của tam giác.



Hình 4.3: Sơ đồ giải bài toán tam giác

Ví dụ 4.16: "Cho hai góc α, β và chiều dài cạnh a của tam giác. Tính chiều dài đường cao hC". Với mạng ngữ nghĩa đã cho trong hình trên, các bước thi hành của thuật toán như sau:

- **Bắt đầu: đỉnh α, β, a** của đồ thị được kích hoạt.
- Công thức (1) được kích hoạt (vì **α, β, a** được kích hoạt). Từ công thức (1) tính được cạnh b. Đỉnh b được kích hoạt.
- Công thức (4) được kích hoạt (vì **α, β**). Từ công thức (4) tính được góc δ
- Công thức (2) được kích hoạt (vì 3 đỉnh **β, δ , b** được kích hoạt). Từ công thức (2) tính được cạnh **c**. Đỉnh **c** được kích hoạt.
- Công thức (3) được kích hoạt (vì 3 đỉnh a, b, c được kích hoạt) . Từ công thức (3) tính được diện tích **S.** Đỉnh S được kích hoạt.
- Công thức (5) được kích hoạt (vì 2 đỉnh S, c được kích hoạt). Từ công thức (5) tính được hC. Đỉnh hC được kích hoạt.
- Giá trị hC đã được tính. Thuật toán kết thúc.

Về mặt chương trình, ta có thể cài đặt mạng ngữ nghĩa giải bài toán tam giác bằng một mảng hai chiều A trong đó:

- ***Cột***: ứng với công thức. Mỗi cột ứng với một công thức tam giác khác nhau (đỉnh hình chữ nhật).
- ***Dòng***: ứng với yếu tố tam giác. Mỗi dòng ứng với một yếu tố tam giác khác nhau (đỉnh hình tròn).
- Phần tử A[i, j] = **-1** nghĩa là trong công thức ứng với cột **j** có yếu tố tam giác ứng với cột **i**. Ngược lại A[i,j] = **0.**
- Để thực hiện thao tác "kích hoạt" một đỉnh hình tròn, ta đặt giá trị của toàn dòng ứng với yếu tố tam giác bằng 1.
- Để kiểm tra xem một công thức đã có đủ n-1 yếu tố hay chưa (nghĩa là kiểm tra điều kiện "*đỉnh hình chữ nhật có cung nối với n đỉnh hình tròn mà n-1 đỉnh hình tròn đã được kích hoạt*"), ta chỉ việc lấy **hiệu** giữa **tổng** số ô có giá trị bằng 1 và **tổng** số ô có giá trị **-1** trên cột ứng với công thức cần kiểm tra. Nếu kết quả bằng **n**, thì công thức đã có đủ n-1 yếu tố.

- Mảng biểu diễn mạng ngữ nghĩa ban đầu

	(1)	(2)	(3)	(4)	(5)
α	**-1**	0	0	**-1**	0
β	**-1**	**-1**	0	**-1**	0
δ	0	**-1**	0	**-1**	0
a	**-1**	0	**-1**	0	0
b	**-1**	**-1**	**-1**	0	0
c	0	**-1**	**-1**	0	**-1**
S	0	0	**-1**	0	**-1**
hC	0	0	0	0	**-1**

Khởi đầu: đỉnh **α, β, a** của đồ thị được kích hoạt.

	(1)	(2)	(3)	(4)	(5)
α	**1**	0	0	**1**	0
β	**1**	**1**	0	**1**	0
δ	0	**-1**	0	**-1**	0
a	**1**	**0**	**1**	**1**	0
b	**-1**	**-1**	**-1**	0	0
c	0	**-1**	**-1**	0	**-1**
S	0	0	**-1**	0	**-1**
hC	0	0	0	0	**-1**

Trên cột **(1)**, hiệu (1+1+1 – (-1)) = 4 nên dòng **b** sẽ được kích hoạt.

	(1)	(2)	(3)	(4)	(5)
α	**1**	0	0	**1**	0
β	**1**	**1**	0	**1**	0
δ	0	**-1**	0	**-1**	0
a	**1**	**0**	**1**	**1**	0
b	**1**	**1**	**1**	0	0
c	0	**-1**	**-1**	0	**-1**
S	0	0	**-1**	0	**-1**
hC	0	0	0	0	**-1**

Trên cột (4), hiệu (1+1+1 – (-1)) = 4 nên dòng **δ** sẽ được kích hoạt.

	(1)	(2)	(3)	(4)	(5)
α	**1**	0	0	**1**	0
β	**1**	**1**	0	**1**	0
δ	0	**1**	0	**1**	0
a	**1**	**0**	**1**	**1**	0
b	**1**	**1**	**1**	0	0
c	0	**-1**	**-1**	0	**-1**
S	0	0	**-1**	0	**-1**
hC	0	0	0	0	**-1**

Trên cột (2), hiệu (1+1+1 – (1)) = 4 nên dòng **c** được kích hoạt.

	(1)	(2)	(3)	(4)	(5)
α	**1**	0	0	**1**	0
β	**1**	**1**	0	**1**	0
δ	0	**1**	0	**1**	0
a	**1**	**0**	**1**	**1**	0
b	**1**	**1**	**1**	0	0
c	0	**1**	**1**	0	**1**
S	0	0	**-1**	0	**-1**
hC	0	0	0	0	**-1**

Trên cột (3), hiệu (1+1+1 – (-1)) = 4 nên dòng **S** được kích hoạt.

	(1)	(2)	(3)	(4)	(5)
α	**1**	0	0	**1**	0
β	**1**	**1**	0	**1**	0
δ	0	**1**	0	**1**	0
a	**1**	**0**	**1**	**1**	0
b	**1**	**1**	**1**	0	0
c	0	**1**	**1**	0	**1**
S	0	0	**1**	0	**1**
hC	0	0	0	0	**-1**

Trên cột (5), hiệu (1+1 – (1)) = 3 nên dòng **hC** được kích hoạt.

Khả năng của hệ thống này không chỉ dừng lại ở việc tính ra giá trị các yếu tố cần thiết, với một chút sửa đổi, chương trình này còn có thể đưa ra cách giải hình thức của bài toán và thậm chí còn có thể chọn được cách giải hình thức tối ưu (tối ưu hiểu theo nghĩa là cách giải sử dụng những công thức đơn giản nhất). Sở dĩ có thể nói như vậy vì cách suy luận của ta trong bài toán này là *tìm kiếm theo chiều rộng*. Do đó, khi đạt đến kết quả, ta có thể có rất nhiều cách khác nhau. Để có thể chọn được giải pháp tối ưu, bạn cần phải định nghĩa được độ "phức tạp" của một công thức. Một trong những tiêu chuẩn thường được dùng là số lượng phép nhân, chia, cộng, trừ, rút căn, tính sin, cos, ... được áp dụng trong công thức. Các phép tính sin, cos và rút căn có độ phức tạp cao nhất, kế đến là nhân chia và cuối cùng là cộng trừ. Cuối cùng bạn có thể cải tiến lại phương pháp suy luận bằng cách vận dụng thuật toán A* với ước lượng **h=0** để có thể chọn ra được "đường đi" tối ưu. Ta chọn ước lượng h=0 vì hai lý do sau (1) không gian bài toán nhỏ nên ta không cần phải giới hạn độ rộng tìm kiếm (2) xây dựng một ước lượng như vậy là tương đối khó khăn, đặc biệt là làm sao để hệ thống không đánh giá quá cao h.

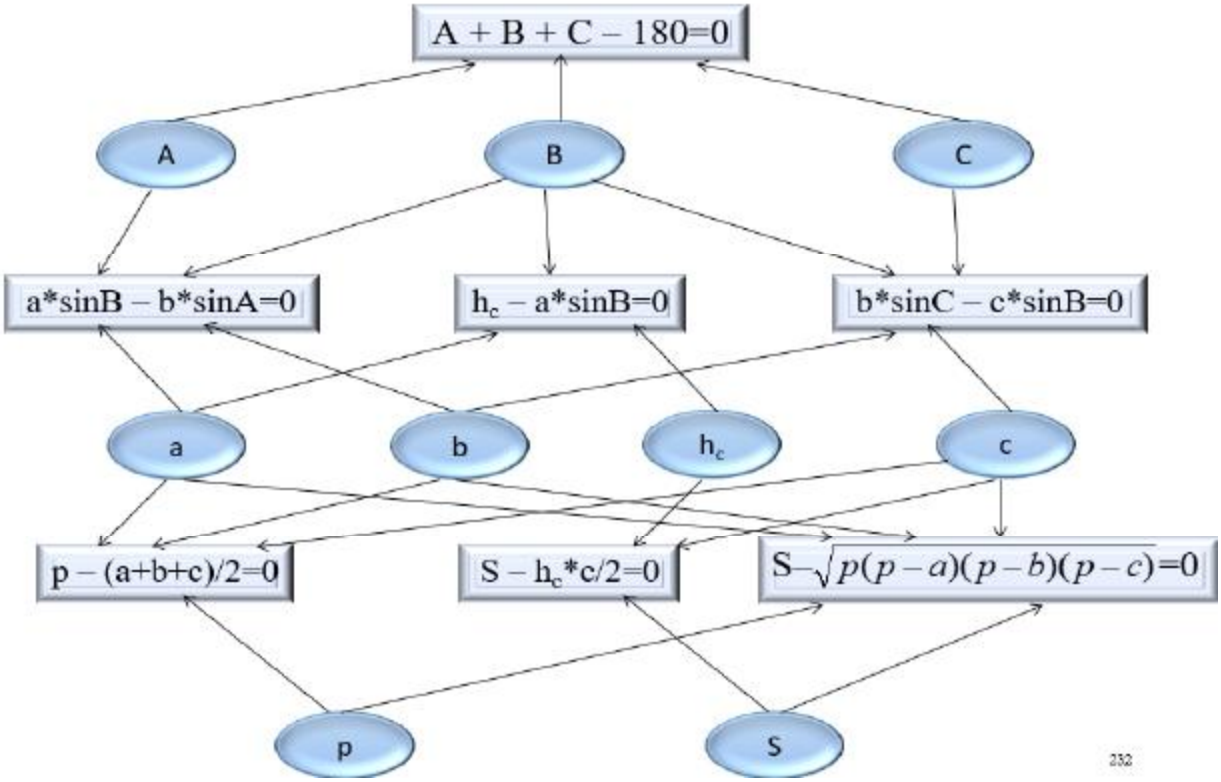
Ví dụ 4.17: Trong tam giác cho các công thức

1. $A + B + C = 180^0$
2. $a*\sin B = b*\sin A$
3. $b * \sin C = c * \sin B$
4. $p = (a + b + c)/2$
5. $h_c = a*\sin B$
6. $S = h_c*c/2$
7. $S = \sqrt{p(p-a)(p-b)(p-c)}$

Trong đó: A, B, C là 3 góc trong của tam giác; a, b, c là 3 cạnh của tam giác.

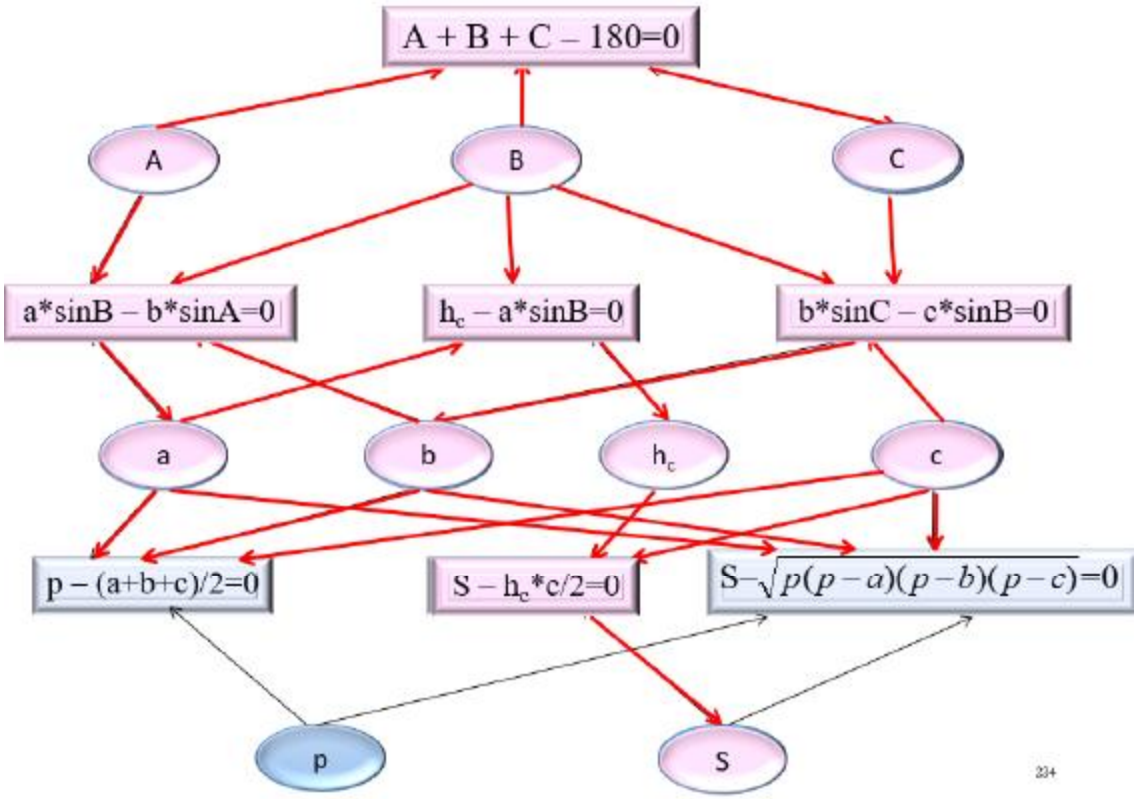
Xây dựng mô hình mạng ngữ nghĩa

Có 16 đỉnh trong đó có 9 đỉnh hình tròn (9 biến) và 7 đỉnh hình chữ nhật (7 công thức).



Cơ chế suy diễn

1. Kích hoạt các biến đã cho ban đầu.
2. Các biến được kích hoạt sẽ truyền động theo cung dẫn đến các đỉnh khác.
3. Đỉnh hình chữ nhật được kích hoạt khi có n – 1 cung trong số n cung có liên hệ với nó được kích hoạt.
4. Đỉnh hình tròn được kích hoạt khi đỉnh hình chữ nhật có cung liên hệ với nó được kích hoạt.



4.5.4 Xử lý ngôn ngữ tự nhiên

Xử lý ngôn ngữ tự nhiên là việc làm cho máy có thể nhận biết và hiểu tiếng nói và ngôn ngữ của con người.

a. Liệu máy có thể nói được như người?

Đây là bài toán tổng hợp tiếng nói, tức việc làm cho máy biết đọc các văn bản thành tiếng người. Có thể hình dung nếu ta đưa cho máy các luật phát âm các âm tiết, bài toán này sẽ là việc áp dụng các luật này vào các âm tiết trong một từ để tạo ra cách đọc từ này. Đã có nhiều hệ thống tạo ra được giọng đọc tự nhiên của con người hoặc đọc giống giọng một người nào đấy, nhất là cho các ngôn ngữ được nghiên cứu nhiều như tiếng Anh. Nhưng vẫn cần rất nhiều thời gian để làm được như vậy cho các ngôn ngữ ít được nghiên cứu như tiếng Việt, hoặc làm cho máy thể hiện được buồn vui mừng giận khi nói như người.

b. Liệu máy có thể nhận biết được tiếng người nói?

Đây là bài toán nhận dạng tiếng nói, tức việc làm cho máy biết chuyển tiếng nói của người từ microphone thành dãy các từ. Dễ thấy đây là bài toán rất khó, vì âm

thanh người nói là liên tục và các âm quyện nối vào nhau, vì mỗi người mỗi giọng, vì có các âm thanh khác nhiễu vào microphone, ... Với tiếng nói chuẩn, các hệ hiện đại cũng mới nhận dạng đúng được khoảng 60-70%. Đại thể, máy mới nhận biết tốt tiếng nói của từng người riêng biệt với lượng nhỏ từ vựng và phải 'tập nghe' với chính giọng của người đó. Với các phương pháp học thống kê trên các kho ngữ âm tốt, ta có thể sớm hy vọng vào các hệ nhận dạng tiếng nói thông minh và chính xác.

c. Liệu máy có hiểu được tiếng nói và văn bản của con người?

Hiểu ngôn ngữ là một đặc trưng tiêu biểu của trí tuệ và việc làm cho máy hiểu được ngôn ngữ là một trong vài vấn đề khó nhất của TTNT nói riêng và CNTT nói chung. Làm sao để máy tự động hiểu đúng nghĩa một câu nói bất kỳ còn là một thách thức lâu dài của ngành TTNT.

d. Dịch tự động

Liên quan đến hiểu ngôn ngữ là việc dịch tự động từ tiếng này sang tiếng khác. Việc dịch này đòi hỏi máy không những phải hiểu đúng nghĩa câu tiếng Việt mà còn phải tạo ra được câu tiếng Anh tương ứng. Nhiều dự án dịch tự động đã được theo đuổi từ hàng chục năm qua, và chắc còn phải tiếp tục nhiều chục năm nữa để có được những hệ dịch tốt. Tuy nhiên, những tiến bộ gần đây của các phương pháp dịch thống kê trên các kho ngữ liệu lớn và tốt có thể cho phép đến một ngày ai cũng nhờ máy đọc được sách báo từ hàng chục thứ tiếng.

e. Tìm kiếm thông tin trên mạng

Đây là lĩnh vực có sự chia sẻ nhiều nhất giữa TTNT và Internet, và ngày càng trở nên hết sức quan trọng. Sẽ sớm đến một ngày, mọi sách báo của con người được số hóa và để lên mạng hay các thư viện số cực lớn. Tuy nhiên nếu dùng Google để tìm kiếm thì có thể chúng ta sẽ nhận được rất nhiều tài liệu không phải thứ chúng ta muốn tìm, cũng như có nhiều tài liệu liên quan không được tìm ra.

Có ít nhất hai cách để TTNT đóng góp vào việc giải bài toán này. Một là hệ tìm kiếm cho phép đưa vào câu hỏi ở dạng ngôn ngữ tự nhiên, phân tích để hiểu nghĩa câu hỏi và có cơ chế tìm kiếm các văn bản trong thư viện theo nghĩa này. Hai là hệ tìm kiếm sẽ mô hình các từ cần tìm kiếm, mỗi mô hình là một tập hợp nhiều từ khác kèm theo phân bố xác suất của chúng theo những quy luật thống kê. Thay vì tìm kiếm trên mạng hay trong thư viện với một số từ khóa, hệ sẽ tìm kiếm với một số tập hợp từ. Với các phương pháp 'thông minh' này, ta hy vọng kết quả tìm kiếm sẽ có ý nghĩa hơn.

BÀI 5: HỌC MÁY

Học xong bài này sinh viên có thể nắm được:

- *Các khái niệm liên quan đến học máy.*
- *Kỹ thuật phân lớp (ID3, C4.5, ILA, Bayes)*
- *Kỹ thuật phân cụm (k – means, k – medoids)*

5.1 GIỚI THIỆU HỌC MÁY

Học máy (machine learning) là một nhánh của ngành trí tuệ nhân tạo. Mục tiêu của học máy là hiểu cấu trúc của dữ liệu và xử lý dữ liệu để tạo ra các mô hình (model) thực hiện các công việc cụ thể. Nếu trong các phương pháp tiếp cận truyền thống, thuật toán là một tập hợp các tri thức và quy luật được lập trình rõ ràng thì ở đây các thuật toán học máy sẽ tự học các tri thức và quy luật đó từ dữ liệu đầu vào.

Hiện nay chưa có định nghĩa thống nhất về học máy. Tuy nhiên có một vài định nghĩa máy học như sau:

- Đại học Stanford – "Học máy là một lĩnh vực trong khoa học máy tính nhằm xây dựng hệ thống có thể thực hiện các hành vi giống con người mà không cần lập trình rõ ràng cho từng hành vi cụ thể này"
- Tom Mitchell (1997) – "Máy tính được gọi là học từ kinh nghiệm (dữ liệu) E với tác vụ (dự đoán, phân lớp, gom nhóm) T và được đánh giá bởi độ đo (độ chính xác) P nếu máy tính khiến tác vụ T này cải thiện được độ chính xác P thông qua dữ liệu E cho trước"
- Ví dụ: phát hiện khuôn mặt người
  - E = quá trình "nhìn" ảnh khuôn mặt người
  - T = vùng nào trên ảnh là mặt người

- P = số lần xác định chính xác

- Arthur Samuel (1959) – "Lĩnh vực nghiên cứu giúp máy tính có khả năng tự học khi không được lập trình trước"

Ứng dụng đầu tiên của học máy là chương trình lọc thư rác (spam filter). Nó là tiền đề phát triển cho hàng trăm ứng dụng học máy khác như hệ thống khuyến nghị, nhận dạng giọng nói, nhận dạng khuôn mặt,... Hiện nay học máy đã ứng dụng trong rất nhiều lĩnh vực khác nhau như sau:

- Chơi cờ: ngày 1 tháng năm 1997, IBM Deep Blue thắng Kasparov ở ván thứ sáu, đánh dấu một bước ngoặt trong lịch sử cờ vua, một bước ngoặt của công nghệ trước ngưỡng cửa thế kỷ mới.

Hình 5.1: Cuộc thi đấu cờ vua giữa IBM Deep Blue và Đại kiện tướng Kasparov.

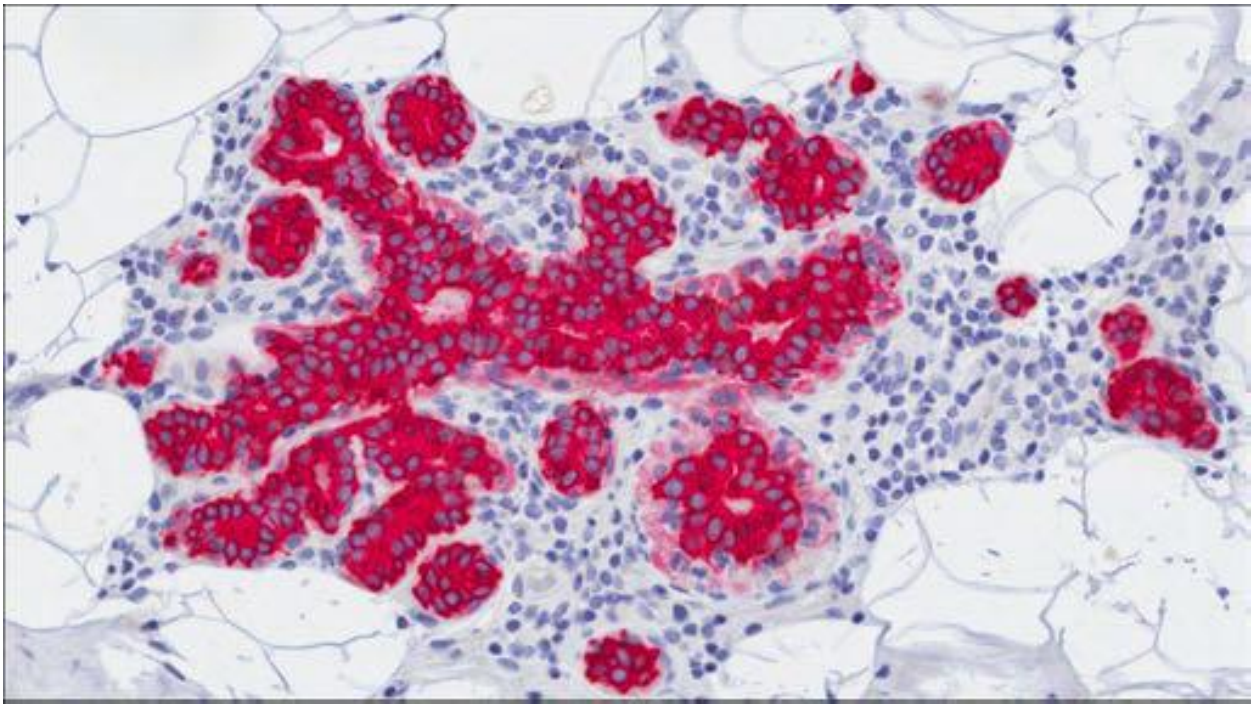
Từ ngày 9 đến ngày 15 tháng 3 năm 2016 tại Seoul (Hàn Quốc) trận đấu cờ vây gồm 5 ván giữa AlphaGo (phần mềm cờ vây máy tính được phát triển bởi Google DeepMind) và Lee Sedol (từng 18 lần vô địch thế giới).

Hình 5.2: Cuộc thi đấu cờ vây giữa AlphaGo và Đại kiện tướng Lee Sedol.

- Trong hệ thống tự động ra quyết định: lọc thư rác, phát hiện gian lận trong tài chính, ...
- Các hệ thống tự động có lập trình phức tạp: xe tự vận hành, nhận dạng ký tự quang học (OCR), nhận dạng khuôn mặt, ...

Hình 5.3: Xe tự vận hành Tesla.

- Chuẩn đoán y khoa: Chuẩn đoán bệnh qua hình ảnh như ung thư, ...



Hình 5.4: Chuẩn đoán ung thư máu qua hình ảnh.

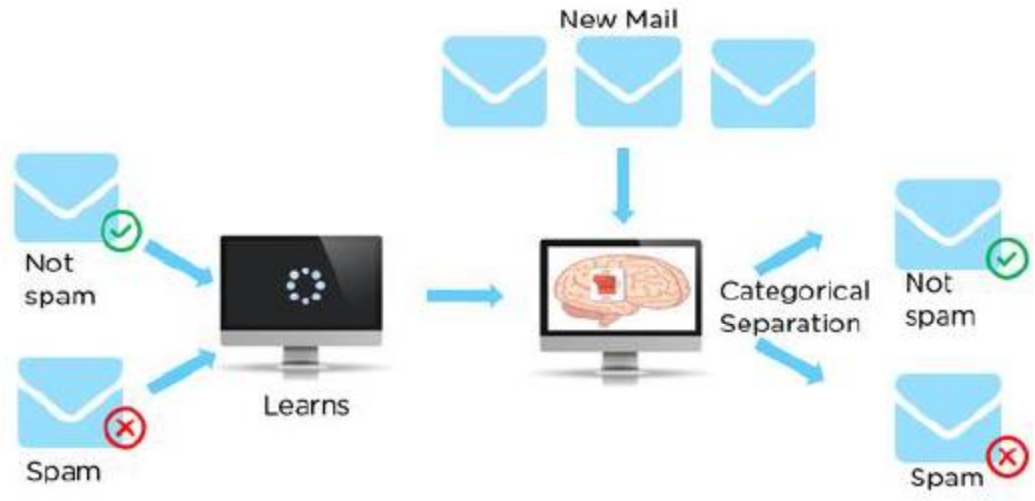
- Robot

Hình 5.5: Robot Asimo.

Các giải thuật học máy được phân ra làm hai loại chính là:

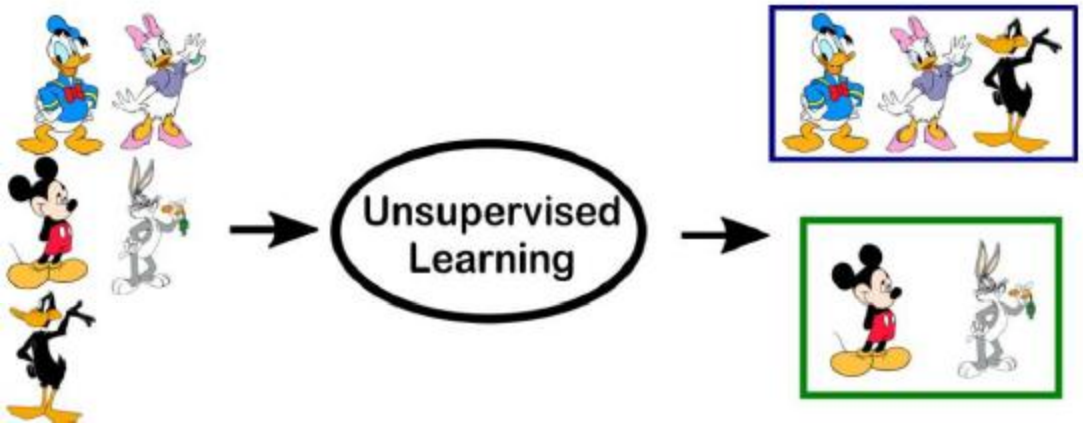
- **Học có giám sát** (*supervised learning*): là phương pháp sử dụng những dữ liệu đã được gán nhãn từ trước để suy luận ra quan hệ giữa đầu vào và đầu ra. Các dữ liệu này được gọi là dữ liệu huấn luyện và chúng là cặp các đầu vào – đầu ra. Học

có giám sát sẽ xem xét các tập huấn luyện này để từ đó có thể đưa ra dự đoán đầu ra cho đầu vào mới chưa gặp bao giờ. Chẳng hạn như dự đoán giá nhà, phân loại email.



Hình 5.6: Minh hoạ học có giám sát qua ứng dụng phân loại thư.

- **Học không giám sát** (*unsupervised learning*): khác với học có giám sát, học không giám sát sử dụng những dữ liệu chưa được gán nhãn để suy luận. Phương pháp này thường được sử dụng để tìm cấu trúc của tập dữ liệu. Tuy nhiên lại không có phương pháp đánh giá được cấu trúc tìm ra được là đúng hay sai. Chẳng hạn như gom cụm dữ liệu, luật kết hợp, giảm số chiều, ...



Hình 5.7: Minh hoạ học không có giám sát.

Ngoài ra còn có một số loại học khác như: học bán giám sát (semi – supervised learning), học tăng cường (reinforcement learning), học tiến hoá (evolutionary learning), ...

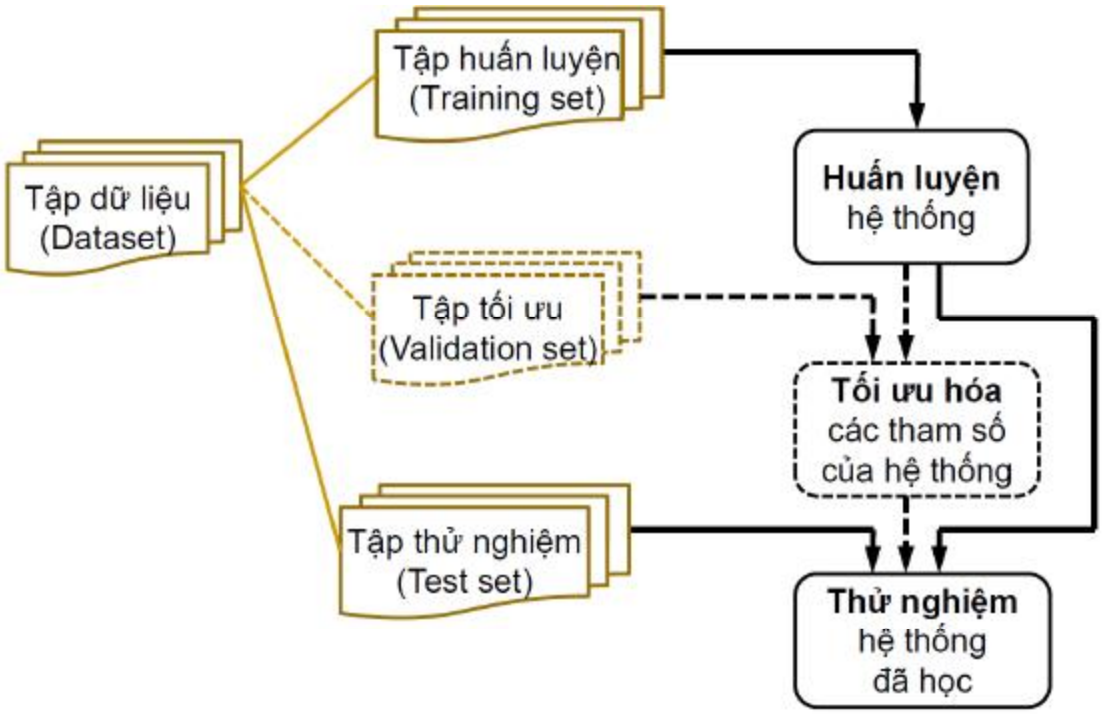
5.2 QUY TRÌNH HỌC MÁY

Để hiểu rõ quá trình học máy, ta cần hiểu rõ một số khái niệm sau:

- Mẫu dữ liệu (học) hoặc ví dụ (sample) hoặc thực thể (instance) hoặc quan sát (observation) hoặc bản ghi (record) là tên gọi đối tượng cần phân lớp hoặc gom nhóm. Chẳng hạn như khi lọc thư rác, mỗi thư được gọi là một mẫu dữ liệu.
- Các mẫu thường được mô tả bằng một tập các thuộc tính, còn được gọi là đặc trưng hoặc biến. Chẳng hạn như trong bài toán chẩn đoán bệnh, thuộc tính là những triệu chứng của người bệnh và các tham số khác: cân nặng, huyết áp, ...

Quy trình học máy chia làm hai giai đoạn

- Giai đoạn huấn luyện/học (training/learning): hệ thống phân tích dữ liệu và nhận ra sự mối quan hệ (có thể là phi tuyến hoặc tuyến tính) giữa các đối tượng dữ liệu. Kết quả của việc học có thể là phân lớp hoặc nhóm các đối tượng, tạo ra các luật, dự đoán lớp cho các đối tượng mới.
- Giai đoạn thử nghiệm (testing): mối quan hệ (các luật, lớp, ...) được tạo ra phải được kiểm nghiệm lại bằng một số hàm tính toán thực thi trên một phần của tập dữ liệu huấn luyện hoặc trên một tập dữ liệu lớn.



Hình 5.8: Quy trình học máy.

❖ **Đối với giải thuật phân lớp**

Đối với giai đoạn thử nghiệm thì dữ liệu thử nghiệm được dùng để ước lượng độ chính xác của phân lớp. Nếu độ chính xác là chấp nhận được thì có thể dùng để phân lớp các mẫu dữ liệu mới.

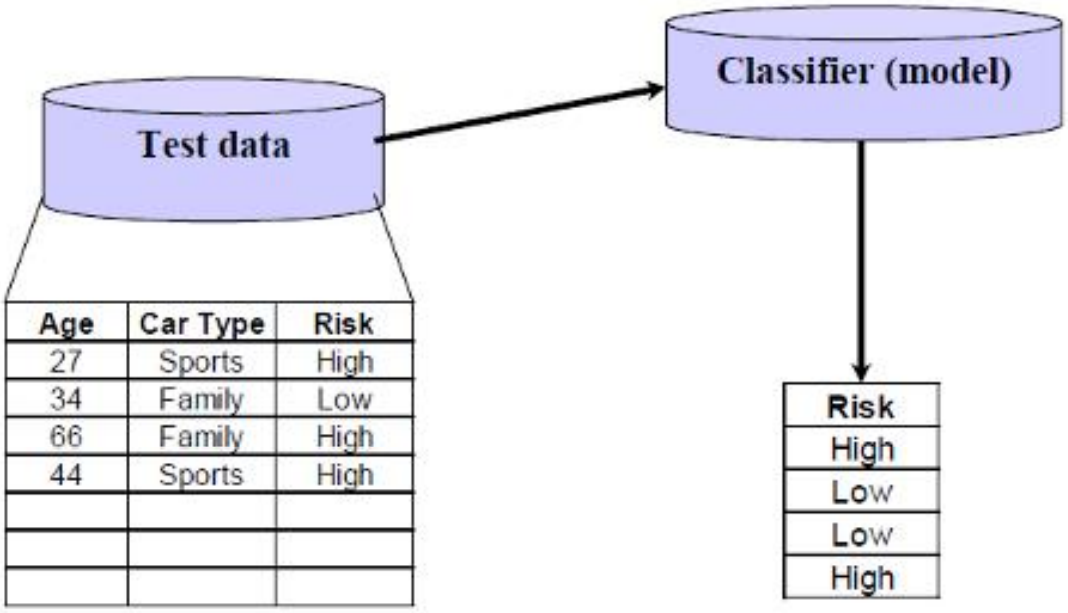
Độ chính xác (accuracy) của phân lớp trên tập thử nghiệm là phần trăm của các mẫu trong tập thử nghiệm được phân lớp đúng.

$$\text{accuracy} = \frac{\text{correctly classified test sample}}{\text{total number of test sample}}$$

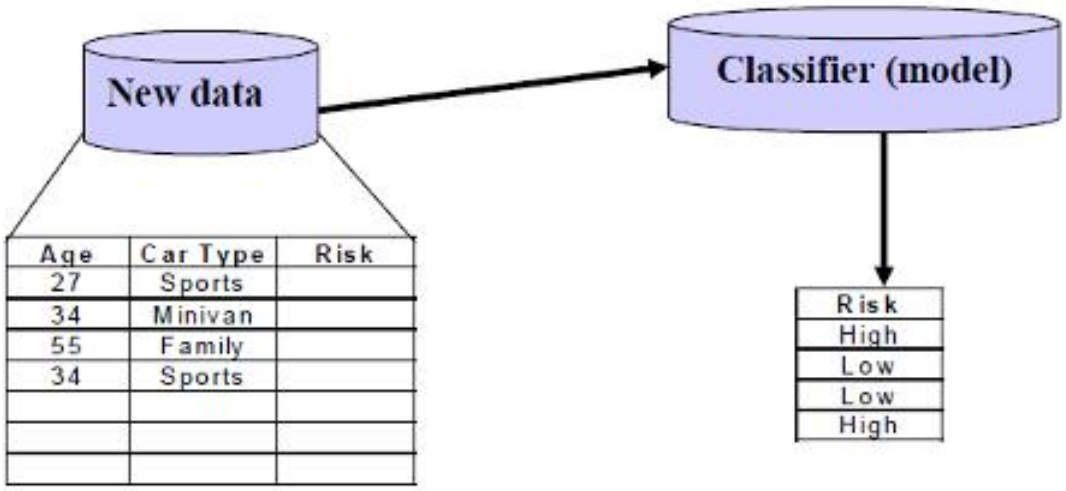
❖ **Các phương pháp tính độ chính xác:** có hai phương pháp đánh giá phổ biến là holdout và k-fold cross-validation, cả hai kỹ thuật này đều dựa trên các phân hoạch ngẫu nhiên tập dữ liệu ban đầu.

- Trong phương pháp holdout, dữ liệu dưa ra được phân chia ngẫu nhiên thành hai phần là: tập dữ liệu huấn luyện và tập dữ liệu thử nghiệm. Thông thường 2/3 dữ liệu cấp cho tập dữ liệu huấn luyện, phần còn lại cho tập dữ liệu thử nghiệm.
- Trong phương pháp *k*-fold cross validation tập dữ liệu ban đầu được chia ngẫu nhiên thành *k* tập con (fold) có kích thước xấp xỉ nhau S_1, S_2, …, S_k. Quá trình học và kiểm tra được thực hiện *k* lần. Tại lần lặp thứ *i*, S_i là tập dữ liệu thử nghiệm, các tập còn lại hợp thành tập dữ liệu huấn luyện. Có nghĩa đầu tiên việc học được thực hiện trên các tập S_2, S_3 …, S_k, sau đó kiểm tra trên tập S_1. Quá trình học tiếp tục được thực hiện trên tập S_1, S_3, S_4, …, S_k, sau đó kiểm tra trên tập S_2; và cứ thế tiếp tục. Độ chính xác là toàn bộ số phân lớp đúng từ *k* lần lặp chia cho tổng số mẫu của tập dữ liệu ban đầu.

Holdout là một kỹ thuật đơn giản để ước lượng độ chính xác, kỹ thuật này sử dụng một tập dữ liệu thử nghiệm với các mẫu đã được gán nhãn lớp. Các mẫu này được chọn ngẫu nhiên và độc lập với các mẫu trong tập dữ liệu huấn luyện. Độ chính xác của mô hình trên *tập dữ liệu thử nghiệm* đã đưa là tỉ lệ phần trăm các các mẫu trong tập dữ liệu thử nghiệm được mô hình phân lớp đúng. Nếu độ chính xác của mô hình là chấp nhận được, thì mô hình được sử dụng để phân lớp những dữ liệu tương lai, hoặc những dữ liệu mà giá trị của thuộc tính phân lớp là chưa biết.



Hình 5.9: Ước lượng độ chính xác của mô hình.



Hình 5.10: Phân lớp dữ liệu mới.

5.3 PHÂN LỚP

Phân lớp (*classification*) và dự đoán (*prediction*) là hai dạng của phân tích dữ liệu nhằm trích rút ra một mô hình mô tả các lớp dữ liệu quan trọng hay dự đoán xu hướng dữ liệu tương lai.

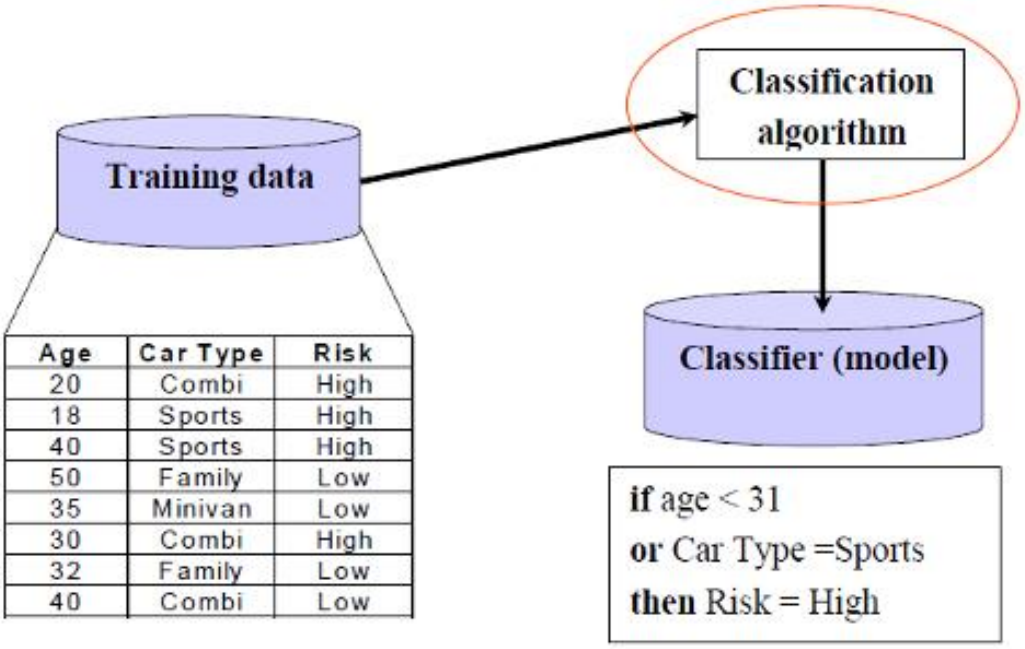
Phân lớp dự đoán giá trị của những nhãn xác định (*categorical label*) hay những giá trị rời rạc (*discrete value*), có nghĩa là phân lớp thao tác với những đối tượng dữ liệu mà có bộ giá trị là biết trước. Trong khi đó, dự đoán lại xây dựng mô hình với các hàm nhận giá trị liên tục. Chẳng hạn như mô hình phân lớp dự báo thời tiết có thể cho

biết thời tiết ngày mai là mưa, hay nắng dựa vào những thông số về độ ẩm, sức gió, nhiệt độ, ... của ngày hôm nay và các ngày trước đó. Hoặc nhờ các luật về xu hướng mua hàng của khách hàng trong siêu thị, các nhân viên kinh doanh có thể ra những quyết sách đúng đắn về lượng mặt hàng cũng như chủng loại bày bán...

Nhiệm vụ của quá trình phân lớp là thiết lập được ánh xạ giữa giá trị của các thuộc tính với các nhãn lớp thông qua việc xây dựng mô hình. Mô hình sau đó sẽ được dùng để xác định nhãn lớp cho các mẫu dữ mới không nằm trong tập mẫu ban đầu.

Đầu vào của quá trình này là một tập dữ liệu có cấu trúc được mô tả bằng các thuộc tính và được tạo ra từ tập các bộ giá trị của các thuộc tính đó. Mỗi bộ giá trị được gọi chung là một phần tử dữ liệu là *mẫu/ví dụ/đối tượng/bản ghi/trường hợp* (*case*). Mỗi phần tử dữ liệu được giả sử thuộc về một lớp định trước, lớp ở đây là giá trị của một thuộc tính được chọn làm *thuộc tính gán nhãn lớp* hay *thuộc tính phân lớp* (*class label attribute*).

Đầu ra của bước này thường là các quy tắc phân lớp dưới dạng luật dạng if-then, cây quyết định, công thức logic, hay mạng nơron.



Age	Car Type	Risk
20	Combi	High
18	Sports	High
40	Sports	High
50	Family	Low
35	Minivan	Low
30	Combi	High
32	Family	Low
40	Combi	Low

Hình 5.11: Quá trình phân lớp dữ liệu

Trong các mô hình phân lớp, cây quyết định được coi là công cụ mạnh, phổ biến và đặc biệt thích hợp với các ứng dụng khai thác dữ liệu. Thuật toán phân lớp là nhân tố trung tâm trong một mô hình phân lớp.

Phân lớp dữ liệu cũng tiến trình qua hai bước:

- **Huấn luyện**: Dữ liệu huấn luyện được phân tích bởi thuật toán phân lớp (có thuộc tính nhãn lớp) để tạo ra bộ phân lớp.
- **Phân lớp:** Dữ liệu kiểm tra được dùng để ước lượng độ chính xác của bộ phân lớp. Nếu độ chính xác là chấp nhận được thì có thể dùng bộ phân lớp để phân lớp các mẫu dữ liệu mới.

5.3.1 Thuật toán ID3

Thuật toán được phát triển bởi Quinlan (trường đại học Syney, Australia) và được công bố vào cuối thập niên 70. ID3 với khả năng lựa chọn thuộc tính tốt nhất để triển khai cây tại mỗi bước. Thuật toán sử dụng khái niệm Entropy và Information Gain để xác định thuộc tính tốt nhất cho quá trình triển khai cây.

Entropy: đại lượng dùng để đo tính thuần nhất của một tập dữ liệu, ký hiệu là E. Entropy của một tập S được tính theo công thức:

$$E(S) = \sum_{i-1}^{n} (-p_i \log_2 p_i)$$

Hoặc $E(S) = -p^+ \log_2 p^+ - p^- \log_2 p^-$

Trong đó: p_i là tỷ lệ các mẫu thuộc lớp i trên tập S các mẫu được kiểm tra, p^+ là tỷ lệ các mẫu thuộc lớp dương, p^- là tỷ lệ các mẫu thuộc lớp âm trong S

- Nếu tất cả các mẫu thành viên trong tập S đều thuộc cùng một lớp thì E(S) = 0.
- Nếu trong tập S có số mẫu phân bố đều nhau thì E(S) = 1.
- Các trường hợp còn lại 0 < E(S) < 1.

Ví dụ 5.2: Từ 14 mẫu của bảng Play-Tennis, 9 thuộc lớp dương và 5 mẫu âm (ký hiệu là [9+, 5-])

$E([9+, 5-]) = - (9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.940$

Information Gain: đại lượng dùng để đo tính hiệu quả của một thuộc tính được lựa chọn cho việc phân lớp.

$$Gain(S, A) = E(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} E(S_v)$$

Giá trị Value(A) là tập các giá trị có thể cho thuộc tính A, *và* S_v là tập con của S mà A nhận giá trị v.

Ví dụ 5.3: Cho values(Wind) = {Weak, Strong}, S = [9+, 5-]

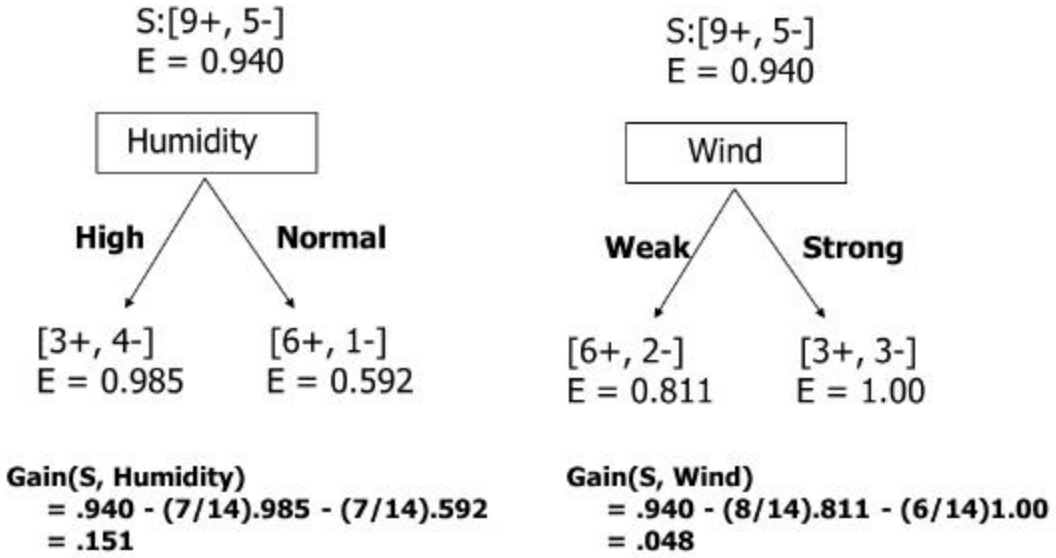
S_{weak} là nút con với trị "weak" là [6+, 2-]

S_{strong} là nút con với trị "strong" là [3+, 3-]

Gain(S, Wind) = E(S) - $\sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} E(S_v)$

= E(S) - (8/14)E(S_{weak}) - (6/14)E(S_{Strong})

= 0.940 - (8/14)0.811 - (6/14)1.00 = 0.048



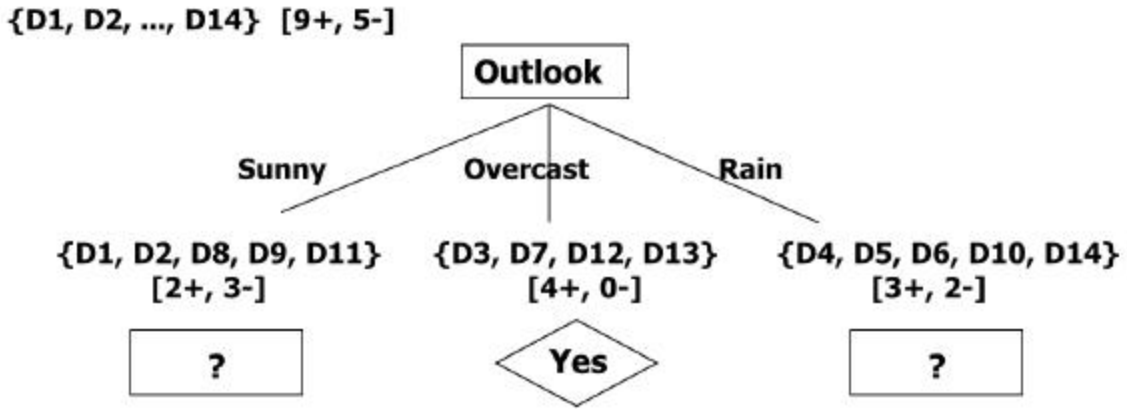
Gain(S, Humidity)
= .940 - (7/14).985 - (7/14).592
= .151

Gain(S, Wind)
= .940 - (8/14).811 - (6/14)1.00
= .048

Information Gain của tất cả các thuộc tính

- Gain (S, Outlook) = 0.246
- Gain (S, Humidity) = 0.151
- Gain (S, Wind) = 0.048
- Gain (S, Temperature) = 0.029

Bước kế tiếp trong tiến trình tăng trưởng trên cây quyết định



Day	Temp	Humidity	Wind	PlayTennis
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

S_{sunny} = {D1, D2, D3, D9, D11}

Gain(S_{sunny}, Humidity) = .970 - (3/5)0.0 - (2/5)0.0 = 0.970

Gain(S_{sunny}, Temperature) = .970 - (2/5)0.0 - (2/5)1.0 - (1/5)0.0 = 0.570

Gain(S_{sunny}, Wind) = .970 - (2/5)1.0 - (3/5)0.918 = 0.019

Điều kiện dừng

1. Từng thuộc tính đã được đưa vào dọc theo con đường trên cây.
2. Các mẫu huấn luyện ứng với nút lá có cùng giá trị thuộc tính đích (chẳng hạn, chúng có entropy bằng 0).

Thuật toán ID3 dùng Information Gain và C4.5, thuật toán được phát triển sau nó, dùng Gain Ratio (một biến thể của Information Gain). Trong quá trình xây dựng cây quyết định theo thuật toán ID3, tại mỗi bước triển khai cây thuộc tính được chọn để triển khai là thuộc tính có giá trị Gain lớn nhất. Thuật toán ID3 chưa giải quyết được vấn đề thuộc tính liên tục, số lượng các thuộc tính còn bị hạn chế và chưa xử lý tốt vấn đề dữ liệu bị thiếu hoặc bị nhiễu.

5.3.2 Thuật toán C4.5

Thuật toán C4.5 do Quilan phát triển vào năm 1996. Đây là một sự cải tiến từ thuật toán ID3 với việc cho phép xử lý trên tập dữ liệu có các thuộc tính số và làm việc được với tập dữ liệu bị thiếu và nhiễu.C4.5 thực hiện phân lớp tập mẫu dữ liệu theo chiến lược phát triển theo chiều sâu: xét tất cả các phép thử có thể để phân chia tập dữ liệu đã cho và chọn ra phép thử có giá trị Gain Raio tốt nhất.

Độ đo Gain có xu hướng thiên vị cho các thuộc tính có nhiều giá trị, ta cần chuẩn hóa độ đo Gain. GainRatio là đại lượng để đánh giá độ hiệu quả của thuộc tính dùng để phát triển cây. Gain Ration được tính theo công thức: GainRatio(A) = Gain(A)/SplitInfoA(D)

$$SpitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} * log_2 \frac{|D_j|}{|D|}$$

Chọn thuộc tính có độ đo Gain Ratio lớn nhất ***Gini Index***

- Tập huấn luyện D chứa các mẫu của m lớp, chỉ mục Gini của tập D là:

$$Gini(D) = 1 - \sum_{i=1}^{m} p^2{}_i$$

Với p_i là sác xuất của lớp C_i trong D

- Tập D phân chia theo thuộc tính A o Thuộc tính A có các giá trị $\{a_1, \ldots, a_v\}$. Dùng thuộc tính A để phân chia tập huấn luyện D thành v tập con $\{D_1, \ldots, D_v\}$ ☐ Chỉ mục Gini của tập D phân chia theo thuộc tính A:

$$gini_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} gini(D_j)$$

Tại mỗi cấp, chúng ta chọn thuộc tính có chỉ mục Gini nhỏ nhất để phân chia tập dữ liệu.

Thuật toán xây dựng cây quyết định mô tả như sau:

Procedure Tao_cay(T)

1. Tính tần xuất các giá trị trong các lớp của T

2. Nếu tất cả các mẫu thuộc cùng một lớp hoặc có rất ít mẫu khác lớp thì Trả về 1 nút lá. Ngược lại, tạo một nút quyết định N

3. Với mỗi thuộc tính A, tính giá trị Gain(A)

4. Tại nút N, thực hiện việc kiểm tra để chọn ra thuộc tính có giá trị Gain tốt nhất. Gọi N.test là thuộc tính có Gain lớn nhất

5. Với mỗi tập con $T' \subseteq T$, kiểm tra

Nếu T' rỗng thì gán nút con này của nút N là nút lá. Ngược lại, gọi hàm Tao_cay(T')

6. Tính toán các lỗi của nút N

Return nút N

Giải thuật C4.5 đã giải quyết được vấn đề làm việc với thuộc tính liên tục, thuộc tính có nhiều giá trị và vấn đề dữ liệu bị thiếu hoặc bị nhiễu. C4.5 thực hiện phân ngưỡng thuộc tính liên tục bằng phép tách nhị phân và dựa vào đại lượng GainRatio để giải quyết được vấn đề thuộc tính có nhiều giá trị. Ngoài ra, trong C4.5 còn có giải thuật cắt tỉa nhánh không phù hợp. Yếu điểm của thuật toán này là làm việc không hiệu quả với CSDL lớn.

5.3.3 Thuật giải Học Quy nạp ILA (ILA - Inductive Learning Algorithm)

Dùng để rút các luật phân lớp từ tập mẫu dữ liệu:

1. Chia bảng con có chứa m mẫu thành n bảng con. Một bảng con ứng với một giá trị của thuộc tính phân lớp. (Lặp lại từ bước 2 đến bước 8 cho mỗi bảng con).
2. Khởi tạo số lượng thuộc tính kết hợp j với j = 1.
3. Với mỗi bảng con đang xét, phân chia các thuộc tính của nó thành một danh sách các thuộc tính kết hợp, mỗi thành phần của danh sách có j thuộc tính phân biệt.
4. Với mỗi kết hợp các thuộc tính trong danh sách trên, đếm số lần xuất hiện các giá trị cho các thuộc tính trong kết hợp đó ở các dòng chưa bị khóa của bảng đang xét nhưng nó không được xuất hiện cùng giá trị ở những bảng con khác. Chọn ra một

kết hợp trong danh sách sao cho nó có giá trị tương ứng xuất hiện nhiều nhất và được gọi là Max_combination.

5. Nếu Max_combination =0 thì j = j+1 quay lại bước 3.
6. Khóa các dòng ở bảng con đang xét mà tại đó giá trị bằng với giá trị tạo ra max_combination.
7. Thêm vào R luật mới với giả thuyết là các giá trị tạo ra Max_combination kết nối các bộ này bằng phép AND, kết luận là giá trị của thuộc tính quyết định trong bảng con đang xét.
8. Nếu tất cả các dòng đều khóa:
   - Nếu còn bảng con thì qua bảng con tiếp theo và quay lại bước 2.
   - Ngược lại chấm dứt thuật toán.

   Ngược lại thì quay lại bước 4.

Ví dụ 5.1: Cho bảng quan sát sau

STT	Kích cỡ	Màu sắc	Hình dáng	Quyết định
1	Vừa	Xanh dương	Hộp	Mua
2	Nhỏ	Đỏ	Nón	Không mua
3	Nhỏ	Đỏ	Cầu	Mua
4	Lớn	Đỏ	Nón	Không mua
5	Lớn	Xanh lá	Trụ	Mua
6	Lớn	Đỏ	Trụ	Không mua
7	Lớn	Xanh lá	Cầu	Mua

Xây dựng luật để xác định thuộc tính thuộc châu cho các mẫu bằng phương pháp ILA, ta chia bảng CSDL thành 2 bảng con:

Bảng con 1: Nhóm Quyết định Mua

STT	Kích cỡ	Màu sắc	Hình dáng	Quyết định
1	Vừa	Xanh dương	Hộp	Mua
3	Nhỏ	Đỏ	Cầu	Mua
5	Lớn	Xanh lá	Trụ	Mua
7	Lớn	Xanh lá	Cầu	Mua

Bảng con 2: Nhóm Quyết định Không mua

STT	Kích cỡ	Màu sắc	Hình dáng	Quyết định
2	Nhỏ	Đỏ	Nón	Không mua
4	Lớn	Đỏ	Nón	Không mua
6	Lớn	Đỏ	Trụ	Không mua

Xét bảng con 1: nhóm Mua

STT	Kích cỡ	Màu sắc	Hình dáng	Quyết định
1	Vừa	Xanh dương	Hộp	Mua
3	Nhỏ	Đỏ	Cầu	Mua
[illegible]	Lớn	Xanh lá	Trụ	Mua
[illegible]	Lớn	Xanh lá	Cầu	Mua

STT	Kích cỡ	Màu sắc	Hình dáng	Quyết định
2	Nhỏ	Đỏ	Nón	Không mua
4	Lớn	Đỏ	Nón	Không mua
6	Lớn	Đỏ	Trụ	Không mua

Bước 2: j = 1

Bước 3: các kết hợp có 1 thuộc tính là [kích cỡ], [màu sắc], [hình dáng]

Bước 4: Đếm các giá trị

[kích cỡ] = {{vừa}:1}

[màu sắc] = {{xanh dương}:1, {xanh lá}:2}

[hình dáng] = {{hộp}:1, {cầu}:2}

Vậy: max_com = 2 tương ứng với màu sắc = xanh lá

Bước 6: Khóa các dòng có STT là 5,7

Bước 7: Tạo ra luật R1: IF màu sắc = xanh lá THEN Quyết định = Mua

Bước 8: Quay lại bước 4

STT	Kích cỡ	Màu sắc	Hình dáng	Quyết định
[illegible]	Vừa	Xanh dương	Hộp	Mua
3	Nhỏ	Đỏ	Cầu	Mua
[illegible]	Lớn	Xanh lá	Trụ	Mua
[illegible]	Lớn	Xanh lá	Cầu	Mua

STT	Kích cỡ	Màu sắc	Hình dáng	Quyết định
2	Nhỏ	Đỏ	Nón	Không mua
4	Lớn	Đỏ	Nón	Không mua
6	Lớn	Đỏ	Trụ	Không mua

Bước 4: Đếm các giá trị

[kích cỡ] = {{vừa}:1}

[màu sắc] = {{xanh dương}:1}

[hình dáng] = {{hộp}:1, {cầu}:1}

Vậy: max_com = 1 tương ứng với kích cỡ = vừa

Bước 6: Khóa dòng có STT là 1

Bước 7: Tạo ra luật R2: IF kích cỡ = vừa THEN Quyết định = Mua

Bước 8: Quay lại bước 4

STT	Kích cỡ	Màu sắc	Hình dáng	Quyết định
✗	Vừa	Xanh dương	Hộp	Mua
✗	Nhỏ	Đỏ	Cầu	Mua
✗	Lớn	Xanh lá	Trụ	Mua
✗	Lớn	Xanh lá	Cầu	Mua

STT	Kích cỡ	Màu sắc	Hình dáng	Quyết định
2	Nhỏ	Đỏ	Nón	Không mua
4	Lớn	Đỏ	Nón	Không mua
6	Lớn	Đỏ	Trụ	Không mua

Bước 4: Đếm các giá trị

[kích cỡ] = {}

[màu sắc] = {}

[hình dáng] = {{cầu}:1}

Vậy: max_com = 1 tương ứng với Hình dáng= cầu

Bước 6: Khóa dòng có STT là 3

Bước 7: Tạo ra luật R3: IF hình dáng = cầu THEN Quyết định = Mua

Bước 8: tất cả các dòng đều bị khóa, xét bảng con kế

Xét bảng con 2: nhóm Không mua

STT	Kích cỡ	Màu sắc	Hình dáng	Quyết định
1	Vừa	Xanh dương	Hộp	Mua
3	Nhỏ	Đỏ	Cầu	Mua
5	Lớn	Xanh lá	Trụ	Mua
7	Lớn	Xanh lá	Cầu	Mua

STT	Kích cỡ	Màu sắc	Hình dáng	Quyết định
✗	Nhỏ	Đỏ	Nón	Không mua
✗	Lớn	Đỏ	Nón	Không mua
6	Lớn	Đỏ	Trụ	Không mua

Bước 2: j = 1

Bước 3: các kết hợp có 1 thuộc tính là [kích cỡ], [màu sắc], [hình dáng]

Bước 4: Đếm các giá trị

[kích cỡ] = {}

[màu sắc] = {}

[hình dáng] = {{nón}:2}

Vậy: max_com = 2 tương ứng với hình dáng = nón

Bước 6: Khóa các dòng có STT là 2,4

Bước 7: Tạo ra luật R4: IF hình dáng = nón THEN QĐ = Không mua

Bước 8: Quay lại bước 4

STT	Kích cỡ	Màu sắc	Hình dáng	Quyết định
1	Vừa	Xanh dương	Hộp	Mua
3	Nhỏ	Đỏ	Cầu	Mua
5	Lớn	Xanh lá	Trụ	Mua
7	Lớn	Xanh lá	Cầu	Mua

STT	Kích cỡ	Màu sắc	Hình dáng	Quyết định
2	Nhỏ	Đỏ	Nón	Không mua
4	Lớn	Đỏ	Nón	Không mua
6	Lớn	Đỏ	Trụ	Không mua

Bước 4: Đếm các giá trị [kích cỡ] = {}; [màu sắc] = {}; [hình dáng] = {}

Vậy: max_com = 0

Bước 5: Do max_com = 0 nên j = 2 và quay lại bước 3

Bước 3: Các kết hợp có 2 thuộc tính là [kích cỡ, màu sắc], [kích cỡ, hình dáng], [màu sắc, hình dáng]

Bước 4: Đếm số lần xuất hiện của các giá trị

[kích cỡ, màu sắc] = {{lớn, đỏ}:1}

[kích cỡ, hình dáng] = {}

[màu sắc, hình dáng] = {{đỏ, trụ}: 1}

max_com = 1 ứng với kích cỡ = lớn và màu sắc = đỏ

STT	Kích cỡ	Màu sắc	Hình dáng	Quyết định
1	Vừa	Xanh dương	Hộp	Mua
3	Nhỏ	Đỏ	Cầu	Mua
5	Lớn	Xanh lá	Trụ	Mua
7	Lớn	Xanh lá	Cầu	Mua

STT	Kích cỡ	Màu sắc	Hình dáng	Quyết định
2	Nhỏ	Đỏ	Nón	Không mua
4	Lớn	Đỏ	Nón	Không mua
6	Lớn	Đỏ	Trụ	Không mua

Bước 4: Đếm số lần xuất hiện của các giá trị

[kích cỡ, màu sắc] = {{lớn, đỏ}:1}

[kích cỡ, hình dáng] = {}

[màu sắc, hình dáng] = {{đỏ, trụ}:1}

max_com = 1 ứng với kích cỡ = lớn và màu sắc = đỏ

Bước 6: Khóa dòng có STT = 6

Bước 7: Tạo ra luật R5: IF Kích cỡ = lớn AND Màu sắc = đỏ THEN Quyết định = Không mua

Bước 8: Tất cả các dòng đều khóa, ngừng thuật toán

5.3.4 Bộ phân lớp Bayes

Bộ phân lớp được xây dựng dựa trên phương pháp phân tích xác suất có điều kiện theo định lý Bayes gọi là bộ phân lớp Bayes. Bộ phân lớp này thực hiện việc dự đoán xác suất một đối tượng sẽ thuộc về lớp cụ thể nào đó.

Bộ phân lớp cơ bản nhất gọi là bộ phân lớp Bayes ngây thơ (naive Bayesian classifier). Bộ phân lớp Bayes ngây thơ được xây dựng dựa trên giả thiết rằng sự tác động của các thuộc tính lên việc xác định giá trị nhãn của một lớp cho trước là độc lập lẫn nhau gọi là độc lập theo điều kiện lớp. Nghĩa là các thuộc tính đầu vào là độc lập (không tương quan hay phụ thuộc) với nhau trong việc liên hệ với đầu ra của đối tượng.

Giả thiết này được đưa ra nhằm làm đơn giản hóa các ràng buộc của bài toán, từ đó giảm thiểu độ phức tạp, giúp việc giải thích nguyên tắc hoạt động của bộ phân lớp được rõ ràng hơn.

Định lý Bayes: được đưa ra bởi nhà toán học Thomas Bayes trong việc tìm xác suất xuất hiện của một đối tượng dựa trên sự xuất hiện (điều kiện) của các đối tượng khác và được mô tả như sau:

Gọi $X(A_1, ..., A_n) \in D$ là một đối tượng dữ liệu gồm n thuộc tính. H là một giả thuyết nêu lên rằng "X thuộc về một lớp C nào đó". Với bài toán phân lớp, cần xác định tính đúng đắn (được xác định bằng xác suất) của giả thuyết này. Hay nói cách khác, khi cho một đối tượng dữ liệu X (với các giá trị cụ thể của n thuộc tính), cần xác định xác suất P(H|X) là xác suất có điều kiện của H khi cho trước X

Theo định lý Bayes thì P(H|X) được xác định như sau:

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

Trong đó

- P(H|X): là xác suất xảy ra H khi X đã xảy ra, gọi xác suất hậu nghiệm (posterior probability)[2].
- P(H), P(X): là xác suất tiêu nghiệm (priori probability)[3].

Do X được xác định bởi các giá trị của các thuộc tính và được biết trước như là dữ liệu đầu vào để bộ phân lớp xác định tên lớp phù hợp cho nó nên X được gọi là đối tượng dự đoán và P(X) là xác suất tiêu nghiệm của đối tượng dự đoán.

Ví dụ 5.2: Cho bảng quan sát sau

ID	Age	Income	Student	Credit rating	Buys computer
1	Young	High	No	Fair	No
2	Young	High	No	Excellent	No
3	Middle	High	No	Fair	Yes
4	Old	Medium	No	Fair	Yes
5	Old	Low	Yes	Fair	Yes
6	Old	Low	Yes	Excellent	No
7	Middle	Low	Yes	Excellent	Yes
8	Young	Medium	No	Fair	No
9	Young	Low	Yes	Fair	Yes
10	Old	Medium	Yes	Fair	Yes
11	Young	Medium	Yes	Excellent	Yes
12	Middle	Medium	No	Excellent	Yes
13	Middle	High	Yes	Fair	Yes
14	Old	Medium	No	Excellent	No

Từ bảng quan sát trên, tính xác suất về khả năng "Buys computer" là Yes và No của khách hàng X có Age = Young và Income = High

[2]Xác suất hậu nghiệm theo thống kê của Bayes là xác suất được điều chỉnh hoặc cập nhật của một biến cố xảy ra sau khi xem xét thông tin mới.

[3] Xác suất tiêu nghiệm là một phương pháp dự đoán khả năng một biến cố sẽ xuất hiện theo một cách nhất định dựa trên tỉ lệ phần trăm, chứ không phải được xác định từ dữ liệu lịch sử

- P("Buys computer" = Yes|Age = Young, Income = High): *xác suất mua máy tính của khách hàng* có tuổi "trẻ" và thu nhập "cao"
- P(Age = Young, Income = High|"Buys computer" = Yes): *xác suất khách hàng đã mua máy tính* có tuổi "trẻ" và thu nhập "cao"
  - P(Age = Young, Income = High|"Buys computer" = Yes) = 0
  - P(Age = Young, Income = High|"Buys computer" = No) = 2/5
- P("Buys computer" = Yes): xác suất mua máy tính của khách hàng nói chung
  - P("Buys computer" = Yes) = 9/14
  - P("Buys computer" = No) = 5/14
- P(Age = Young, Income = High): *xác suất khách hàng* có tuổi "trẻ" và thu nhập "cao" → P(Age = Young, Income = High) = 2/14

Khi đó theo định lý Bayes, ta có:

$$P(\text{"Buys computer"} = \text{Yes}|\text{Age} = \text{Young}, \text{Income} = \text{High})$$
$$= \frac{P(\text{Age} = \text{Young}, \text{Income} = \text{High}|\text{"Buys computer"} = \text{Yes}) \times P(\text{"Buys computer"} = \text{Yes})}{P(\text{Age} = \text{Young}, \text{Income} = \text{High})}$$
$$= \frac{0 \times \frac{9}{14}}{\frac{2}{14}} = 0$$

$$P(\text{"Buys computer"} = \text{No}|\text{Age} = \text{Young}, \text{Income} = \text{High})$$
$$= \frac{P(\text{Age} = \text{Young}, \text{Income} = \text{High}|\text{"Buys computer"} = \text{No}) \times P(\text{"Buys computer"} = \text{No})}{P(\text{Age} = \text{Young}, \text{Income} = \text{High})}$$
$$= \frac{\frac{2}{5} \times \frac{5}{14}}{\frac{2}{14}} = 1$$

Như vậy có thể nói rằng xác suất của khách hàng X có Age = Young và Income = High với "Buys computer" = Yes là 0% và |"Buys computer" = No là 100%. Do đó nhãn phù hợp cho khách hàng X này là No.

Bộ phân lớp Bayes ngây thơ thực hiện như sau:

- Cho tập huấn luyện D với nhãn của các lớp C_i, $\forall i = \overline{1, m}$ (m lớp)

- Mỗi đối tượng trong tập D biểu diễn bởi vector n chiều, $X = (x_1, x_2, ..., x_n)$ với x_i là giá trị của thuộc tính A_i trong n thuộc tính $A_1, A_2, ..., A_n$
- X được phân vào lớp C_i có xác suất hậu nghiệm trên điều kiện X là lớn nhất

$$\text{argmax } P(C_j|X) = \frac{P(X|C_j)P(C_j)}{P(X)} \; \forall i = \overline{1,m} \quad \text{với } P(C_j) = \frac{|C_j, D|}{|D|}$$

Vì P(X) không đổi $\rightarrow \text{argmax } P(C_j|X) = \text{argmax } P(X|C_j)P(C_j)$

- Việc tính $P(X|C_j)$ với X là một vector n chiều → độ phức tạp lớn

Để giảm độ phức tạp, phương pháp phân lớp Bayes ngây thơ giả định rằng các giá trị của thuộc tính trong cùng một đối tượng dữ liệu tương ứng với một giá trị về lớp cho trước là độc lập lẫn nhau. Khi đó

$$P(X|C_j) = \prod_{k=1}^{n} P(x_k|C_j) \text{ với } x_k \text{ là giá trị của thuộc tính } A_k \text{ trong } X$$

Như vậy $P(x_k|C_j)$ dễ dàng tính được từ tập huấn luyện D. Với mỗi thuộc tính A_k, ta sẽ tính được $P(x_k|C_j)$ theo các cách thức khác nhau tương ứng với thuộc tính A_k có kiểu dữ liệu là rời rạc (categorical – valued) hoặc liên tục (continuous – valued).

- Nếu thuộc tính A_k là rời rạc thì $P(x_k|C_j)$ được tính như sau:

$$P(x_k|C_j) = \frac{|\{X'|x'_k = x_k \wedge X' = C_j\}|}{|C_{j,D}|}$$

- Số lượng các đối tượng dữ liệu thuộc lớp C_j trong D mà có giá trị của thuộc tính A_k là x_k, chia cho số lượng các đối tượng thuộc lớp C_j trong D

Các bước thực hiện như sau:

(1)Huấn luyện Bayes ngây thơ (trên tập huấn luyện)

- Lượng giá $P(C_j)$
- Lượng giá $P(x_k|C_j)$

(2)X được gán vào lớp có giá trị lớn nhất

$$\text{argmax } P(C_j) \prod_{k=1}^{n} P(x_k|C_j)$$

Ví dụ 5.3: lấy lại ví dụ 5.2

- m = 2 lớp → C_1 = Yes, C_2 = No
- Cần phân lớp đối tượng X (Age = Young, Income = Medium, Student = Yes, Credit rating = Fair)

  X thuộc về C_1 hoặc C_2 → $\text{argmax } P(C_i|X) = \text{argmax } P(X|C_i)P(C_i)$, i = 1,2
- Tính $P(C_1)$, $P(C_2)$

  P("Buys computer" = Yes) = 9/14

  P("Buys computer" = No) = 5/14
- Tính $P(X|C_i)$: n = 4 thuộc tính

$$P(X|C_i) = \prod_{k=1}^{4} P(x_k|C_i) = P(x_1|C_i)P(x_2|C_i)P(x_3|C_i)P(x_4|C_i) \text{ với } i = 1,2$$

- Với lớp C_1

  P(Age = Young|"Buys computer" = Yes) = 2/9

  P(Income = Medium|"Buys computer" = Yes) = 4/9

  P(Student = Yes|"Buys computer" = Yes) = 6/9

  P(Credit rating = Fair|"Buys computer" = Yes) = 6/9
- Với lớp C_2

  P(Age = Young|"Buys computer " = No) = 3/5

  P(Income = Medium|"Buys computer " = No) = 2/5

  P(Student = Yes|"Buys computer " = No) = 1/5

  P(Credit rating = Fair|"Buys computer" = No) = 2/5

  → $P(X|C_1)$ = P(X|"Buys computer" = Yes) = 2/9 × 4/9 × 6/9 × 6/9

  và $P(X|C_2)$ = P(X|"Buys computer" = No) = 3/5 × 2/5 × 1/5 × 2/5

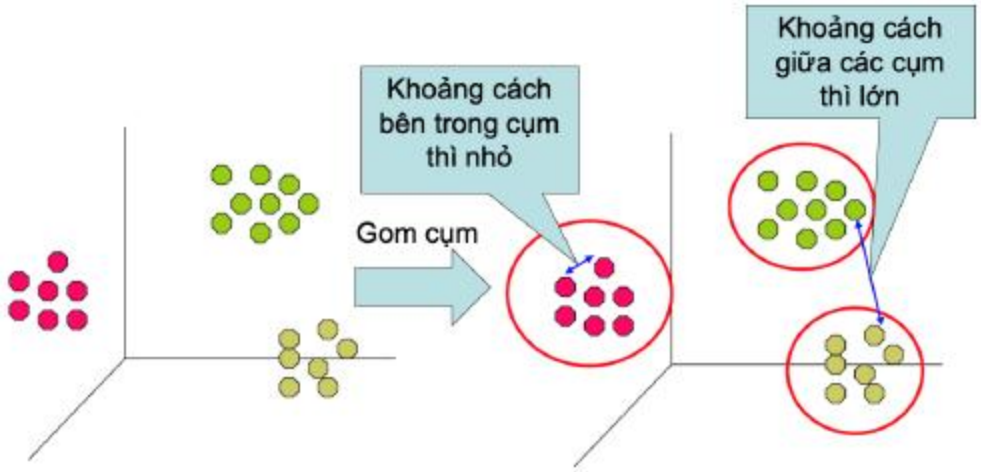
  → $P(X|C_1)P(C_1)$ = 2/9 × 4/9 × 6/9 × 6/9 × 9/14 ≈ 0.028

và $P(X|C_2)P(C_2)$ = 3/5 × 2/5 × 1/5 × 2/5 × 5/14 ≈ 0.007

Như vậy $P(X|C_1)P(C_1) > P(X|C_2)P(C_2)$ nên X được xếp vào lớp C_1

5.4 GOM CỤM DỮ LIỆU

Gom cụm dữ liệu (data clustering) là hình thức học không giám sát trong đó các mẫu học chưa được gán nhãn. Mục đích của gom cụm dữ liệu là tìm những mẫu đại diện hoặc gom dữ liệu tương tự nhau (theo một chuẩn đánh giá nào đó) thành những cụm. Các mẫu dữ liệu trong các cụm khác nhau có độ tương tự thấp hơn các mẫu nằm trong cùng một cụm.



Hình 5.12: Gom cụm dữ liệu

Một số ứng dụng tiêu biểu gom cụm như:

- Nhóm các tài liệu liên quan để dễ tìm kiếm hoặc cùng kiểu truy cập.
- Nhóm các gien và protein có cùng chức năng.
- Nhóm các cổ phiếu có cùng biến động.
- Nhóm các vùng theo lượng mưa.
- Nhóm các khu vực có loại đất giống nhau trong địa lý.
- Xác định nhóm nhà theo loại nhà, giá trị và vị trí địa lý
- Xác định nhóm đối tượng chơi game.
- Nhận dạng mẫu (pattern recognition)
- Phân tích dữ liệu không gian (spatial data analysis)
- Phân mảnh thị trường (market segmentation)

- Xem xét phân bố dữ liệu
- Tiền xử lý cho các thuật toán khác
- Khám phá thói quen và nhu cầu của khách hàng để có phương pháp tiếp thị thích hợp
- ...

Một phương pháp gom cụm tốt nếu đạt được các tính chất sau:

- Có độ tương tự cao trong cùng cụm (intra – class)
- Có độ tương tự thấp giữa các cụm (inter – class)
- Có khả năng phát hiện các mẫu ẩn.
- Có khả năng làm việc hiệu quả với lượng dữ liệu lớn (scalability).
- Có khả năng làm việc với nhiều loại dữ liệu khác nhau: có khả năng khám phá ra các cụm có phân bố theo các dạng khác nhau.
- Có khả năng làm việc với nhiễu và mẫu cá biệt.
- Không bị ảnh hưởng bởi thứ tự nhập của dữ liệu.
- Làm việc tốt trên CSDL có số chiều cao.
- Chấp nhận các ràng buộc do người dùng chỉ định
- Có thể hiểu và sử dụng được kết quả gom cụm.

Dựa trên cách tiếp cận và thuật toán sử dụng, người ta phân các thuật toán gom cụm theo các phương pháp chính sau:

- Các phương pháp phân hoạch
- Các phương pháp phân cấp.
- Các phương pháp dựa trên mật độ.
- Các phương pháp dựa trên mô hình.
- Các phương pháp dựa trên lưới.

Có thể dùng ma trận dữ liệu để mô hình hóa bài toán gom cụm. Ma trận biểu diễn không gian dữ liệu gồm n đối tượng theo p thuộc tính. Ma trận này biểu diễn môi quan hệ đối tượng theo thuộc tính:

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

Để biểu diễn khoảng cách giữa hai đối tượng trong không gian dữ liệu gồm n đối tượng theo p thuộc tính, ta dùng ma trận phân biệt như sau

$$\begin{bmatrix} 0 & & & \\ d(2,1) & 0 & & \\ d(3,1) & d(3,2) & & \\ \vdots & \vdots & \cdots & \vdots \\ d(n,1) & d(n,2) & \cdots & 0 \end{bmatrix}$$ với d(i, j) là khoảng cách giữa đối tượng i và đối tượng j

Để đánh giá độ tương tự giữa các đối tượng cần có độ đo khoảng cách được định nghĩa trong không gian dữ liệu đang xét. Không có một độ đo nào có thể dùng chung cho mọi trường hợp. Tùy theo mục tiêu khảo sát và bản chất dữ liệu người dùng phải chọn độ đo khoảng cách phù hợp với ứng dụng đang triển khai.

Gọi K là không gian dữ liệu. x, y, z là các đối tượng tuỳ ý trong không gian K. Độ đo d là hàm số $d: K \times K \to \mathbb{R}$ thoả:

(I) $d(x, y) \geq 0$

(II) $d(x, y) = 0$

(III) $d(x, y) = d(y, x)$

(IV) $d(x, y) \leq d(x, z) + d(z, y)$

Giá trị của độ đo d(x,y) càng nhỏ thì x và y càng gần nhau (càng tương tự nhau).

Trong CSDL có thể có nhiều kiểu thuộc tính khác nhau. Một đối tượng dữ liệu được đặc trưng bằng nhiều thuộc tính có kiểu cơ sở. Để xây dựng được một độ đo tốt, có thể áp dụng cho dữ liệu tổng quát, ta cần phải xây dựng được độ đo tốt cho các kiểu cơ sở. Các kiểu cơ sở gồm trị khoảng (interval – valued), nhị phân đối xứng (symmetric binary), nhị phân bất đối xứng (asymmetric binary), định danh (nominal), thứ tự (ordinal), tỷ lệ khoảng (ratio – scaled). Trong khuôn khổ của tài liệu, ta chỉ quan tâm đến kiểu cơ sở là trị khoảng.

Các biến trị khoảng là độ đo liên tục của các đại lượng tuyến tính đơn giản như trọng lượng, chiều cao, nhiệt độ, tuổi, ... Các đơn vị đo khác nhau ảnh hưởng rất nhiều đến kết quả gom cụm. Do đó tùy vào lĩnh vực ứng dụng và tiêu chí của phương pháp tiếp cận mà chuẩn hoá dữ liệu để đưa chúng về cùng một đơn vị.

❖ Phương pháp chuẩn hoá dữ liệu các độ đo

- Tính độ lệch tuyệt đối trung bình (mean absolute deviation) s_f đối với thuộc tính f của n đối tượng trong CSDL

$$s_f = \frac{1}{n}\sum_{i-1}^{n}|x_{if} - m_f| \quad \text{với } m_f = \frac{1}{n}\sum_{i-1}^{n} x_{if} \quad (m_f \text{ là giá trị trung bình})$$

- Tính độ đo chuẩn hóa (z – score)

$$z_{if} = \frac{x_{if} - m_f}{s_f}$$

Sai số tuyệt đối trung bình càng lớn thì hiện tượng cá biệt càng giảm Do đó độ đo được chọn sẽ ảnh hưởng đến kết quả phân tích mẫu cá biệt.

❖ Các độ đo thông dụng

- Độ đo khoảng cách Minkowski

$$d(i,j) = \left(\sum_{k-1}^{n}|x_{ik} - x_{jk}|^q\right)^{\frac{1}{q}}$$

Ba trường hợp đặc biệt của khoảng cách Minkowski xảy ra khi q = 1, q = 2 và q = ∞

- q = 1 (Manhattan)

$$d(i,j) = \sum_{k-1}^{n}|x_{ik} - x_{jk}|$$

- q = 2 (Euclide)

$$d(i,j) = \sqrt{\sum_{k-1}^{n}|x_{ik} - x_{jk}|^2}$$

- q = ∞ (Chebyshev)

$$d(i,j) = \max_{k-1}|x_{ik} - x_{jk}|$$

- Mỗi thuộc tính có thể được gán trọng số, khi đó khoảng cách Euclide như sau

$$d(i,j) = \sqrt{\sum_{k-1}^{n} w_i |x_{ik} - x_{jk}|^2}$$

Khoảng cách có trọng số là sự cải tiến của khoảng cách Minkowski, trong đó có tính đến ảnh hưởng của từng thuộc tính đến khoảng cách giữa hai đối tượng. Thuộc tính có trọng số càng lớn thì ảnh hưởng càng nhiều đến khoảng cách. Việc chọn trọng số tuỳ thuộc vào ứng dụng và mục tiêu cụ thể.

Gom cụm bằng phân hoạch là phương pháp đơn giản và nền tảng nhất trong số các phương pháp gom cụm. Ý tưởng là phân hoạch một CSDL gồm n đối tượng thành tập k cụm (với k cho trước) sao cho:

(I) Mỗi cụm chứa ít nhất một đối tượng

(II) Mỗi đối tượng thuộc về một cụm duy nhất

Các phương pháp tiếp cận phân hoạch:

- Tối ưu toàn cục bằng vét cạn: có thể có $(k^n - (k-1)^n - \cdots - 1)$ khả năng phân hoạch khác nhau. Đây là con số quá lớn nếu n là khá lớn do đó hầu như không thể thực hiện được.
- Các phương pháp heuristic:
  - k – means (MacQueen' 67): mỗi cụm được đại diện bằng trọng tâm của cụm.
  - k – medoids (Kaufman & Rouseau '87) còn được gọi là PAM (partition around medoids): mỗi cụm được đại diện bởi một đối tượng của cụm.

5.4.1 Thuật toán k – means

Giả sử tập dữ liệu D có n đối tượng. Phương pháp phân hoạch phân phối các đối tượng trong D thành k cụm $C_1, \ldots, C_k$ với $C_i \subset D$, $C_i \cap C_j = \varnothing (1 \le i, j \le k)$. mỗi cụm C_i được biểu diễn bằng giá trị trọng tâm (centroid) của nó. Thuật toán gồm bốn bước như sau:

(1)Chọn ngẫu nhiên k đối tượng làm trọng tâm (centroid) ban đầu của k cụm.
(2)Gán (hoặc gán lại) từng đối tượng vào cụm có trọng tâm gần đối tượng đang xét

nhất.

Nếu không có phép gán lại nào thì dừng. Vì không có phép gán lại nào có nghĩa là các cụm đã ổn định và thuật toán không thể cải thiện làm giảm độ phân biệt hơn được nữa.

(3)Tính lại trọng tâm cho từng cụm.

(4)Quay lại bước (2)

Giá trị trọng tâm c_i của mỗi cụm C_i được tính như sau

$$c_i = \frac{1}{|C_i|}\sum_{p \in C_i} p$$

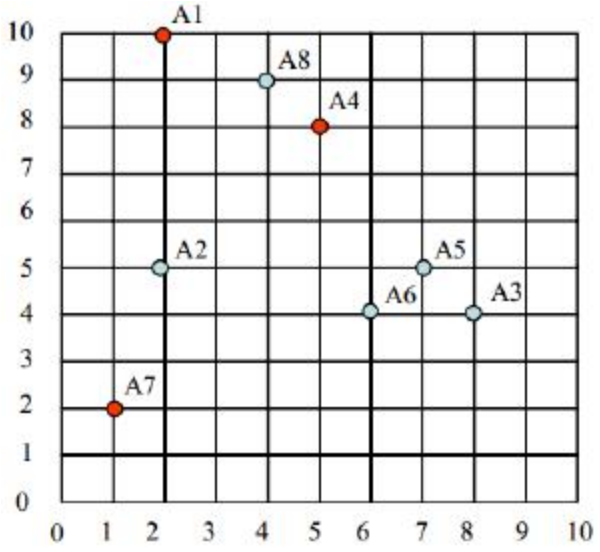
Chất lượng (còn gọi là chi phí – cost) của cụm C_i có thể được đo bằng độ biến đổi bên trong cụm là bằng tổng bình phương lỗi (sai số) khoảng cách của tất cả đối tượng trong cụm đến trọng tâm. Khi đó tổng của sai số bình phương cho tất cả các đối tượng trong tập dữ liệu như sau

$$E = \sum_{i-1}^{k}\sum_{p \in C_i} d^2(p, c_i) \quad \text{với d là hàm tính khoảng cách}$$

Ví dụ 5.4: Sử dụng thuật toán k – means và khoảng cách Euclide để gom cụm 8 đối tượng vào ba cụm: $A_1 = (2, 10)$, $A_2 = (2, 5)$, $A_3 = (8, 4)$, $A_4 = (5, 8)$, $A_5 = (7, 5)$, $A_6 = (6, 4)$, $A_7 = (1, 2)$, $A_8 = (4, 9)$. Giả sử khởi tạo trọng tâm là $k_1 = A_1$, $k_2 = A_4$ và $k_3 = A_7$. Chạy k – means một lần.

a. Xác định các cụm được hình thành

b. Trọng tâm mới của từng cụm



Hình 5.13: Các đối tượng của tập dữ liệu ban đầu và khởi tạo trọng tâm

Bảng khoảng cách Euclide giữa các đối tượng

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
A_1	0	$\sqrt{25}$	$\sqrt{72}$	$\sqrt{13}$	$\sqrt{50}$	$\sqrt{52}$	$\sqrt{65}$	$\sqrt{5}$
A_2		0	$\sqrt{37}$	$\sqrt{28}$	$\sqrt{25}$	$\sqrt{17}$	$\sqrt{10}$	$\sqrt{20}$
A_3			0	$\sqrt{25}$	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{53}$	$\sqrt{41}$
A_4				0	$\sqrt{13}$	$\sqrt{17}$	$\sqrt{52}$	$\sqrt{2}$
A_5					0	$\sqrt{2}$	$\sqrt{45}$	$\sqrt{25}$
A_6						0	$\sqrt{29}$	$\sqrt{29}$
A_7							0	$\sqrt{58}$
A_8								0

a. Xác định các cụm được hình thành

- Xét A_1

  $d(A_1, k_1) = 0$ vì k_1 là A_1

  $d(A_1, k_2) = \sqrt{13}$

  $d(A_1, k_3) = \sqrt{65}$

  $\rightarrow A_1 \in C_1$

- Xét A_2

  $d(A_2, k_1) = \sqrt{25}$

  $d(A_2, k_2) = \sqrt{18}$

  $d(A_2, k_3) = \sqrt{10}$

  $\rightarrow A_2 \in C_3$

- Xét A_3

  $d(A_3, k_1) = \sqrt{72}$

  $d(A_3, k_2) = \sqrt{25}$

  $d(A_3, k_3) = \sqrt{53}$

  $\rightarrow A_3 \in C_2$

- Xét A_4

  $d(A_4, k_1) = \sqrt{13}$

  $d(A_4, k_2) = 0$ vì k_2 là A_4

  $d(A_4, k_3) = \sqrt{52}$

  $\rightarrow A_4 \in C_2$

- Xét A_5

  $d(A_5, k_1) = \sqrt{50}$

  $d(A_5, k_2) = \sqrt{13}$

  $d(A_5, k_3) = \sqrt{45}$

  $\rightarrow A_5 \in C_2$

- Xét A_6

  $d(A_6, k_1) = \sqrt{52}$

  $d(A_6, k_2) = \sqrt{17}$

  $d(A_6, k_3) = \sqrt{29}$

  $\rightarrow A_6 \in C_2$

- Xét A_7

  $d(A_7, k_1) = \sqrt{65}$

  $d(A_7, k_2) = \sqrt{52}$

  $d(A_7, k_3) = 0$ vì k_3 là A_7

  $\rightarrow A_7 \in C_3$

- Xét A_8

  $d(A_8, k_1) = \sqrt{5}$

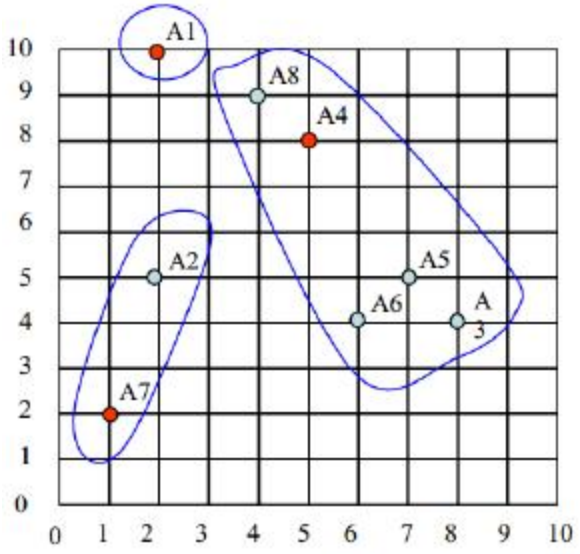
  $d(A_8, k_2) = \sqrt{2}$

  $d(A_7, k_3) = \sqrt{58}$

  $\rightarrow A_8 \in C_2$

Các cụm sau một lần chạy k – means:

- $C_1 = \{A_1\}$
- $C_2 = \{A_3, A_4, A_5, A_6, A_8\}$
- $C_3 = \{A_2, A_7\}$



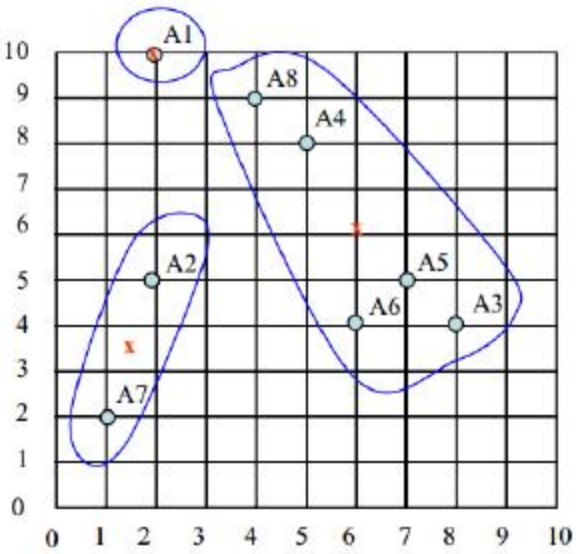
Hình 5.14: Các đối tượng được phân cụm sau lần chạy đầu tiên

b. Trọng tâm mới của từng cụm

$$c_1 = (2, 10)$$

$$c_2 = \left(\frac{8 + 5 + 7 + 6 + 4}{5}, \frac{4 + 8 + 5 + 4 + 9}{5}\right) = (6, 6)$$

$$c_3 = \left(\frac{2 + 1}{2}, \frac{5 + 2}{2}\right) = (1.5, 3.5)$$



Hình 5.15: Các trọng tâm mới của từng cụm

5.4.2 Thuật toán k – medoids

Thuật toán k – means nhạy cảm với nhiễu hay giá trị cá biệt (outlier). Chẳng hạn như một đối tượng có giá trị cực lớn có thể làm sai lệch đáng kể phân bố của dữ liệu.

Thuật toán k – medoids sẽ khắc phục vấn đề trên của thuật toán k – means. Thay vì lấy giá trị trọng tâm làm điểm đối chiếu, thuật toán lấy đối tượng nằm ở vị trí trọng tâm của cụm.

Các bước thực hiện như sau:

(1)Chọn ngẫu nhiên k điểm O_i (i = 1, … , k) làm trọng tâm (medoids) ban đầu của k cụm

(2)Gán (hoặc gán lại) từng đối tượng vào cụm có trọng tâm gần đối tượng đang xét nhất

(3)Với mỗi đối tượng trọng tâm O_i (i = 1, ..., k)

(I) Lần lượt xét các điểm không là trọng tâm (non – medoids) x

(II) Tính S là độ lợi khi hoán đổi O_i bởi x

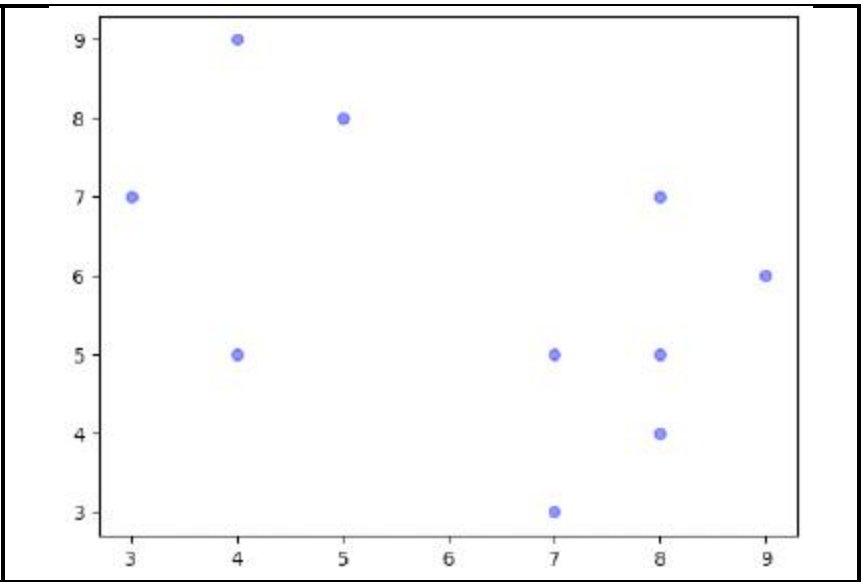
$$S = E_x - E_{O_i} \text{ với } E_o = \sum_{i-1}^{k} \sum_{p \in C_i} d^2(p, o) \quad (o \text{ là x hoặc } O_i)$$

(III) Nếu S < 0 thì thay thế O_i trong bộ k trọng tâm bởi x (chọn trọng tâm mới tốt hơn)

(4)Nếu có ít nhất một sự thay đổi trong bước (3) thì tiếp tục quay về (2). Ngược lại thì kết thúc thuật toán

Ví dụ 5.5: cho 10 đối tượng như sau với khoảng cách Manhattan và k = 2

O_i	x	y
0	8	7
1	3	7
2	4	9
3	9	6
4	8	5
5	5	8
6	7	3
7	8	4
8	7	5
9	4	5



- Chọn ngẫu nhiên hai đối tượng là hai trọng tâm O_4 = (8, 5) và O_9 = (4, 5). Tính các khoảng cách từ O_4 và O_9 đến các đối tượng khác.

O_i	x	y	Khoảng cách từ O_4	Khoảng cách từ O_9
0	8	7	2	6
1	3	7	7	3
2	4	9	8	4
3	9	6	2	6
4	8	5	-	-
5	5	8	6	4
6	7	3	3	5
7	8	4	1	5
8	7	5	1	3
9	4	5	-	-

→ C_1 = {O_0, O_3, O_4, O_6, O_7, O_8,} và C_2 = {O_1, O_2, O_5, O_9}

→ Tổng chi phí: (2 + 2 + 3 + 1 + 1) + (3 + 4 + 4) = 20

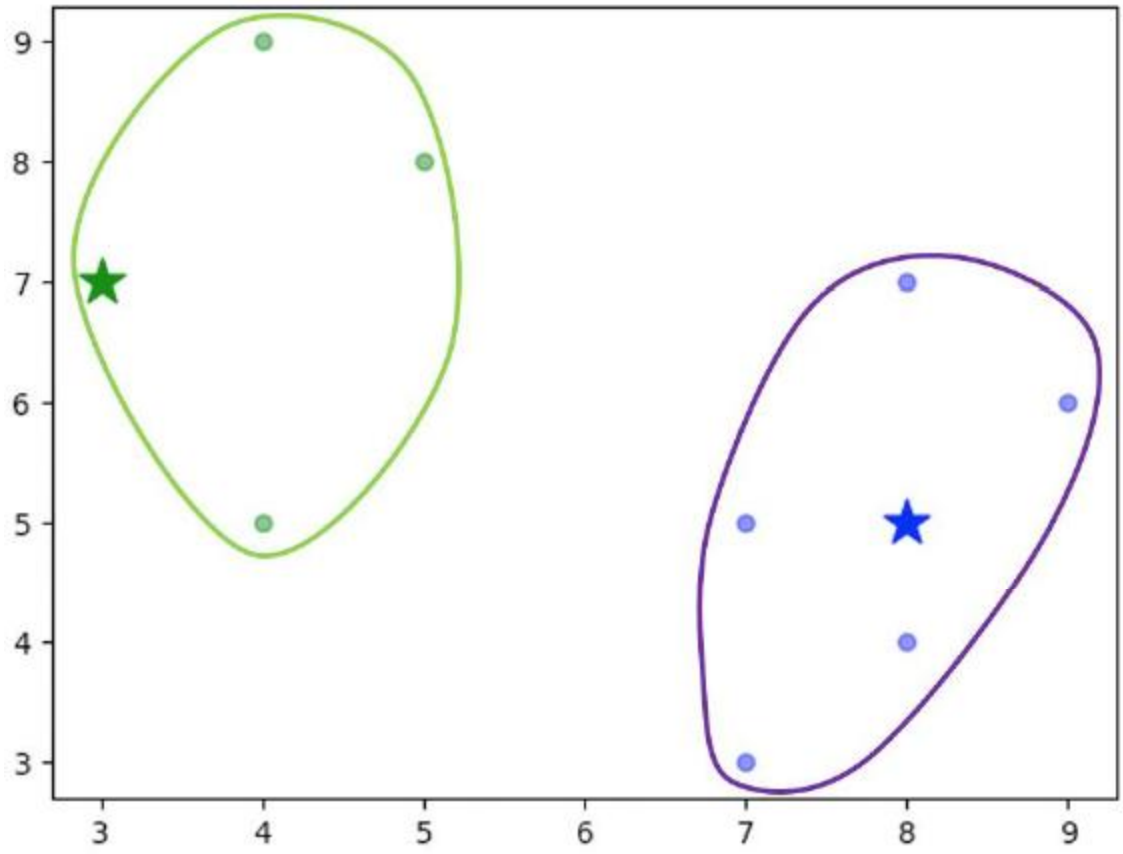
- Chọn O_7 = {8, 4} thay thế O_4. Tính các khoảng cách từ O_7 và O_9 đến các đối tượng khác

O_i	x	y	Khoảng cách từ O_7	Khoảng cách từ O_9
0	8	7	3	6
1	3	7	8	3
2	4	9	9	4
3	9	6	3	6
4	8	5	1	4
5	5	8	7	4
6	7	3	2	5
7	8	4	-	-
8	7	5	2	3
9	4	5	-	-

→ Tổng chi phí: (3 + 3 + 1 + 2 + 2) + (3 + 4 + 4) = 22

Do đó độ lợi S = 22 – 20 = 2 > 0 → Vẫn giữ nguyên hai đối tượng trọng tâm là O_4 và O_9

Tương tự ta tìm được hai đối tượng trọng tâm cuối cùng là O_1 và O_4



Hình 5.15: Kết quả thực hiện thuật toán k – **medoits.**

TÀI LIỆU THAM KHẢO

1. Bạch Hưng Khang, Hoàng Kiếm – Trí tuệ nhân tạo – Các phương pháp và ứng dụng - NXB Khoa học kỹ thuật, 1989
2. Đinh Mạnh Tường, Giáo trình Trí tuệ nhân tạo, Khoa CNTT – ĐH quốc gia Hà Nội
3. Võ Đình Bảy, Slide Bài giảng TTNT, ĐH Công nghệ Tp. HCM.
4. Stuart Russell, Peter Norvig (2010). Artificial Intelligence: A Modern Approach. Published by Prentice Hall (3rd Edition).
5. Miroslav Kubat (2017). An Introduction to Machine Learning by Springer International Publishing (2nd Edition)
6. Jiawei Han, Micheline Kamber, Jian Pei (2011). Data Mining: Concepts and Techniques by Published Morgan Kaufmann (3rd Edition)