

NHẬP MÔN TRÍ TUỆ NHÂN TẠO

Chương 1: Giới thiệu chung

Biên soạn: TS Ngô Hữu Phúc

Bộ môn: Khoa học máy tính

Mobile: 098 56 96 580

Email: ngohuuphuc76@gmail.com

Thông tin chung

- Thông tin về nhóm môn học:

TT	Họ tên giáo viên	Học hàm	Học vị	Đơn vị công tác (Bộ môn)
1	Ngô Hữu Phúc	GVC	TS	BM Khoa học máy tính
2	Trần Nguyên Ngọc	GVC	TS	BM Khoa học máy tính
3	Hà Chí Trung	GVC	TS	BM Khoa học máy tính
4	Trần Cao Trường	GV	ThS	BM Khoa học máy tính

- Thời gian, địa điểm làm việc: Bộ môn Khoa học máy tính Tầng 2, nhà A1.
- Địa chỉ liên hệ: Bộ môn Khoa học máy tính, khoa Công nghệ thông tin.
- Điện thoại, email: 069-515-329, ngohuuphuc76.mta@gmail.com.

Cấu trúc môn học

- Chương 1: Giới thiệu chung.
- Chương 2: Logic hình thức.
- Chương 3: Các phương pháp tìm kiếm mù.
- Chương 4: Các phương pháp tìm kiếm có sử dụng thông tin.
- Chương 5: Các chiến lược tìm kiếm có đối thủ.
- Chương 6: Các bài toán thỏa ràng buộc.
- Chương 7: Nhập môn học máy.

Bài 1: Giới thiệu chung (1/2)

Chương 1, mục: 1.1 – 1.9

Tiết: 1-3; Tuần thứ: 1.

Mục đích, yêu cầu:

1. Nắm được sơ lược về Học phần, các chính sách riêng của giáo viên, địa chỉ Giáo viên, bầu lớp trưởng Học phần.
2. Nắm được các khái niệm về Trí tuệ nhân tạo.
3. Nắm được các lĩnh vực có liên quan đến Trí tuệ nhân tạo.
4. Nắm được những vấn đề cốt lõi của Trí tuệ nhân tạo.

Hình thức tổ chức dạy học: Lý thuyết.

Thời gian: 3 tiết.

Địa điểm: Giảng đường do Phòng Đào tạo phân công

Nội dung chính: (Slides)

Bài 1: Giới thiệu chung (2/2)

1. Giới thiệu các thông tin liên quan đến khoá học.
2. Yêu cầu của khoá học.
3. Khái niệm về Trí tuệ nhân tạo.
4. Các lĩnh vực liên quan đến trí tuệ nhân tạo.
5. Lịch sử hình thành khoa học về trí tuệ nhân tạo.
6. Các lĩnh vực và ứng dụng của trí tuệ nhân tạo.
7. So sánh giữa lập trình hệ thống và lập trình AI.
8. Những vấn đề chưa được giải quyết.
9. Những vấn đề cốt lõi của trí tuệ nhân tạo.

Tài liệu tham khảo

Tài liệu môn học:

- Artificial Intelligence: A Modern Approach, S.J. Russell and P. Norvig, 2nd Edition, Prentice-Hall, 2003.
- Essentials of Artificial Intelligence , M.Ginsberg, Morgan Kaufmann, 1993.
- Trí tuệ nhân tạo: Các phương pháp giả quyết vấn đề và kỹ thuật xử lý tri thức, Nguyễn Thanh Thủy.
- Trí tuệ nhân tạo, Đỗ Trung Tuấn.

Một số website:

- <http://www.cs.adfa.edu.au/~z3013620/we/course.htm>
- <http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-034Spring-2005/CourseHome/index.htm>

1. Giới thiệu chung về khóa học

- I. Giới thiệu chung về TTNT.
- II. Logic hình thức.
- III. Các phương pháp tìm kiếm mù.
- IV. Các giải thuật tìm kiếm có kinh nghiệm.
- V. Kiểm tra giữa kỳ.
- VI. Các giải thuật tìm kiếm có đối thủ.
- VII. Các bài toán thỏa ràng buộc.
- VIII. Nhập môn máy học.
- IX. Một số ứng dụng trong thực tế.

2. Yêu cầu của khóa học

- Thực hiện đúng hướng dẫn của Học viện về đánh giá.
- Nắm chắc nội dung lý thuyết và áp dụng trong bài tập cụ thể.
- Học viên phải đi học đầy đủ.
- Học viên tham gia bài kiểm tra giữa kỳ.
- Bài thi hết môn gồm 02 phần:
 - Phần lý thuyết.
 - Phần bài tập (được giao vào tuần thứ 6 của môn học).

3. Khái niệm về Trí tuệ nhân tạo (1/)

- Hiện nay, trên thế giới có nhiều định nghĩa khác nhau về trí tuệ nhân tạo. Tuy nhiên, vẫn chưa thống nhất một dạng định nghĩa.
- Mặc dù vậy, có 2 trường phái về khái niệm AI:
 - **Strong AI:** *Có thể tạo ra thiết bị có trí thông minh và các chương trình máy tính thông minh hơn người!!!*
 - **Weak AI:** *Chương trình máy tính có thể mô phỏng các hành vi thông minh của con người!!!*

3. Khái niệm về Trí tuệ nhân tạo (2/)

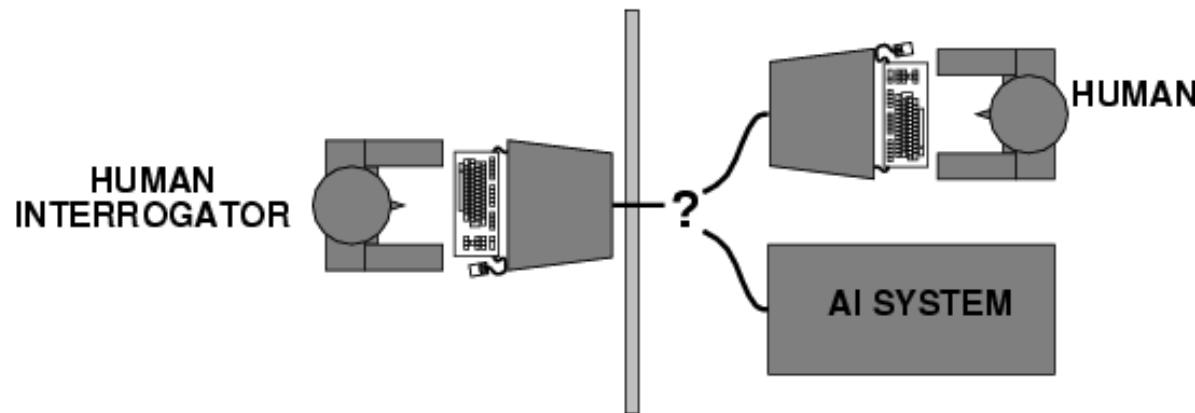
Có 4 quan điểm về AI:

Suy nghĩ như người	Suy nghĩ có lý trí
Hành động như người	Hành động có lý trí

Tài liệu tập trung vào nhóm quan điểm “hành động có lý trí”.

Hành động như người: Turing Test

- Turing (1950) "Computing machinery and intelligence":
- "Máy tính có thể nghĩ?" → "Máy tính có thể hành động thông minh?"
- Turing Test: Trò chơi bắt chước người.



- **Ưu điểm của Turing Test**
 - Khái niệm khách quan về trí tuệ
 - Tránh đi những thảo luận về quá trình bên trong và ý thức
 - Loại trừ định kiến thiên vị của người thẩm vấn

Các ý kiến phản đối Turing Test

- Thiêng vị các nhiệm vụ giải quyết vấn đề bằng ký hiệu.
- Trói buộc sự thông minh máy tính theo kiểu con người, trong khi con người có:
 - Bộ nhớ giới hạn
 - Có khuynh hướng nhầm lẫn

Tuy nhiên, trắc nghiệm Turing đã cung cấp một cơ sở cho nhiều sơ đồ đánh giá dùng thực sự cho các chương trình TTNT hiện đại.

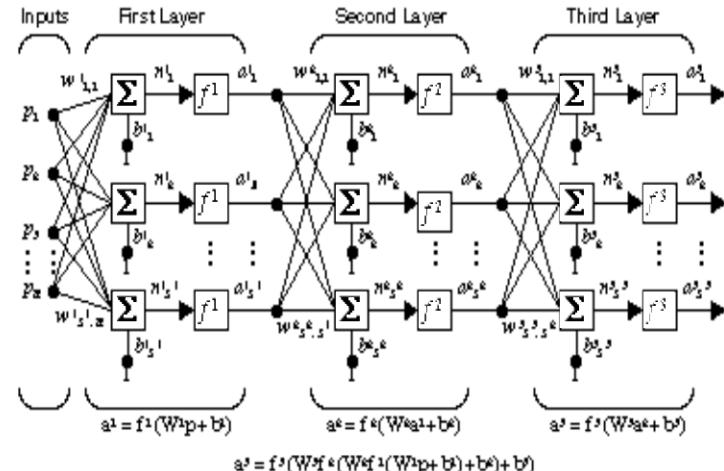
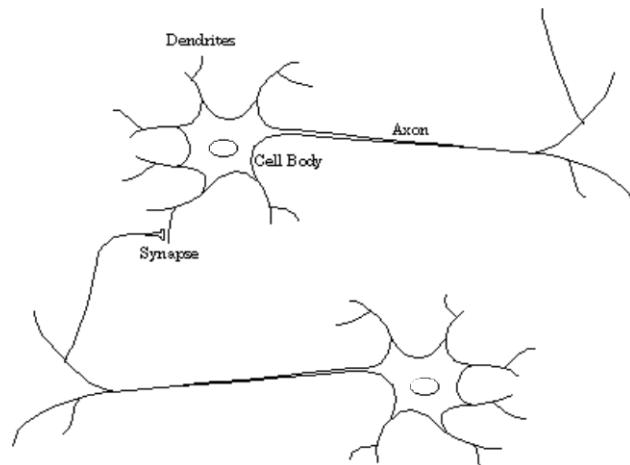
Suy nghĩ như người

Suy nghĩ như người:

- Cách tiếp cận cuối thế kỷ 19, đầu thế kỷ 20 về tâm lý học nhận thức. Chủ yếu quan tâm đến việc nghiên cứu xem trí tuệ của con người là gì? các chức năng thể hiện trí tuệ như: xử lý ngôn ngữ, nghĩ, hoc, lập luận được thực hiện như thế nào?

Hai cách tiếp cận:

- Trên xuống: Tâm lý học nt → Symbolism (Simon & Newell, 1961).
- Dưới lên: Neural and Brain Science (Mc Culloch, Pitt 1950s) → Artificial Neural Networks.



Suy nghĩ có lý trí

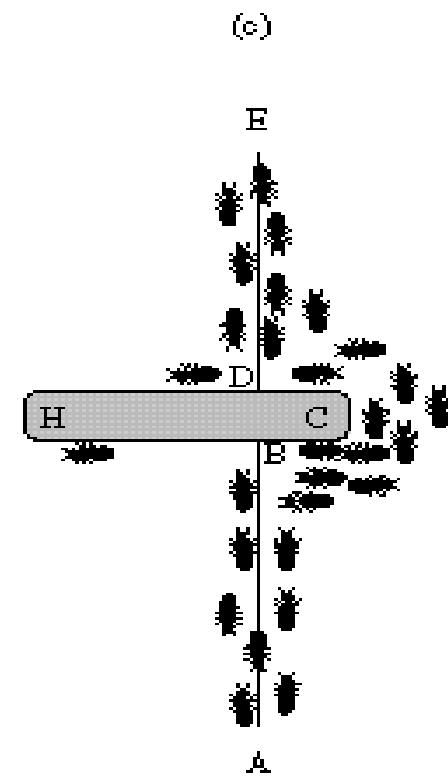
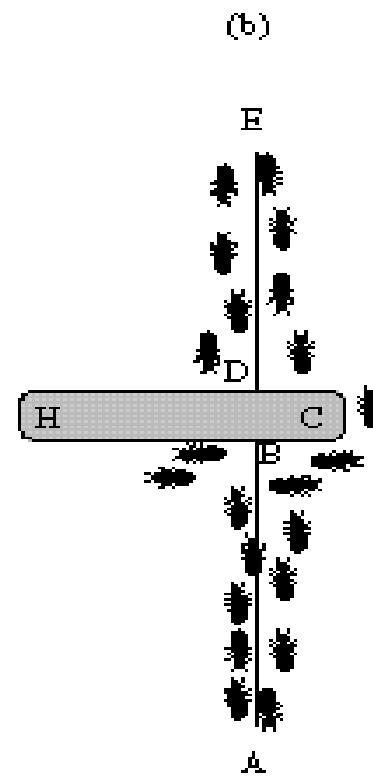
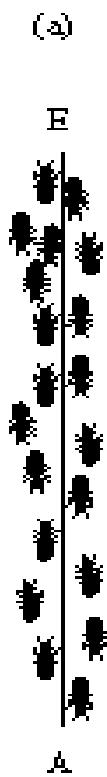
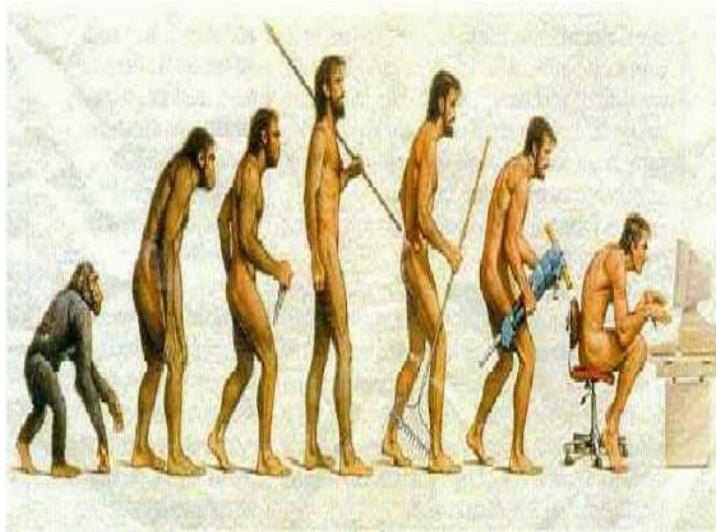
- **Bắt đầu từ thời Hylap cổ đại (Rule of Arguments) cho đến G. Boole (Mathematical Model of Thoughts), cho đến Hilbert: Logics. (nhưng không phải các hành vi thông minh đều có thể biểu diễn bằng Logic!)**
- **Các vấn đề:**
 1. Không phải các hành vi thông minh đều có thể biểu diễn bằng logic.
 2. Mục đích của suy nghĩ là gì?

Hành động có lý trí

- **Doing the right thing** (not “Doing the thing right”).
- Hành vi được coi là thông minh nếu giúp cho tác nhân (agent) thực hiện hành vi tăng cơ hội thực hiện được đích đặt ra cho nó với điều kiện thông tin phương tiện cho phép của môi trường mà nó đang tồn tại.
- Như vậy: Lợi điểm của định nghĩa:
 - Thông minh không nhất thiết phải là con người hay giống người!!!
 - Hành vi thông minh không nhất thiết phải thực hiện thông qua suy nghĩ, lý luận.

Ví dụ về TTNT

- Ví dụ: Sự tiến hóa (Evolutionary Intelligence), Tính bầy đàn (Swarm Intelligence).



Một số định nghĩa về TTNT trong tài liệu tham khảo

- Trí tuệ nhân tạo giúp tạo ra máy tính có khả năng suy nghĩ...máy tính có trí tuệ theo đầy đủ nghĩa của từ này (Haugeland, 1985).
- Trí tuệ nhân tạo là khoa học nghiên cứu xem làm thế nào để máy tính có thể thực hiện được những công việc mà hiện con người con làm tốt hơn máy tính (Rich and Knight, 1991).
- TTNT là khoa học nghiên cứu về các hoạt động trí não thông qua các mô hình tính toán (Chaniak và McDemott, 1985).
- Nghiên cứu các mô hình tính toán để máy tính có thể nhận thức, lập luận, và hành động (Winston, 1992).
- TTNT nghiên cứu các hành vi thông minh mô phỏng trong các vật thể nhân tạo (Nilsson 1998).

Định nghĩa

- 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!!!

4. Các lĩnh vực liên quan đến TTNT

- **Tâm lý học nhận thức.**
- **Thần kinh học.**
- **Lý thuyết về hệ thống (cybernetics).**
- **Toán Logic và Logic học.**
- **Sinh học tiến hóa.**
- **Khoa học về hành vi bầy đàn.**
- **Tổ chức học.**
- **Thống kê học.**
-

5. Lịch sử hình thành khoa học TTNT

Ba giai đoạn:

- Symbolism (70-80) (Automated Reasoning and Proofing, Expert Systems, Logic Programming,...).
- Connectionism (80s-90s) (Neural Networks, Statistical Learning, Support Vector Machines, Probabilistic Graph Learning,....).
- Evolutionary Computation (90s-?) (Evolutionary Programming, Evolutionary Strategies, Genetic Algorithms) , Intelligent Multi Agent Systems.

5. Lịch sử hình thành và phát triển (t)

- 1930-A.M.Turing đưa ra các kết quả nghiên cứu về máy thông minh, chương trình thông minh đến trắc nghiệm thông minh, đồng thời đưa ra các kết quả cơ sở quan trọng về máy Turing.
- Phát hiện quan trọng của Turing là chương trình có thể lưu trữ trong bộ nhớ để sau đó được thực hiện trên cơ sở các phép toán cơ bản thao tác với các đại lượng là số 0 và 1 của hệ đếm nhị phân.
- Việc lưu giữ chương trình trong máy cho phép thay đổi chức năng của nó một cách nhanh chóng và dễ dàng thông qua việc nạp chương trình mới khác vào bộ nhớ.
- Điều trên làm cho máy có khả năng học và suy nghĩ đáy chính là biểu hiện đầu tiên của các máy tính được trang bị TTNT.

5. Lịch sử hình thành và phát triển (t)

- 1956-Chương trình tìm dẫn xuất trong các hệ hình thức.
- 1959-Máy giải toán vạn năng (MP3).
- 1960-Mc Kathy đưa ra ngôn ngữ trí tuệ nhân tạo (Lisp-List Processing).
- 1961-Minsky đưa ra ngôn ngữ AI
- Tri thức + Điều khiển = chương trình.
- 1962- Tính tích phân bất định
- 1963- Chương trình Heuristic-(gợi mở).
- 1964-Giải phương trình đại số sơ cấp.

5. Lịch sử hình thành và phát triển (t)

- (chương trình ELIDA - phân tích tâm lý).
- 1966- Phân tích và tổng hợp tiếng nói.
- 1968-Robot.
- Học nói.
- 1972-A. Camerauls (ngôn ngữ Prolog-chương trình Logic)
- 1970-1980: Xử lý ngôn ngữ tự nhiên.
- Cuối 80: Hệ chuyên gia xử lý ngôn ngữ tự nhiên.
- 1981-đề án tạo ra các máy tính thế hệ 5 của Nhật.
- 1986,1987 đến nay-Phát triển mạng Neural và ứng dụng.

6. Các lĩnh vực ứng dụng của TTNT

- Xử lý ngôn ngữ tự nhiên và giao diện người máy.
- Lập luận và giải quyết vấn đề tự động.
- Chuẩn đoán, chữa trị với tri thức chuyên gia.
- Nhìn và nhận dạng.
- Xử lý âm thanh tiếng nói.
- Phát hiện tri thức tự động từ dữ liệu.
- Lập lịch, kế hoạch tự động.
- Xây dựng các trò chơi thông minh.
- Mô phỏng thông minh.
- Giải các bài toán xã hội, thiên nhiên thông qua mô phỏng thông minh.
- Cuộc sống nhân tạo.
-

Một số ví dụ về TTNT

- Chương trình chơi cờ trên máy Deep Blue đánh bại đại kiện tướng Kasparov (1997).
- Hệ chuyên gia MYCIN (1984, Standford) không thua kém chuyên gia người trong việc chuẩn đoán bệnh.
- Chiến tranh vùng vịnh 1991, Kỹ thuật TTNT được dùng để lập lịch và lên kế hoạch hậu cần.
- Chiến tranh vùng vịnh lần 2 (2003). Chiến tranh mô phỏng trên máy tính.
- Chương trình lập lịch và điều khiển thông minh trên xe tự hành và Robot tự hành của NASA.
- Máy nhận dạng mắt người tại sân bay Heathrow.
-

7. So sánh giữa lập trình hệ thống và lập trình AI

Lập trình hệ thống

- Dữ liệu + Thuật toán = Chương trình.
- Xử lý dữ liệu.
- Dữ liệu trong bộ nhớ được đánh địa chỉ số
- Xử lý theo các thuật toán.
- Định hướng xử lý các đại lượng định lượng số.
- Xử lý tuần tự theo mẻ.
- Không giải thích trong quá trình thực hiện.
- Kết quả chính xác, không được mắc lỗi.

Lập trình A.I

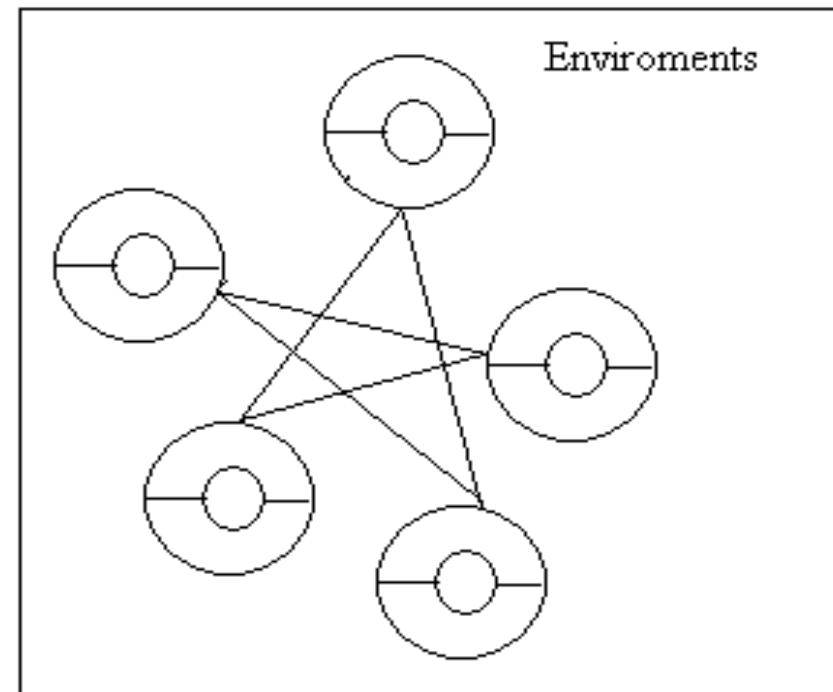
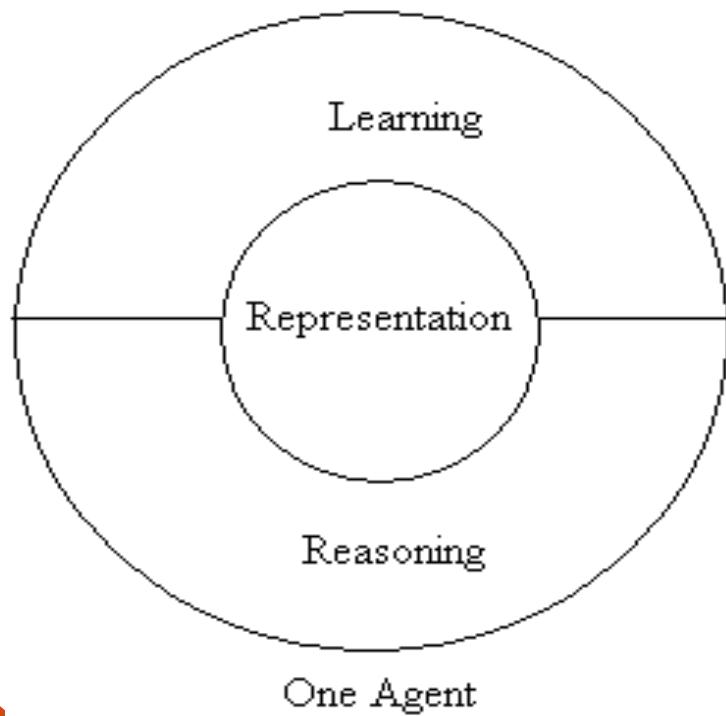
- Tri thức + Điều khiển =Chương trình.
- Xử lý dữ liệu định tính(các ký hiệu tương trưng).
- Xử lý dựa trên tri thức cho phép dùng các thuật giải heuristic, các cơ chế suy diễn.
- Tri thức được cấu trúc hoá, để trong bộ nhớ làm việc theo ký hiệu.
- Định hướng xử lý các đại lượng định tính (logic), các ký hiệu tương trưng và danh sách.
- Xử lý theo chế độ tương tác (hội thoại ngôn ngữ tự nhiên).
- Có giải thích hành vi của hệ thống trong quá trình thực hiện.
- Kết quả tốt, cho phép mắc lỗi.

8. Những vấn đề chưa được giải quyết

- Chương trình chưa tự sinh ra được heuristic
- Chưa có khả năng xử lý song song của con người
- Chưa có khả năng diễn giải một vấn đề theo nhiều phương pháp khác nhau như con người.
- Chưa có khả năng xử lý thông tin trong môi trường liên tục như con người.
- Chưa có khả năng học như con người.
- Chưa có khả năng tự thích nghi với môi trường.

9. Những vấn đề cơ bản của TTNT

- Biểu diễn (*representation*).
- Lập luận (*reasoning*).
- Học (*learning*).
- Tương tác (*interaction*).



NHẬP MÔN TRÍ TUỆ NHÂN TẠO

Chương 2: Logic hình thức

Biên soạn: TS Ngô Hữu Phúc

Bộ môn: Khoa học máy tính

Mobile: 098 56 96 580

Email: ngohuuphuc76@gmail.com

Thông tin chung

- Thông tin về nhóm môn học:

TT	Họ tên giáo viên	Học hàm	Học vị	Đơn vị công tác (Bộ môn)
1	Ngô Hữu Phúc	GVC	TS	BM Khoa học máy tính
2	Trần Nguyên Ngọc	GVC	TS	BM Khoa học máy tính
3	Hà Chí Trung	GVC	TS	BM Khoa học máy tính
4	Trần Cao Trưởng	GV	ThS	BM Khoa học máy tính

- Thời gian, địa điểm làm việc: Bộ môn Khoa học máy tính Tầng 2, nhà A1.
- Địa chỉ liên hệ: Bộ môn Khoa học máy tính, khoa Công nghệ thông tin.
- Điện thoại, email: 069-515-329, ngohuuphuc76.mta@gmail.com.

Cấu trúc môn học

- Chương 1: Giới thiệu chung.
- Chương 2: Logic hình thức.
- Chương 3: Các phương pháp tìm kiếm mù.
- Chương 4: Các phương pháp tìm kiếm có sử dụng thông tin.
- Chương 5: Các chiến lược tìm kiếm có đối thủ.
- Chương 6: Các bài toán thỏa ràng buộc.
- Chương 7: Nhập môn học máy.

Bài 2: Logic hình thức

Chương 2, mục: 2.1 – 2.4

Tiết: 1-3; 4-6; Tuần thứ: 2,3.

Mục đích, yêu cầu:

1. Nắm được Logic hình thức.
2. Nắm được sự tương đương logic.
3. Nắm được phương pháp lập luận và suy diễn sử dụng logic.

Hình thức tổ chức dạy học: Lý thuyết.

Thời gian: 3 tiết.

Địa điểm: Giảng đường do Phòng Đào tạo phân công

Nội dung chính: (Slides)

Nội Dung

- Lựa chọn hành động dựa trên tri thức.
- Hang Wumpus .
- Logic.
- Logic Mệnh đề.
- Tính tương đương, tính thoả được.
- Lập luận & chứng minh tự động trên Logic Mệnh đề
 - lập luận tiến
 - lập luận lùi
 - phép giải

Cơ Sở Tri Thức



- Cơ sở tri thức = tập các câu trong một ngôn ngữ hình thức nào đó
- Giải quyết vấn đề bằng đặc tả
 - Cơ sở tri thức (KB) biểu diễn điều mà agent cần biết
- Sau đó để giải quyết vấn đề chỉ cần ra lệnh “what to do?”. Cơ sở tri thức và cơ chế lập luận sẽ giúp agent tự giải quyết vấn đề.
- Do đó agent có thể được dùng tuỳ thuộc vào cấp độ tri thức chứ không phụ thuộc vào cài đặt. (cấu trúc dữ liệu, thuật toán, ...).

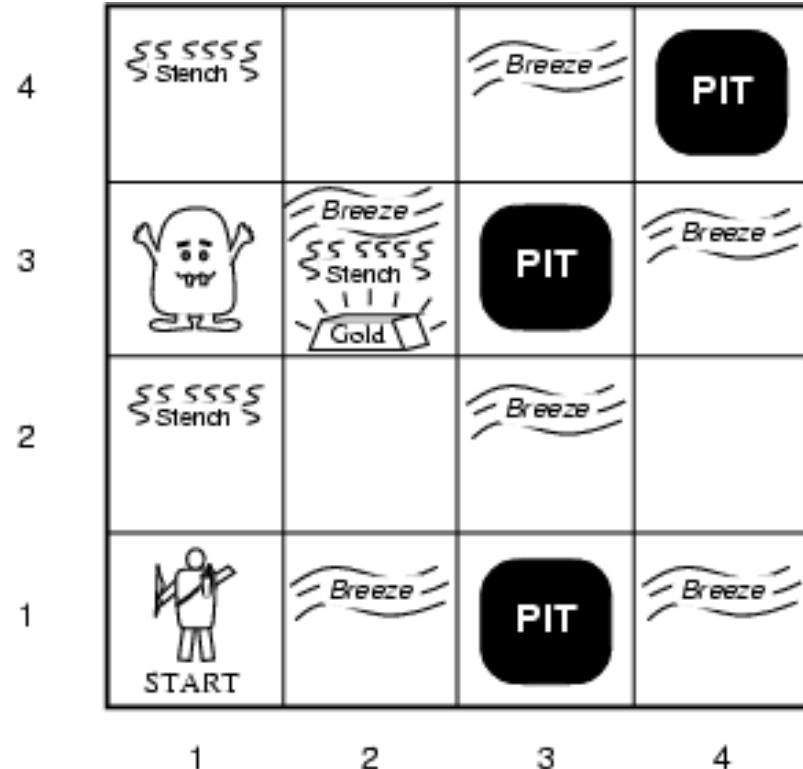
Khung mẫu cho Agent tựa tri thức

```
function KB-AGENT(percept) returns an action
    static: KB, a knowledge base
          t, a counter, initially 0, indicating time
    TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
    action  $\leftarrow$  ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t  $\leftarrow$  t + 1
    return action
```

- Agent phải có khả năng:
 - biểu diễn trạng thái, hành động etc.
 - Tiếp nạp dữ liệu mới từ bên ngoài.
 - Thay đổi nhận thức (biểu diễn) thế giới bên ngoài.
 - Suy diễn những sự kiện ẩn (không thấy) của thế giới bên ngoài
 - Dẫn đến hành động thích hợp trên cơ sở suy diễn.

Hang Wumpus

- Điểm hiệu quả
 - gold +1000, death -1000
 - -1 / 1 bước, -10 nếu dùng cung
- Môi Trường
 - ô cạnh ô có Wumpus có mùi thối.
 - Ô cạnh bẫy có tiếng gió thổi.
 - Ô bên cạnh ô đựng vàng có ánh kim
 - Bắn Wumpus nếu đối diện với nó.
 - Chỉ được dùng một mũi tên
 - Chộp lấy vàng nếu ở cùng ô
 - Thả vàng rơi trong cùng ô
- Sensors: mùi, tiếng gió, ánh kim, xóc, tiếng rên la.
- Actuators: quay trái, phải, tiến, chộp, thả, bắn.



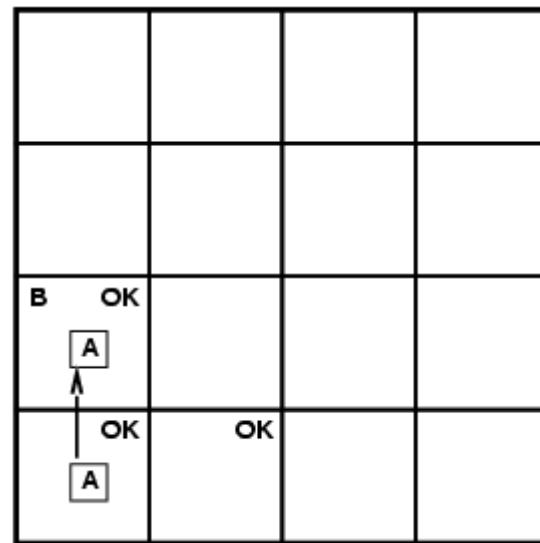
Đặc Điểm bài toán Hang Wumpus

- Quan sát tất cả các trạng thái? không – chỉ quan sát được cục bộ
- Đơn định? Có
- Lời giải? Dãy các hành động để đạt được điểm thưởng cao nhất.
- Tính động? Tĩnh – Wumpus và bẫy ở nguyên vị trí.
- Rời rạc Có

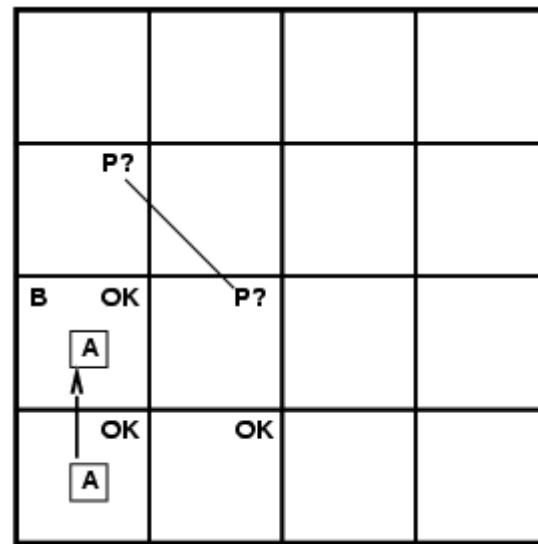
Ví dụ

OK			
OK	OK		

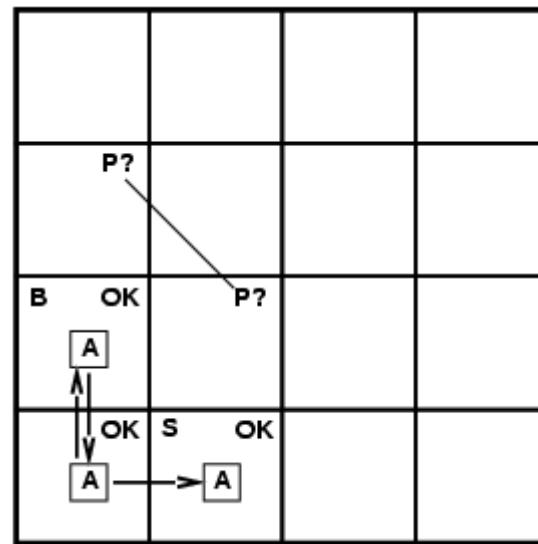
Ví dụ



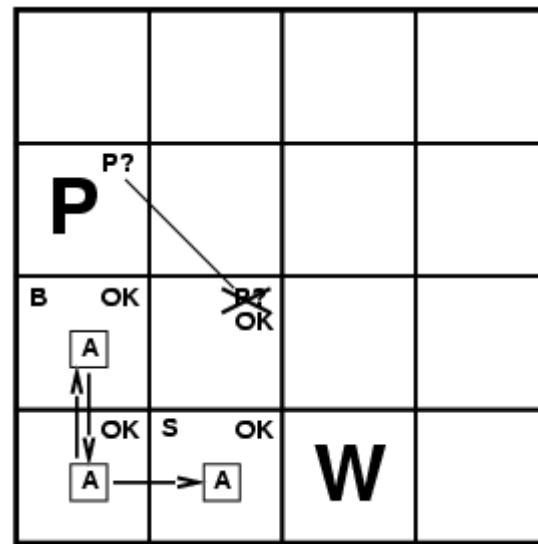
Ví dụ



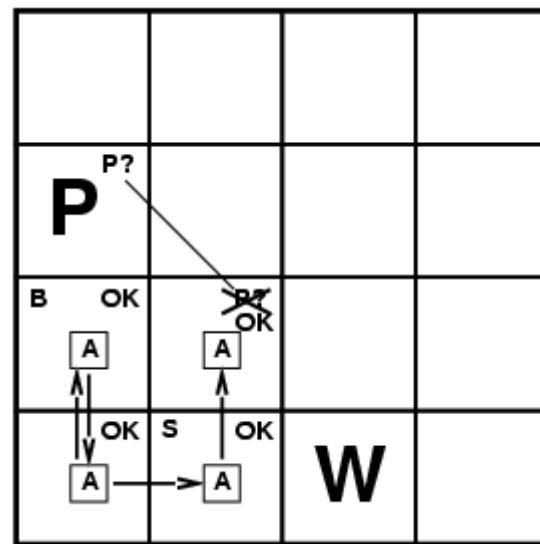
Ví dụ



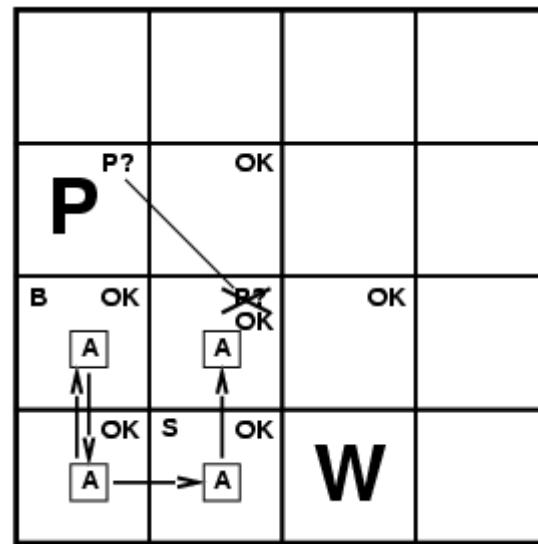
Ví dụ



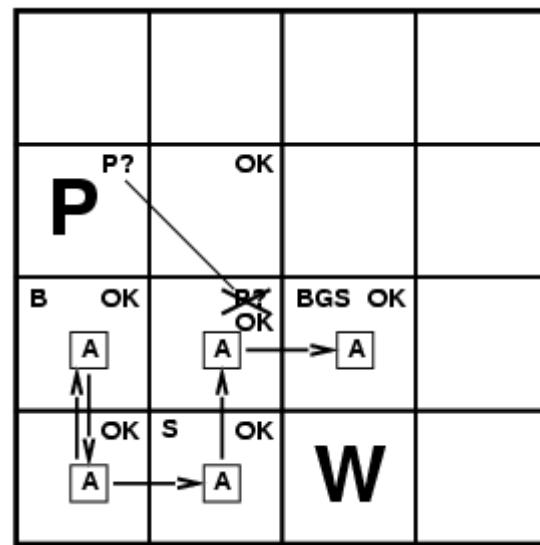
Ví dụ



Ví dụ



Ví dụ



Logic

- **Logics** ngôn ngữ hình thức biểu diễn thông tin như các kết luận có thể trích rút, suy diễn từ tri thức và quan sát môi trường xung quanh.
- **Cú pháp** định nghĩa cấu trúc câu cho Logic.
- **Ngữ nghĩa** xác định nghĩa của câu
 - i.e. xác lập tính đúng đắn của một mệnh đề trong hoàn cảnh (thế giới) cụ thể.
- **Ví dụ ngôn ngữ số học**
 - $x+2 \geq y$ là câu; $x^2+y > \{ \}$ không phải là câu
 - $x+2 \geq y$ là đúng nếu số $x+2$ không nhỏ hơn số y
 - $x+2 \geq y$ đúng khi $x = 7, y = 1$
 - $x+2 \geq y$ sai khi $x = 0, y = 6$

Hệ quả logic

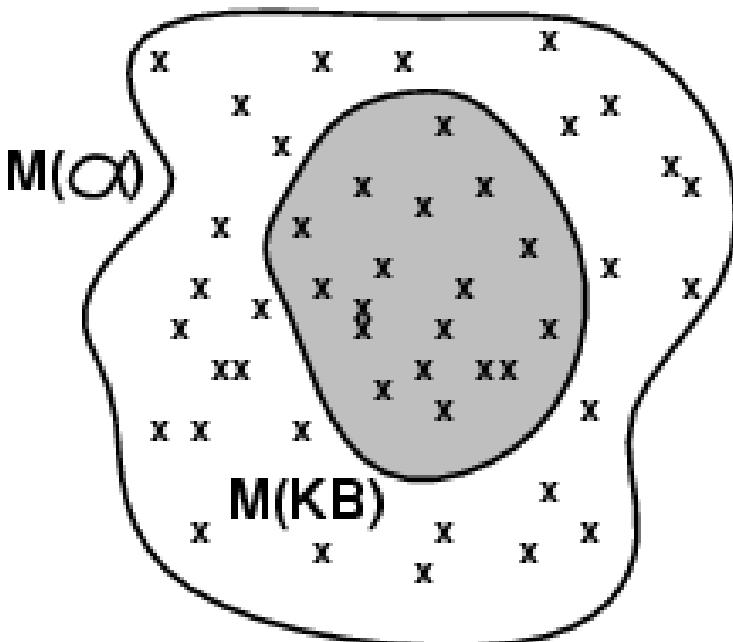
- Hệ quả logic là việc đúng của một (số) mệnh đề dẫn theo mệnh đề khác đúng

$$KB \models \alpha$$

- Cơ sở tt KB dẫn ra α (hay α là hệ quả Logic của KB) khi và chỉ khi α đúng trong mọi thế giới mà KB đúng.
 - VD KB có “đội MU thắng” và “Đội Chelsea thắng” dẫn ra “Một trong hai đội MU hoặc Chelsea thắng”
 - E.g., $x+y = 4$ dẫn ra $4 = x+y$
 - Quan hệ dẫn được (hệ quả logic) là mối quan hệ giữa các mệnh đề (i.e., cú pháp) dựa trên ngữ nghĩa.

Models

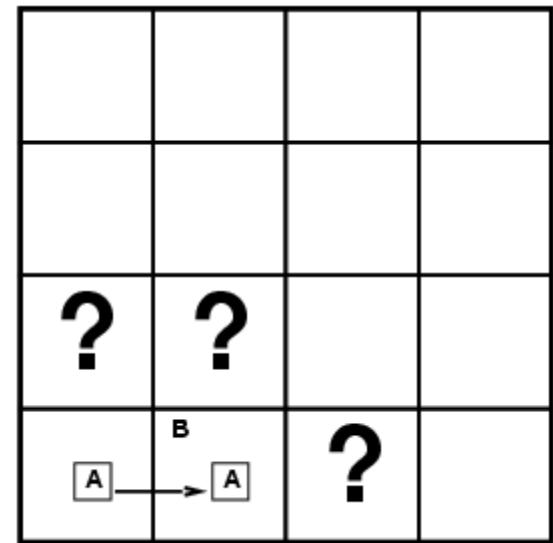
- **models**, thế giới (ngữ cảnh) mà tại đó các mệnh đề Logic được đánh giá tính đúng sai.
- Gọi m là model của mệnh đề α nếu α đúng trong m
- $M(\alpha)$ là tập tất cả các model của α
- Ta có KB $\models \alpha$ khi và chỉ khi
$$M(KB) \subseteq M(\alpha)$$



Ví dụ

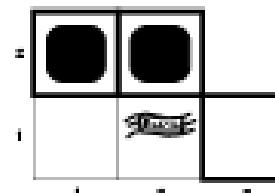
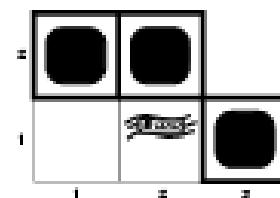
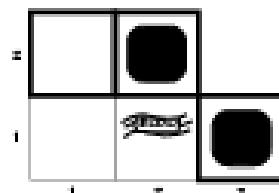
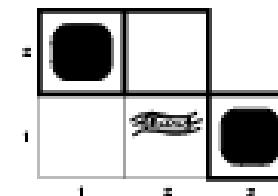
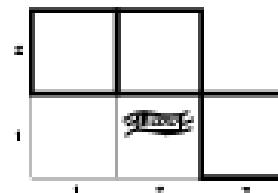
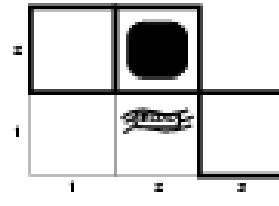
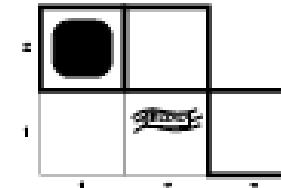
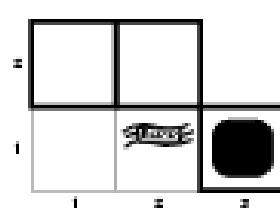
Sau khi xuất phát tại $[1,1]$,
sang phải, nghe tiếng
gió ở ô $[2,1]$

- Xét các mô hình có thể,
giải sử chỉ tính khả
năng có hay không có
hố ở các ô bên cạnh.

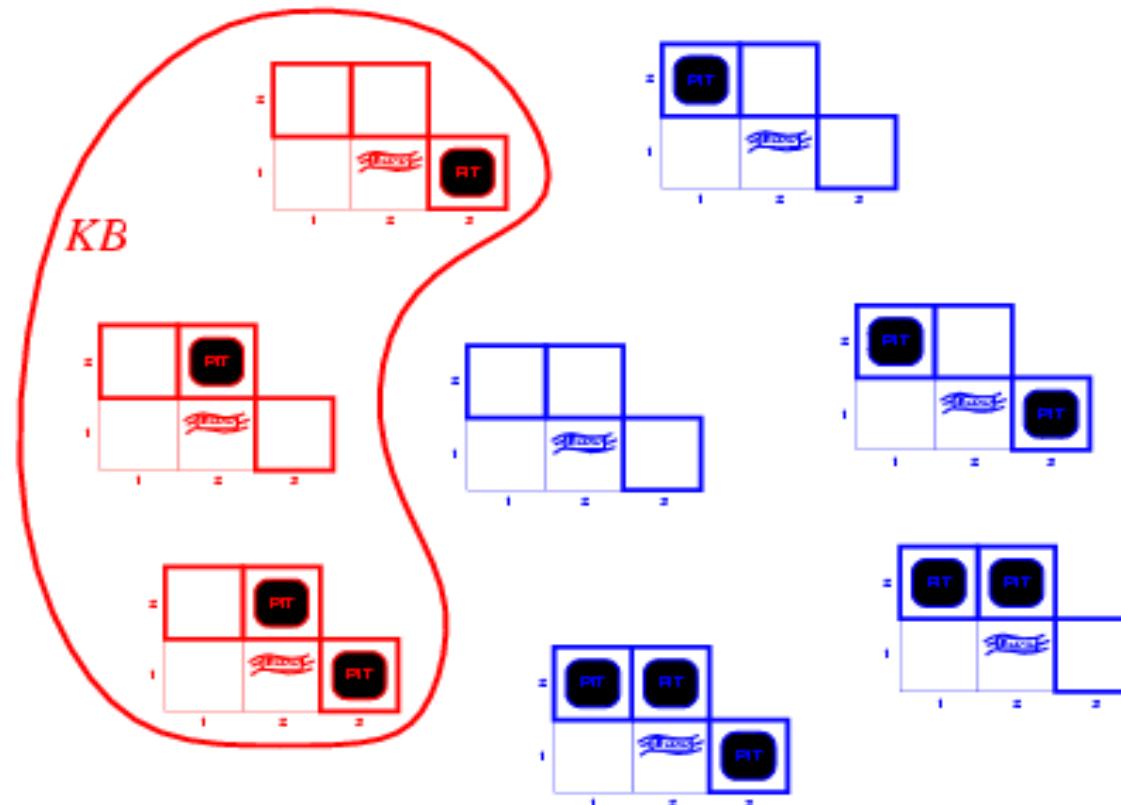


3 lựa chọn ô \Rightarrow 8 mô hình

Ví dụ

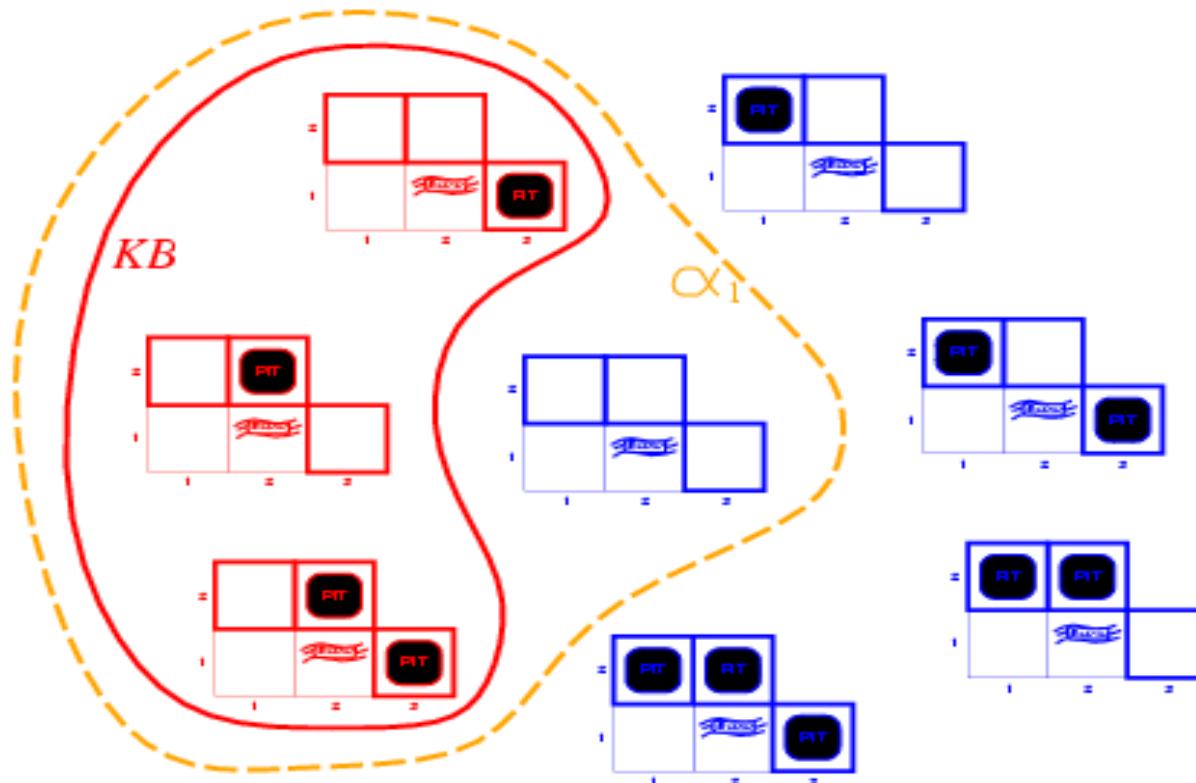


Ví dụ



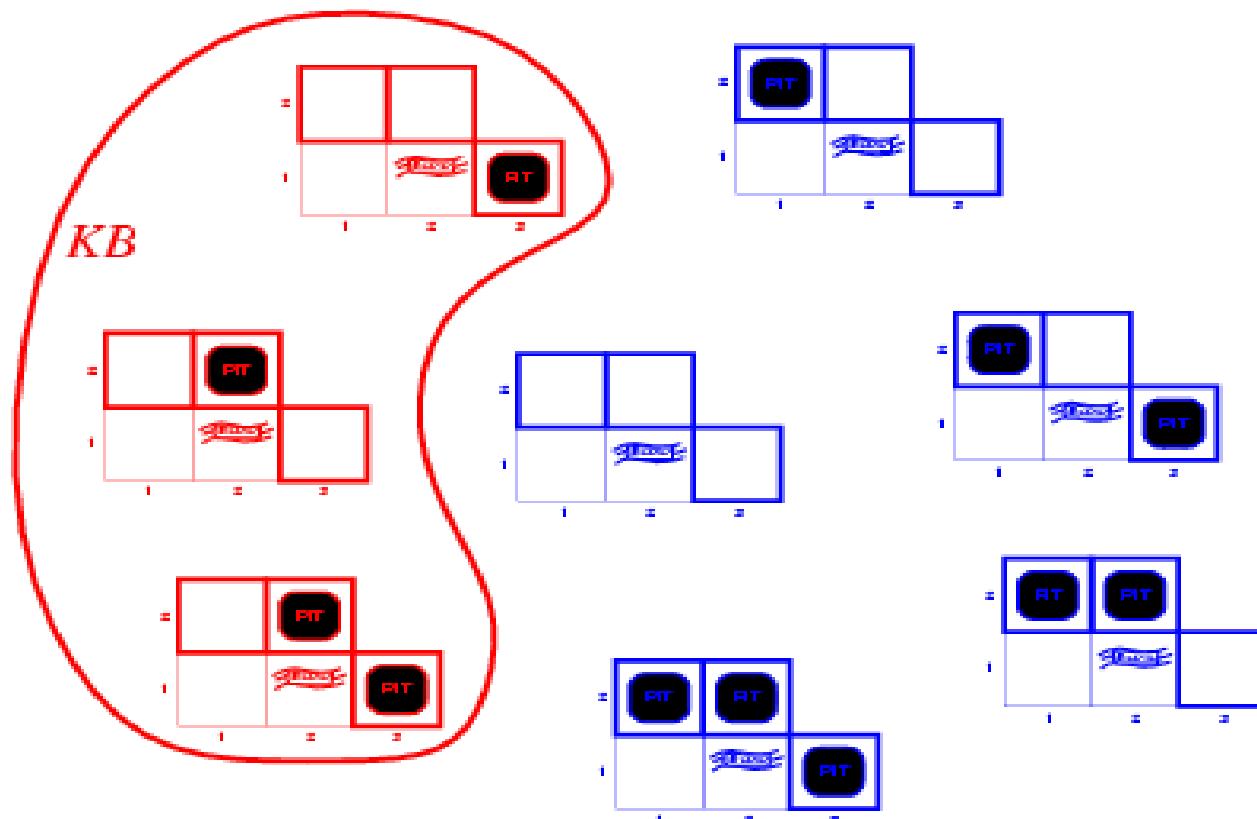
- $KB = \text{luật} + \text{quan sát, tiếp nhận từ môi trường}$

Ví dụ



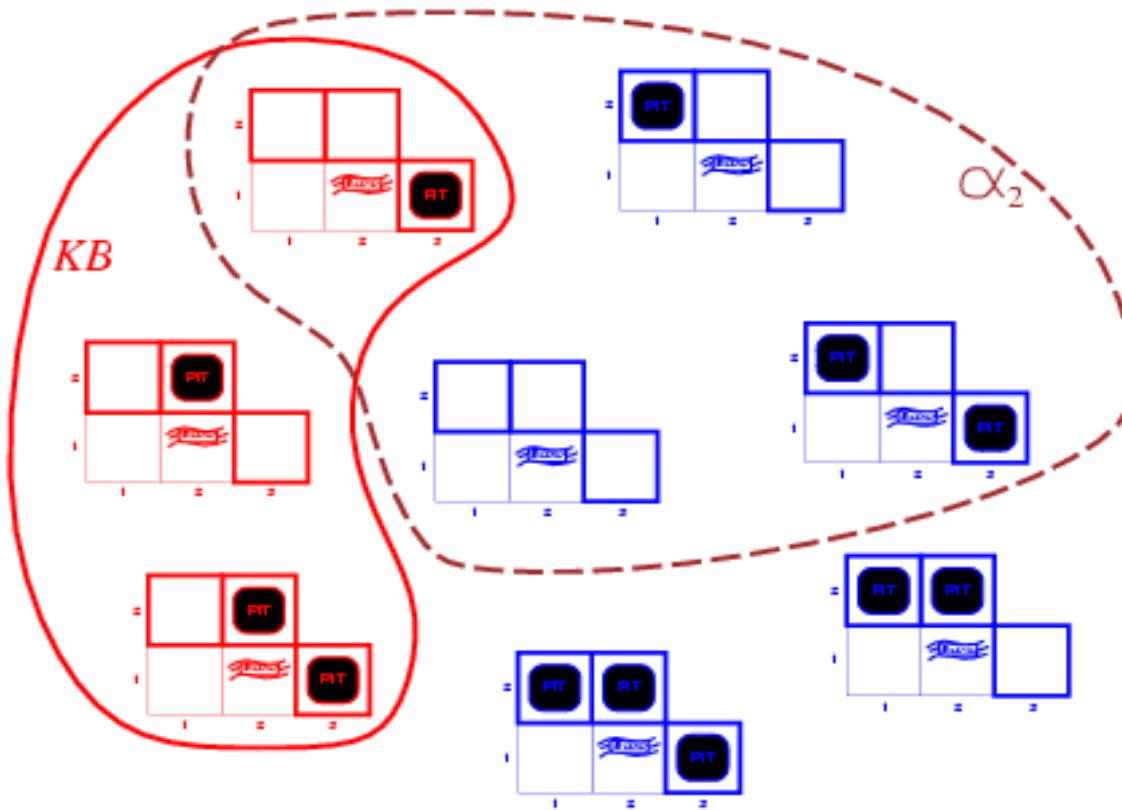
- $KB = \text{luật} + \text{quan sát tiếp nhận từ môi trường}$
- $\alpha_1 = "[1,2] là an toàn"$, $KB \models \alpha_1$, chứng minh bằng kiểm tra models

Ví dụ



- $KB = \text{luật} + \text{quan sát tiếp nhận từ môi trường}$

Ví dụ



- $KB = \text{luật} + \text{quan sát tiếp nhận từ môi trường}$
- $\alpha_2 = "[2,2] \text{ an toàn}", KB \not\models \alpha_2$

Lập Luận

- $KB \vdash_i \alpha$ = mệnh đề α dẫn được từ KB bằng thủ tục (cơ chế lập luận) i .
- Chặt: i là chặt khi và chỉ khi nếu $KB \vdash_i \alpha$, thì $KB \models \alpha$.
- Đủ: i là đủ khi và chỉ khi $KB \models \alpha$, thì $KB \vdash_i \alpha$

Logic Mệnh Đề: Cú pháp

- Logic mệnh đề là loại logic đơn giản nhất (tương đương đại số Boolean), dùng để biểu diễn tri thức bậc 0 (monadic).
- Giả sử $P_1, P_2 \dots$ là các mệnh đề
 - Nếu S là mệnh đề, $\neg S$ là mệnh đề
 - Nếu S_1 và S_2 là các mệnh đề, $S_1 \wedge S_2$ là mệnh đề
 - Nếu S_1 và S_2 là các mệnh đề, $S_1 \vee S_2$ là một mệnh đề
 - Nếu S_1 và S_2 là các mệnh đề, $S_1 \Rightarrow S_2$ là một mệnh đề
 - Nếu S_1 và S_2 là các mệnh đề, $S_1 \Leftrightarrow S_2$ là một mệnh đề

Logic mệnh đề: ngữ nghĩa

Mỗi model xác lập giá trị true/false cho mỗi ký hiệu mệnh đề

E.g.	$P_{1,2}$	$P_{2,2}$	$P_{3,1}$
	false	true	false

Với 3 mệnh đề thì có thể có 8 model có thể liệt kê đầy đủ

Luật để xác định giá trị dựa trên Model m :

$\neg S$	is true iff	S is false	
$S_1 \wedge S_2$	is true iff	S_1 is true and	S_2 is true
$S_1 \vee S_2$	is true iff	S_1 is true or	S_2 is true
$S_1 \Rightarrow S_2$	is true iff	S_1 is false or	S_2 is
true			
i.e.,	is false iff	S_1 is true and	S_2 is false
$S_1 \Leftrightarrow S_2$	is true iff	$S_1 \Rightarrow S_2$ is true	
and $S_2 \Rightarrow S_1$ is true			

Thực chất là đánh giá đệ quy:

$$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \text{true} \wedge (\text{true} \vee \text{false}) = \text{true} \wedge \text{true} = \text{true}$$

Bảng giá trị luận lý

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

Ví dụ

$P_{i,j}$ nhận giá trị đúng nếu có hố trong ô $[i, j]$.

$B_{i,j}$ nhận giá trị đúng nếu có tiếng gió trong ô $[i, j]$.

$$\neg P_{1,1}$$

$$\neg B_{1,1}$$

$$B_{2,1}$$

- “Hố tạo nên tiếng gió ở các ô xung quanh”

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$

$$B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$$

Lập Luận Dựa Trên Bảng Luận Lý

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
false	true							
false	false	false	false	false	false	true	false	true
:	:	:	:	:	:	:	:	:
false	true	false	false	false	false	false	false	true
false	true	false	false	false	false	true	<u>true</u>	<u>true</u>
false	true	false	false	false	true	false	<u>true</u>	<u>true</u>
false	true	false	false	false	true	true	<u>true</u>	<u>true</u>
false	true	false	false	true	false	false	false	true
:	:	:	:	:	:	:	:	:
true	false	false						

Lập Luận Bằng Liệt Kê Models

- Tìm kiếm (theo chiều sâu) và liệt kê các mô hình

```
function TT-ENTAILS?( $KB, \alpha$ ) returns true or false
```

$symbols \leftarrow$ a list of the proposition symbols in KB and α

```
return TT-CHECK-ALL( $KB, \alpha, symbols, []$ )
```

```
function TT-CHECK-ALL( $KB, \alpha, symbols, model$ ) returns true or false
```

```
if EMPTY?( $symbols$ ) then
```

```
    if PL-TRUE?( $KB, model$ ) then return PL-TRUE?( $\alpha, model$ )
```

```
    else return true
```

```
else do
```

```
     $P \leftarrow FIRST(symbols); rest \leftarrow REST(symbols)$ 
```

```
    return TT-CHECK-ALL( $KB, \alpha, rest, EXTEND(P, true, model)$ ) and
```

```
        TT-CHECK-ALL( $KB, \alpha, rest, EXTEND(P, false, model)$ )
```

- Với n mệnh đề, độ phức tạp thời gian $O(2^n)$, không gian $O(n)$

Quan hệ Logic Tương Đương

- Hai mệnh đề là khi và chỉ khi chúng cùng đúng trên các model : $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \text{ commutativity of } \wedge$$

$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \text{ commutativity of } \vee$$

$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \text{ associativity of } \wedge$$

$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \text{ associativity of } \vee$$

$$\neg(\neg\alpha) \equiv \alpha \text{ double-negation elimination}$$

$$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha) \text{ contraposition}$$

$$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta) \text{ implication elimination}$$

$$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \text{ biconditional elimination}$$

$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \text{ de Morgan}$$

$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \text{ de Morgan}$$

$$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \text{ distributivity of } \wedge \text{ over } \vee$$

$$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \text{ distributivity of } \vee \text{ over } \wedge$$

Tính chân lý và Thoả được

Một mệnh đề được gọi là chân lý (tòan đúng) nếu nó đúng trên mọi models,

e.g., True , $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Liên hệ với phép suy dẫn qua định lý suy diễn:

$KB \models \alpha$ khi và chỉ khi $(KB \Rightarrow \alpha)$ là chân lý

Một mệnh đề gọi là thoả được nếu nó đúng trên một số model nào đó:

e.g., $A \vee B$, C

Một mệnh đề gọi là toàn sai nếu nó sai trên mọi model

e.g., $A \wedge \neg A$

Liên hệ với phép suy dẫn;

$KB \models \alpha$ khi và chỉ khi $(KB \wedge \neg \alpha)$ là toàn sai.

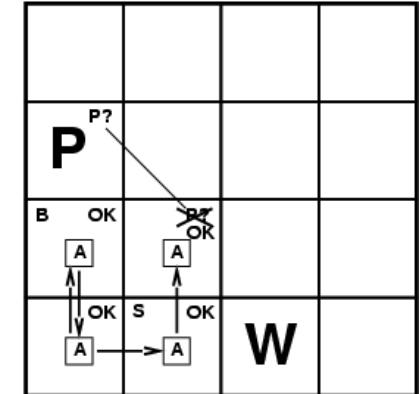
Phương Pháp Chứng Minh (suy dẫn)

- Chia làm hai loại
 - Áp dụng luật suy diễn:
 - Sinh hợp lệ các mệnh đề mới từ mệnh đề cũ.
 - Chứng minh = Dãy các áp dụng luật suy diễn, có thể dùng các luật suy diễn như toán tử chuyển trạng trong các thuật toán tìm kiếm.
 - Kiểm tra Model
 - Xét bảng luận lý (tỷ lệ mũ với n)
 - Cải tiến Back-Tracking, Davis--Putnam-Logemann-Loveland (DPLL)
 - Tìm kiếm Heuristic trong không gian các Model (chặt nhưng không đủ)

Phép Giải

Dạng chuẩn (CNF)

E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$



- Luật suy dẫn bằng phép giải (cho CNF):

$$\frac{f_i \vee \dots \vee f_k, \quad m_1 \vee \dots \vee m_n}{}$$

$$f_i \vee \dots \vee f_{i-1} \vee f_{i+1} \vee \dots \vee f_k \vee m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n$$

trong đó f_i và m_j là các literal nghịch nhau:

$$\text{E.g., } \frac{P_{1,3} \vee P_{2,2}, \quad P_{1,3}}{\neg P_{2,2}}$$

- Phép giải là chặt và dù đối với Logic mệnh đề.

Chuyển công thức sang dạng CNF

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})\beta$$

1. Khử \Leftrightarrow : thay $\alpha \Leftrightarrow \beta$ bằng $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.

$$(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$$

2. Khử \Rightarrow : Thay $\alpha \Rightarrow \beta$ bằng $\neg\alpha \vee \beta$.

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$$

3. Đẩy \neg vào trong: sử dụng luật de Morgan's và phủ định của phủ định :

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$$

4. Áp dụng luật phân phối :

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$$

Thuật Toán cho Phép Giải

- chứng minh bằng phản chứng, i.e., chứng minh rằng $KB \wedge \neg\alpha$ là luôn sai

```
function PL-RESOLUTION( $KB, \alpha$ ) returns true or false
```

```
clauses  $\leftarrow$  the set of clauses in the CNF representation of  $KB \wedge \neg\alpha$ 
```

```
new  $\leftarrow \{ \}$ 
```

```
loop do
```

```
    for each  $C_i, C_j$  in clauses do
```

```
        resolvents  $\leftarrow$  PL-RESOLVE( $C_i, C_j$ )
```

```
        if resolvents contains the empty clause then return true
```

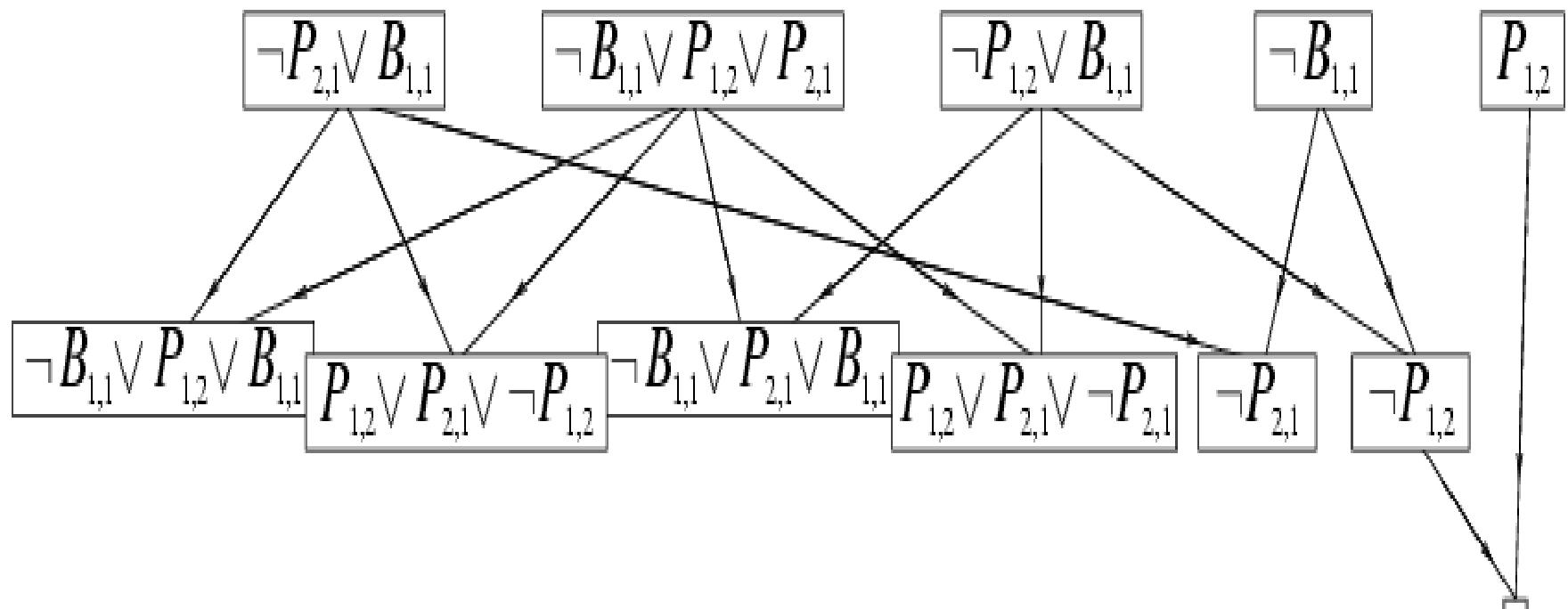
```
        new  $\leftarrow$  new  $\cup$  resolvents
```

```
    if new  $\subseteq$  clauses then return false
```

```
    clauses  $\leftarrow$  clauses  $\cup$  new
```

Ví dụ

- $KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$ $\alpha = \neg P_{1,2}$



Lập Luận Tiến/Lùi

- Dạng Horn (restricted)
 - KB = kết hợp của các câu dạng Horn
 - Câu dạng Horn =
 - ký hiệu mệnh đề; hoặc
 - (hợp (and) các mệnh đề) \Rightarrow mệnh đề
 - E.g., $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$
 -
 - Tam đoạn luận (cho dạng Horn): đủ Horn KBs

$$\frac{\alpha_1, \dots, \alpha_n, \quad \alpha_1 \wedge \dots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

- có thể cài đặt cơ chế lập luận hướng tiến/lùi.
- Các thuật toán này tự nhiên và chạy với thời gian đa thức.

Lập luận tiến

- Ý tưởng: “cháy” luật có phần tiền đề thỏa được trong KB , sau đó thêm phần kết luậ vào KB cho đến khi tìm được đích (trả lời câu hỏi) cần tìm)

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

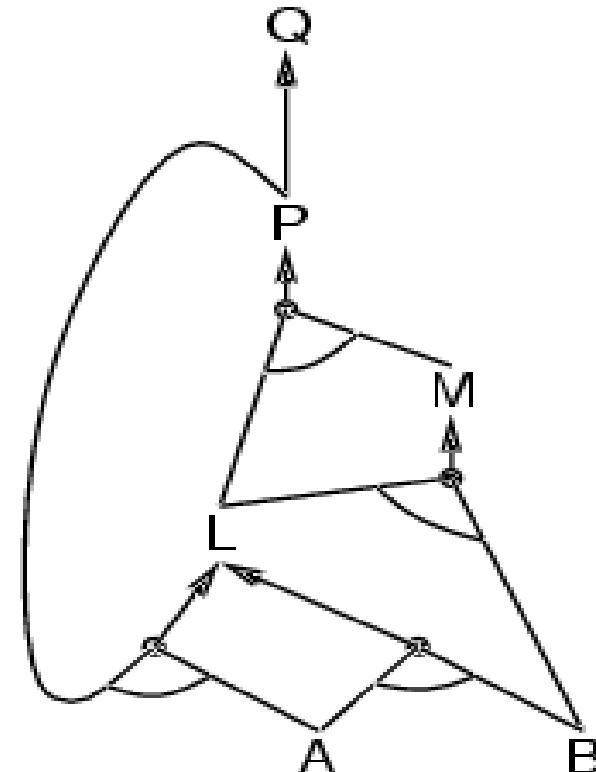
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

$$A$$

$$B$$



Thuật Toán cho lập luận tiền

function PL-FC-ENTAILS?(KB, q) returns *true or false*

local variables: *count*, a table, indexed by clause, initially the number of premises
inferred, a table, indexed by symbol, each entry initially *false*
agenda, a list of symbols, initially the symbols known to be true

while *agenda* is not empty do

$p \leftarrow \text{POP}(\text{agenda})$

 unless *inferred*[p] do

inferred[p] $\leftarrow \text{true}$

 for each Horn clause c in whose premise p appears do

 decrement *count*[c]

 if *count*[c] = 0 then do

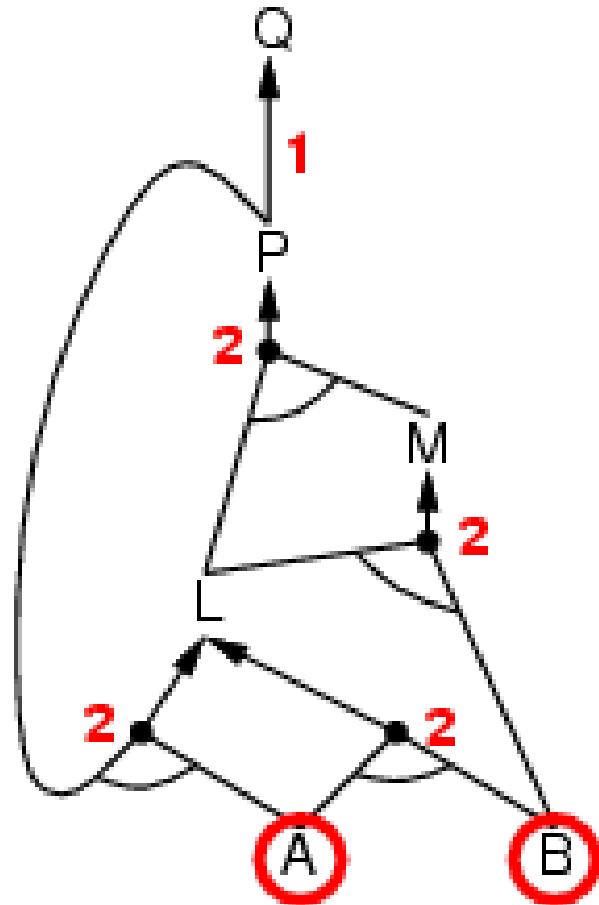
 if HEAD[c] = q then return *true*

 PUSH(HEAD[c], *agenda*)

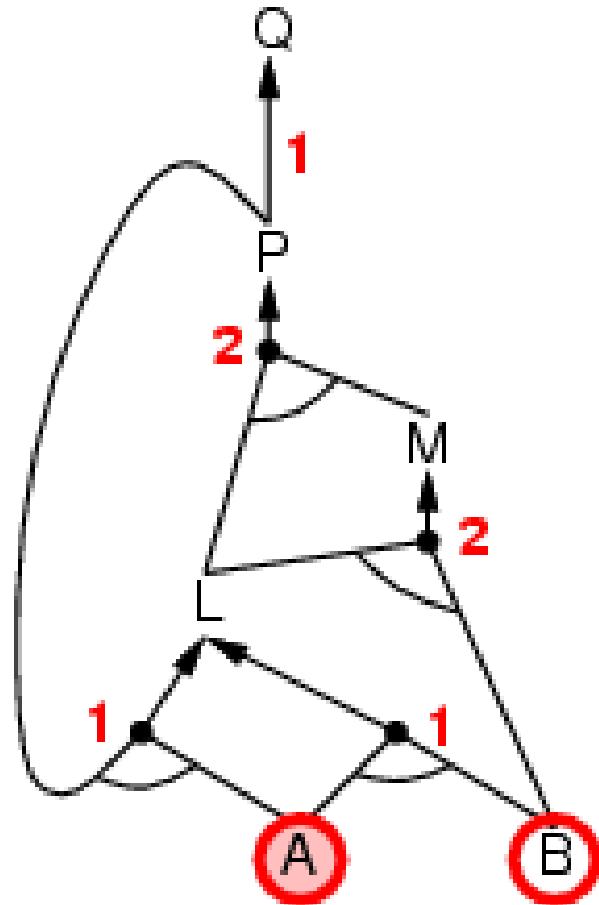
 return *false*

- Lập luận tiền là chặt & đủ đối với các KB dạng Horn

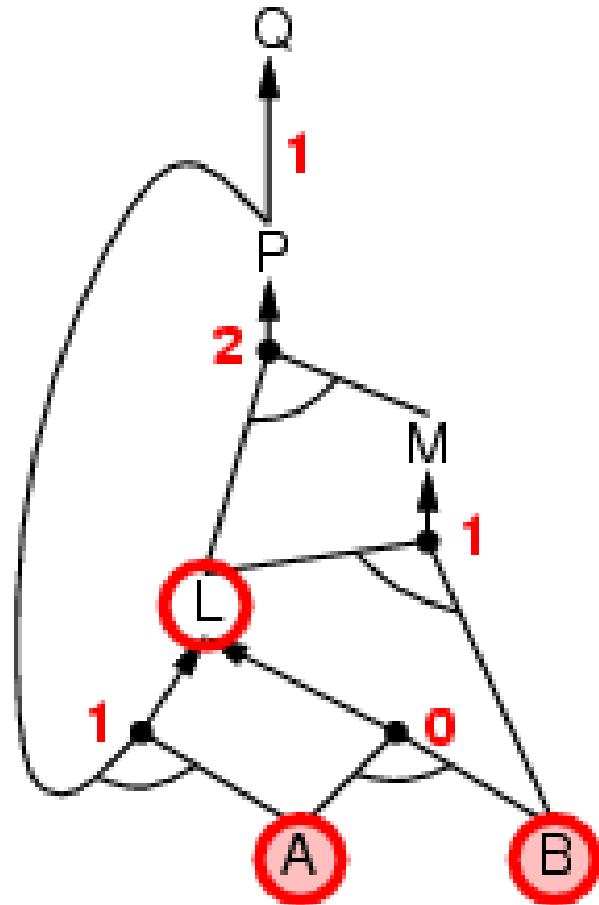
Ví dụ minh họa



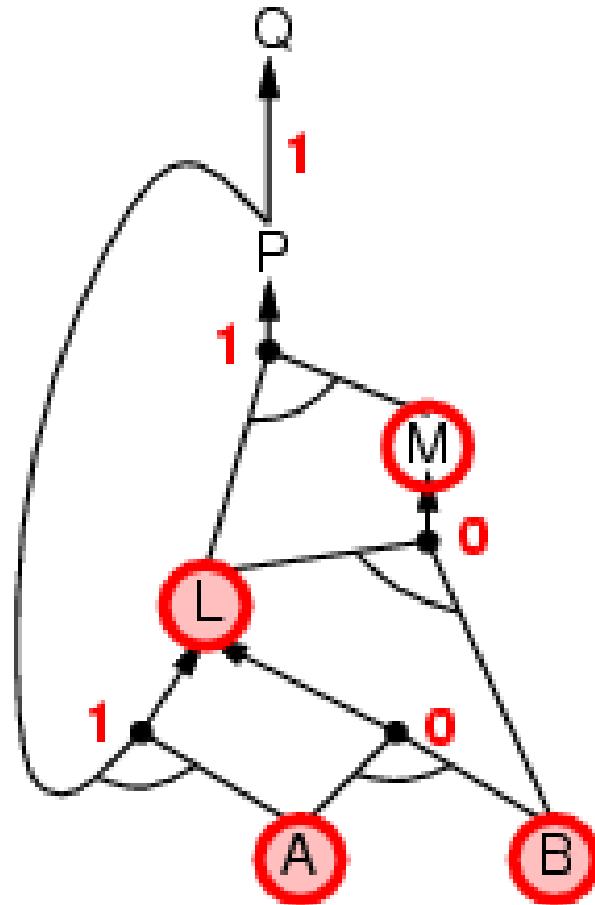
Ví dụ minh họa



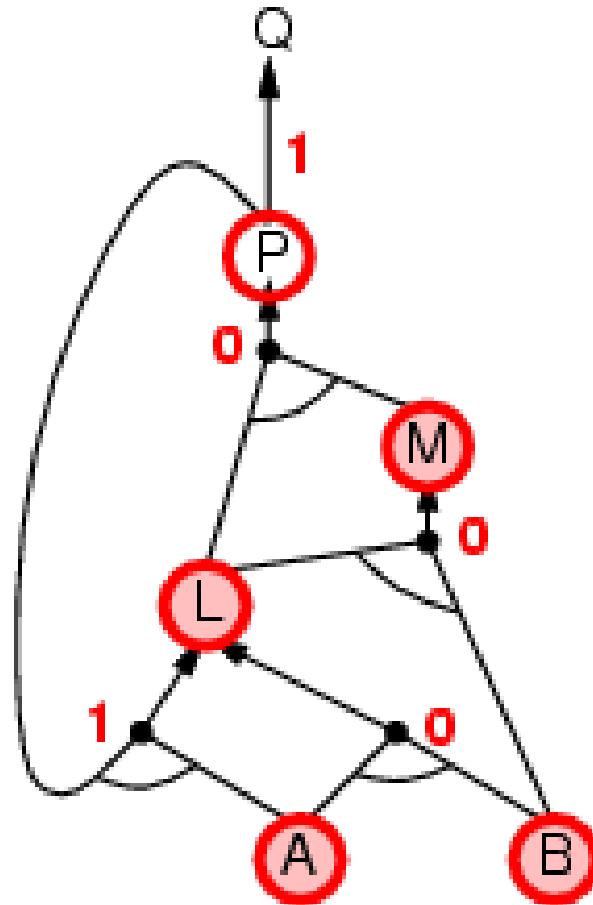
Ví dụ minh họa



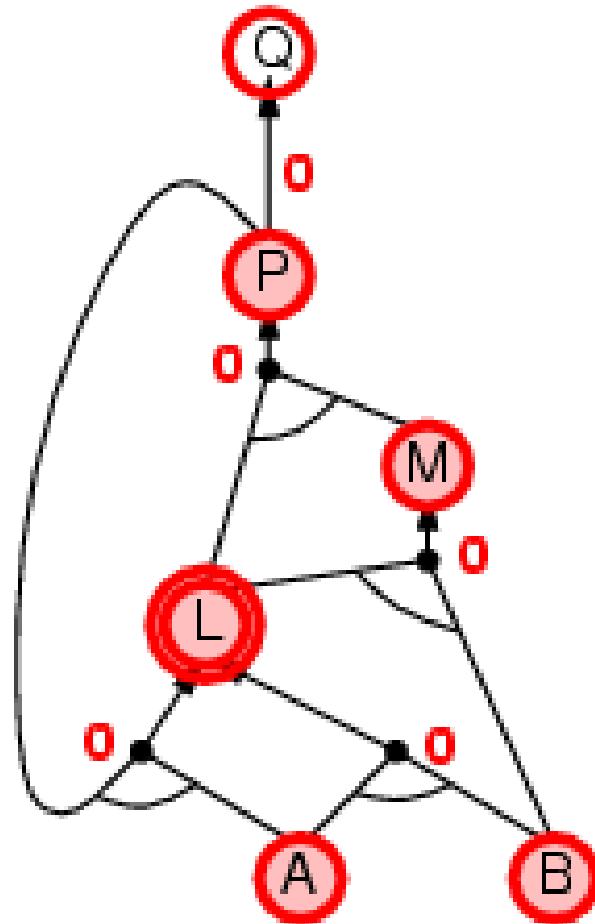
Ví dụ minh họa



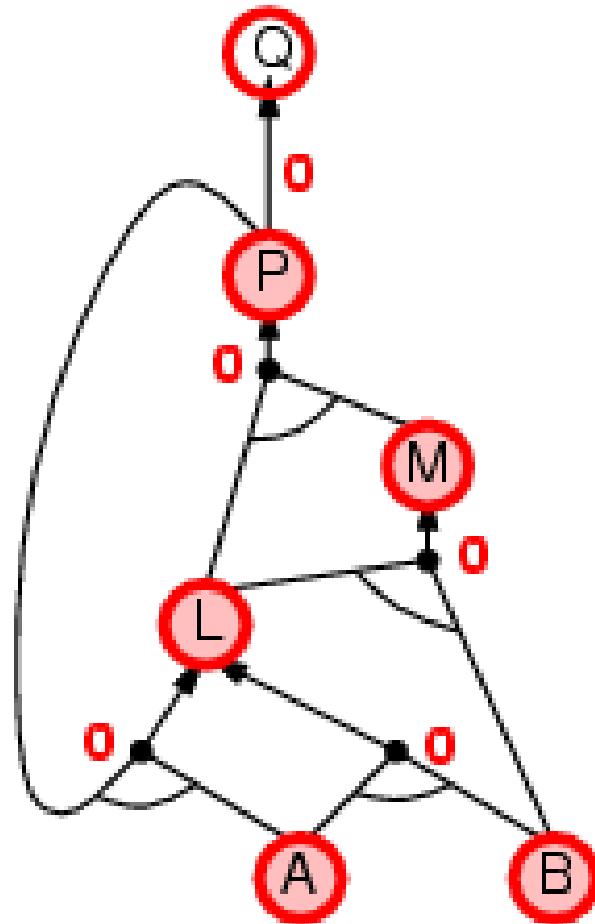
Ví dụ minh họa



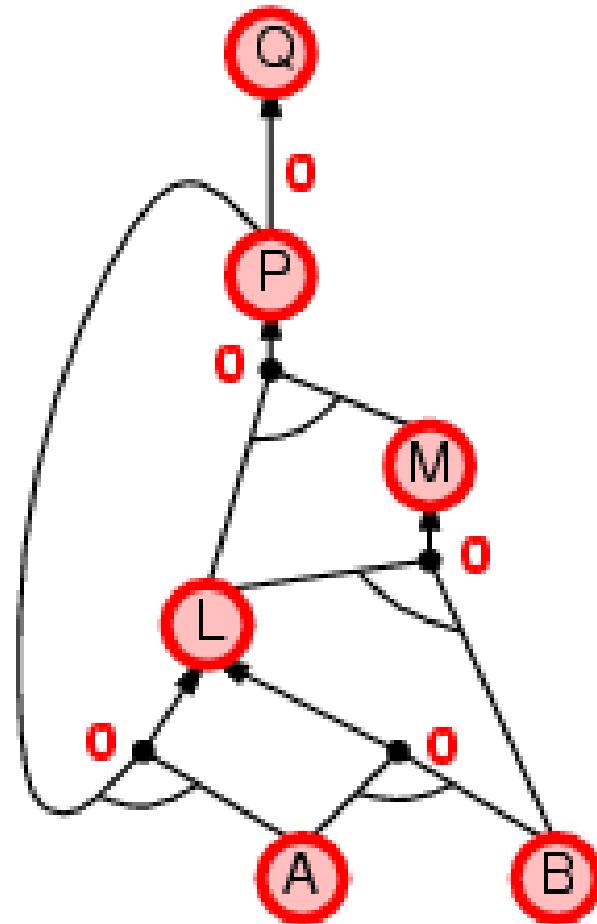
Ví dụ minh họa



Ví dụ minh họa



Ví dụ minh họa



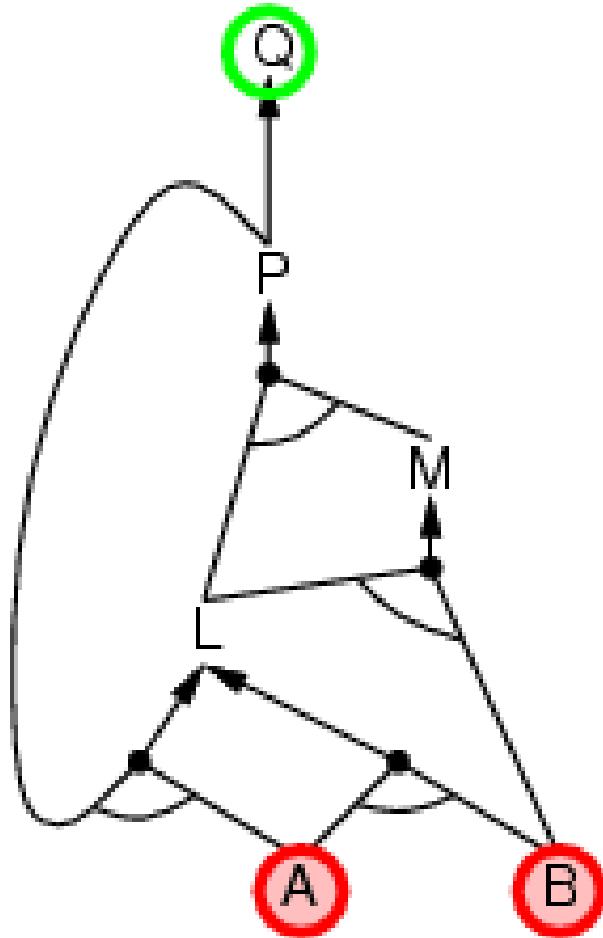
Lập luận lùi

Ý tưởng: lần ngược từ câu hỏi q :
để chứng minh q

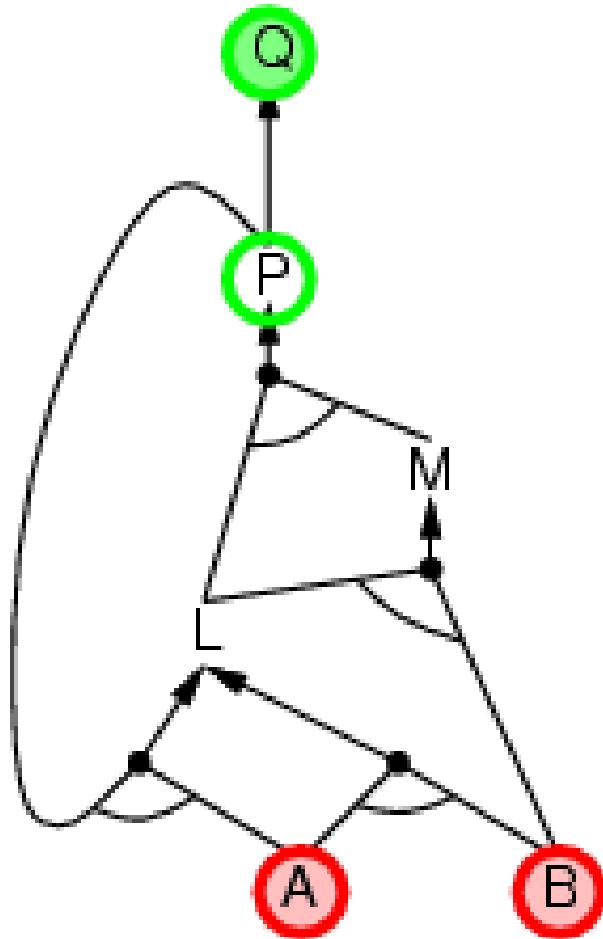
kiểm tra xem q đã được dẫn ra chưa, nếu không
tìm cách chứng minh các mệnh đề ở phần đầu luật
có chứa q ở phần kết luận.

Tránh lặp quẩn: Lưu trữ các đích đã được chứng
minh và trước khi chứng minh kiểm tra xem
đích cần chứng minh đã có trong goal stack
chưa?

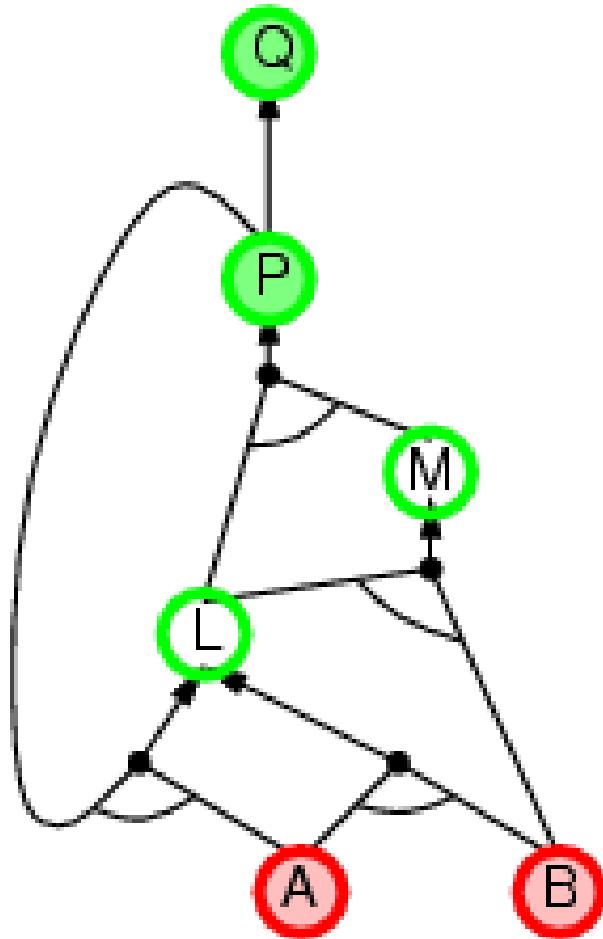
Ví Dụ Minh Họa



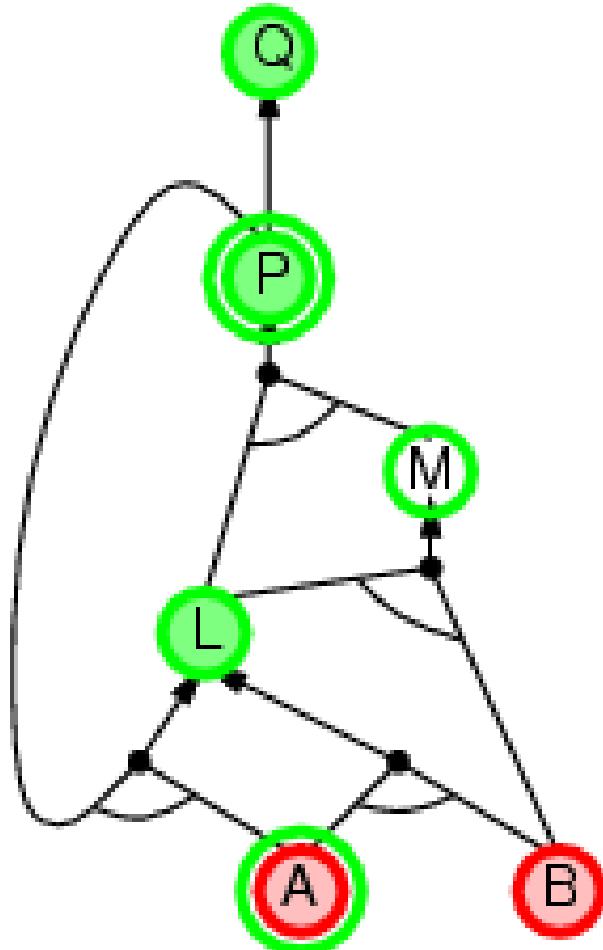
Ví Dụ Minh Họa



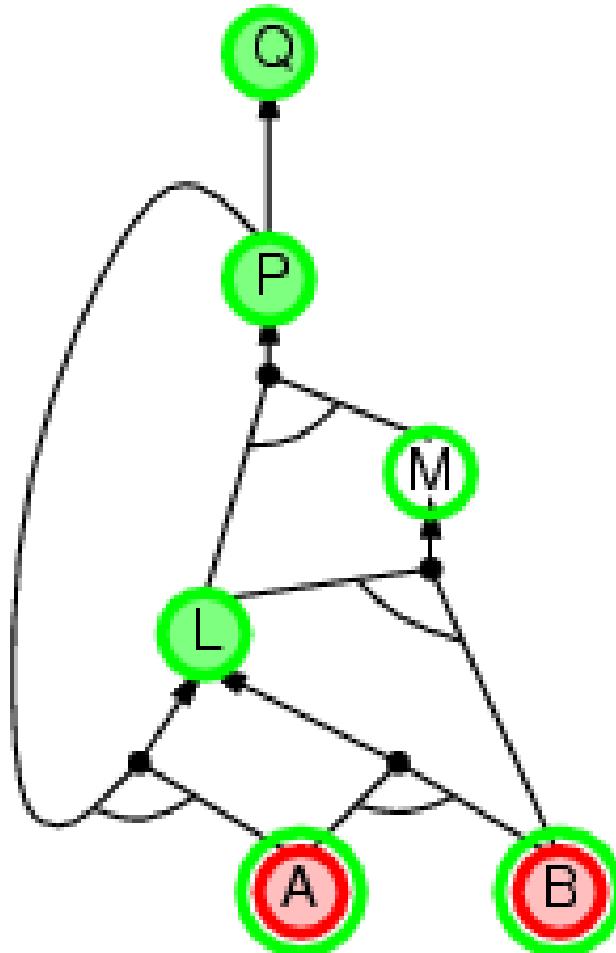
Ví Dụ Minh Họa



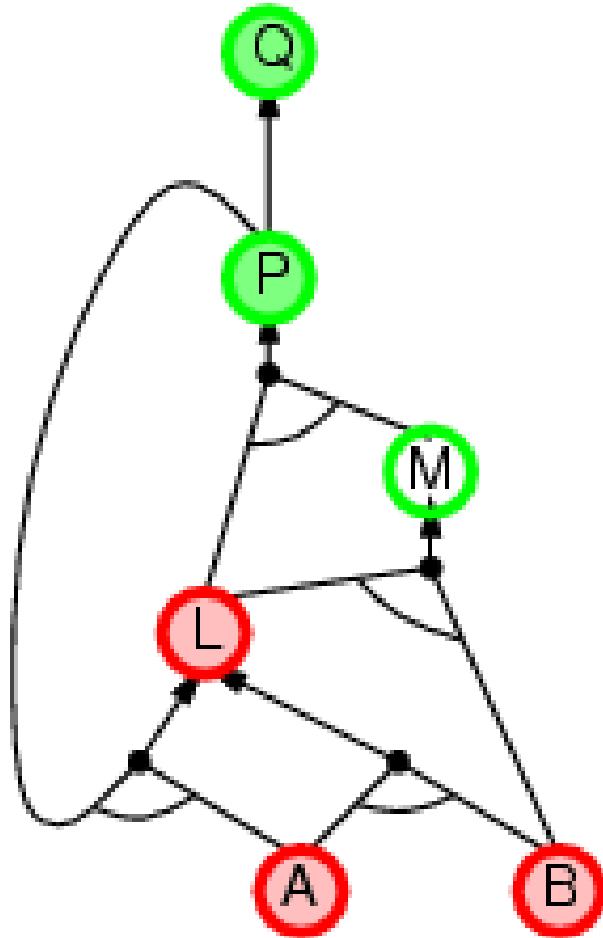
Ví Dụ Minh Họa



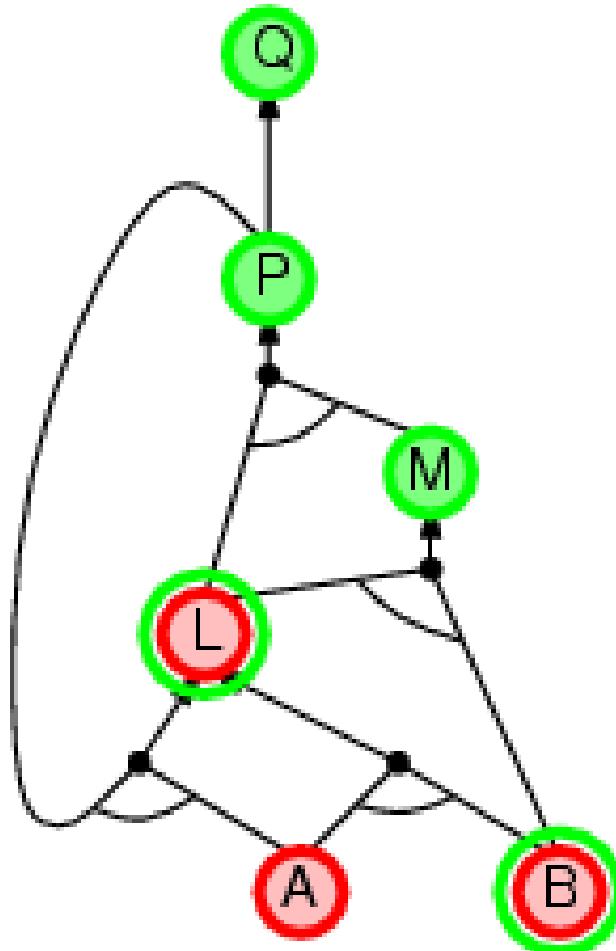
Ví Dụ Minh Họa



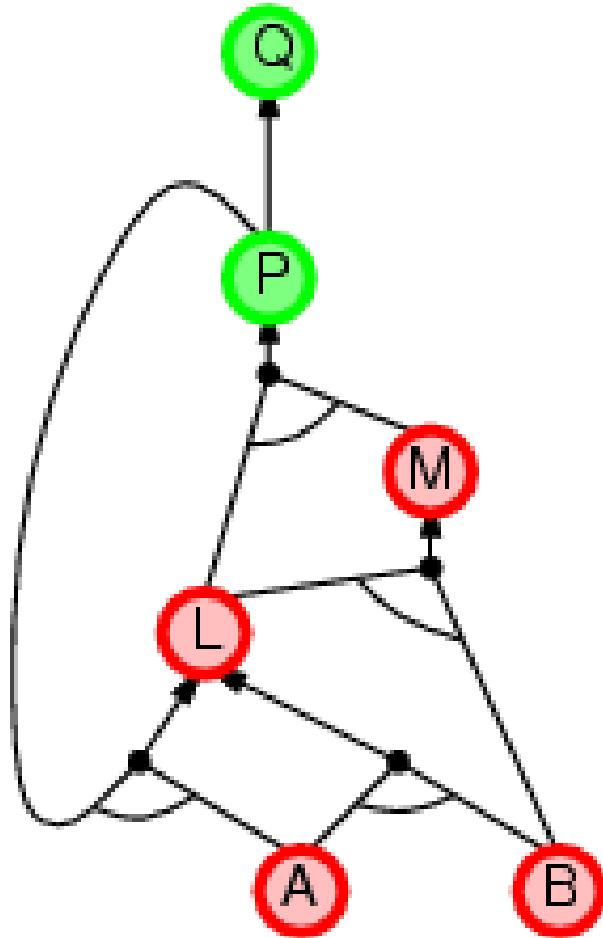
Ví Dụ Minh Họa



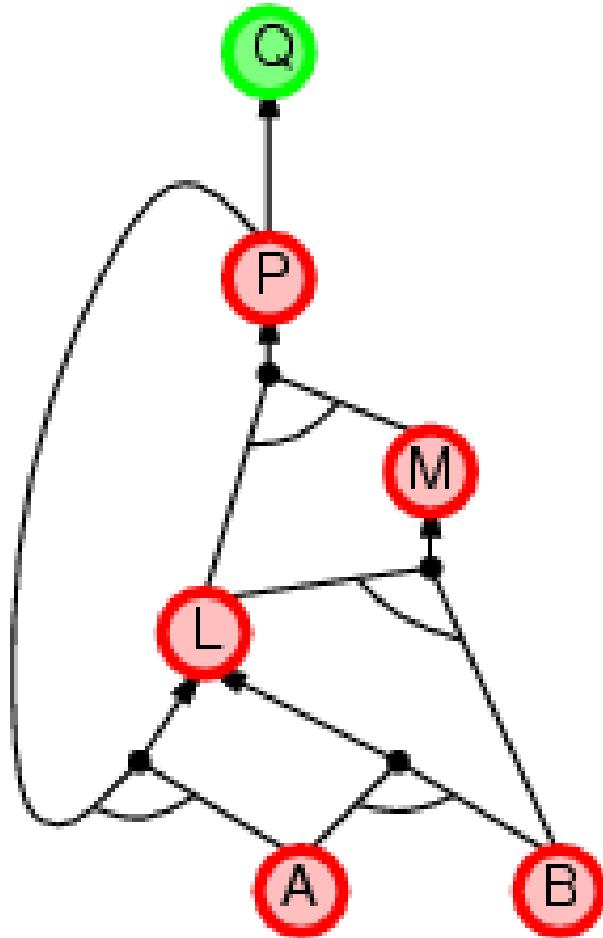
Ví Dụ Minh Họa



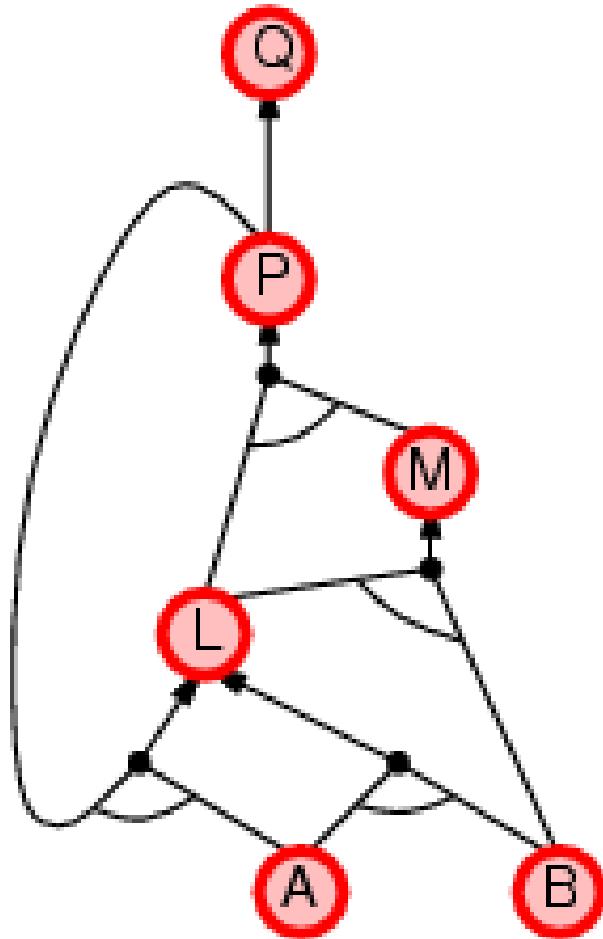
Ví Dụ Minh Họa



Ví Dụ Minh Họa



Ví Dụ Minh Họa



So Sánh Lập Luận Tiến/Lùi

- FC hướng dữ liệu, tự động, xử lý không hướng đích.
 - e.g., nhận dạng, ra quyết định,...
- chứng minh nhiều thứ không liên quan đến đích
- BC hướng đích, thích hợp cho giải quyết vấn đề, chuẩn đoán nguyên nhân,...

Một Số Thuật Toán Kiểm Tra Model cho Bài Toán Thoả Được

1. BackTracking (đầy đủ):

DPLL (Davis, Putnam, Logemann, Loveland)

2. Tìm kiếm địa phương (không đầy đủ):

- WalkSAT

Giải Thuật DPLL

Kiểm tra tính thoả được của công thức Logic ở dạng CNF

Liệt kê mô hình với một số Heuristics:

1. Kết thúc sớm:

Một câu là đúng nếu có một literal là đúng.

Một câu là sai nếu tất cả các literal là sai (do đó toàn công thức sai)

2. Mệnh đề nhất quán

luôn xuất hiện với cùng một dấu trong mọi câu

e.g., $(A \vee \neg B)$, $(\neg B \vee \neg C)$, $(C \vee A)$, A và B nhất quán, C không nhất quán.
cho mệnh đề nhất quán bằng true.

3. Câu đơn vị

Câu đơn vị: câu chỉ có một literal

Literal trong câu đó phải bằng true.

Giải Thuật DPLL

function DPLL-SATISFIABLE?(*s*) **returns** *true* or *false*

inputs: *s*, a sentence in propositional logic

clauses \leftarrow the set of clauses in the CNF representation of *s*

symbols \leftarrow a list of the proposition symbols in *s*

return DPLL(*clauses, symbols, []*)

function DPLL(*clauses, symbols, model*) **returns** *true* or *false*

if every clause in *clauses* is true in *model* **then return** *true*

if some clause in *clauses* is false in *model* **then return** *false*

P, value \leftarrow FIND-PURE-SYMBOL(*symbols, clauses, model*)

if *P* is non-null **then return** DPLL(*clauses, symbols-P, [P = value | model]*)

P, value \leftarrow FIND-UNIT-CLAUSE(*clauses, model*)

if *P* is non-null **then return** DPLL(*clauses, symbols-P, [P = value | model]*)

P \leftarrow FIRST(*symbols*); *rest* \leftarrow REST(*symbols*)

return DPLL(*clauses, rest, [P = true | model]*) **or**

DPLL(*clauses, rest, [P = false | model]*)

Giải Thuật WalkSAT

- tìm kiếm địa phương – không đầy đủ.
- hàm lượng giá: min-conflict heuristic - tối thiểu hóa số lượng các câu không thoả.
- Cân bằng giữa chiến lược tham ăn và chiến lược tìm kiếm ngẫu nhiên.

Giải Thuật WalkSAT

```
function WALKSAT(clauses, p, max-flips) returns a satisfying model or failure
inputs: clauses, a set of clauses in propositional logic
        p, the probability of choosing to do a “random walk” move
        max-flips, number of flips allowed before giving up

model  $\leftarrow$  a random assignment of true/false to the symbols in clauses
for i = 1 to max-flips do
    if model satisfies clauses then return model
    clause  $\leftarrow$  a randomly selected clause from clauses that is false in model
    with probability p flip the value in model of a randomly selected symbol
        from clause
    else flip whichever symbol in clause maximizes the number of satisfied clauses
return failure
```

Agent có khả năng lập luận cho bài toán hang Wumpus

Sử dụng cơ sở tri thức dạng logic mệnh đề:

$$\neg P_{1,1}$$

$$\neg W_{1,1}$$

$$B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y})$$

$$S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y})$$

$$W_{1,1} \vee W_{1,2} \vee \dots \vee W_{4,4}$$

$$\neg W_{1,1} \vee \neg W_{1,2}$$

$$\neg W_{1,1} \vee \neg W_{1,3}$$

...

⇒ 64 mệnh đề, 155 câu

```

function PL-WUMPUS-AGENT(percept) returns an action
  inputs: percept, a list, [stench,breeze,glitter]
  static: KB, initially containing the “physics” of the wumpus world
    x, y, orientation, the agent’s position (init. [1,1]) and orient. (init. right)
    visited, an array indicating which squares have been visited, initially false
    action, the agent’s most recent action, initially null
    plan, an action sequence, initially empty

  update x,y,orientation, visited based on action
  if stench then TELL(KB,  $S_{x,y}$ ) else TELL(KB,  $\neg S_{x,y}$ )
  if breeze then TELL(KB,  $B_{x,y}$ ) else TELL(KB,  $\neg B_{x,y}$ )
  if glitter then action  $\leftarrow$  grab
  else if plan is nonempty then action  $\leftarrow$  POP(plan)
  else if for some fringe square  $[i,j]$ , ASK(KB,  $(\neg P_{i,j} \wedge \neg W_{i,j})$ ) is true or
    for some fringe square  $[i,j]$ , ASK(KB,  $(P_{i,j} \vee W_{i,j})$ ) is false then do
      plan  $\leftarrow$  A*-GRAPH-SEARCH(ROUTE-PB([x,y], orientation, [i,j], visited))
      action  $\leftarrow$  POP(plan)
  else action  $\leftarrow$  a randomly chosen move
  return action

```

Hạn chế của Logic mệnh đề

- Trong ví dụ trên, mỗi một ô phải có công thức mà tính chất rất giống nhau.
Nguyên nhân là do Logic mệnh đề không có khả năng biểu diễn quan hệ giữa các đối tượng trong thế giới của Agent.

Một số ví dụ khác...

Chỉ có thể giải quyết được với Logic vị từ!

Đọc Thêm

1. Giáo trình chương 7.
2. *Symbolic Logic and Mechanical Theorem Proving*, C.L. Chang and R.C. Lee (chương 5).
3. *Automated Reasoning*, L. Wos, R. Overbeek, E. Lusk, and J. Boyle (ứng dụng của lập luận tự động).
4. *An Introduction to Expert Systems*, P. Jackson (chương 5).

Câu hỏi ôn tập

1. Trình bày về logic mệnh đề: cú pháp, ngữ nghĩa, mô hình, tính chân lý, tính thoả được, tính hằng sai.
2. Trình bày phương pháp chuyển công thức logic mệnh đề sang dạng CNF.
3. Cài đặt phép giải và xây dựng chương trình chứng minh tự động cho phép giải.
4. Cài đặt cơ chế lập luận tiến/lùi trên cơ sở tri thức gồm các câu dạng horn và ứng dụng để xây dựng các hệ chuyên gia.
5. Cài đặt, thí nghiệm và đánh giá DPLL, WalkSAT...
6. Nghiên cứu cài đặt các phương pháp tìm kiếm heuristics khác (SA, GA, ACO,...) cho bài toán SAT.
7. Cài đặt Agent có khả năng lập luận cho bài toán hang Wumpus.

NHẬP MÔN TRÍ TUỆ NHÂN TẠO

Chương 2: Logic hình thức

Biên soạn: TS Ngô Hữu Phúc

Bộ môn: Khoa học máy tính

Mobile: 098 56 96 580

Email: ngohuuphuc76@gmail.com

Thông tin chung

- Thông tin về nhóm môn học:

TT	Họ tên giáo viên	Học hàm	Học vị	Đơn vị công tác (Bộ môn)
1	Ngô Hữu Phúc	GVC	TS	BM Khoa học máy tính
2	Trần Nguyên Ngọc	GVC	TS	BM Khoa học máy tính
3	Hà Chí Trung	GVC	TS	BM Khoa học máy tính
4	Trần Cao Trưởng	GV	ThS	BM Khoa học máy tính

- Thời gian, địa điểm làm việc: Bộ môn Khoa học máy tính Tầng 2, nhà A1.
- Địa chỉ liên hệ: Bộ môn Khoa học máy tính, khoa Công nghệ thông tin.
- Điện thoại, email: 069-515-329, ngohuuphuc76.mta@gmail.com.

Cấu trúc môn học

- Chương 1: Giới thiệu chung.
- Chương 2: Logic hình thức.
- Chương 3: Các phương pháp tìm kiếm mù.
- Chương 4: Các phương pháp tìm kiếm có sử dụng thông tin.
- Chương 5: Các chiến lược tìm kiếm có đối thủ.
- Chương 6: Các bài toán thỏa ràng buộc.
- Chương 7: Nhập môn học máy.

Bài 2: Logic hình thức

Chương 2, mục: 2.1 – 2.4

Tiết: 1-3; 4-6; Tuần thứ: 2,3.

Mục đích, yêu cầu:

1. Nắm được Logic vị từ bậc 1.
2. Nắm được phương pháp biểu diễn tri thức trong Logic vị từ bậc 1.
3. Nắm được phương pháp lập luận và suy diễn trong Logic vị từ bậc 1.

Hình thức tổ chức dạy học: Lý thuyết.

Thời gian: 3 tiết.

Địa điểm: Giảng đường do Phòng Đào tạo phân công

Nội dung chính: (Slides)

Nội dung

- Logic vị từ bậc 1.
- Biểu diễn tri thức trong Logic vị từ bậc 1.
- Lập luận và suy diễn trong Logic vị từ bậc 1.

Ưu nhược điểm của logic mệnh đề

- ☺ logic mệnh đề mang tính đặc tả.
- ☺ logic mệnh đề cho phép biểu diễn thông tin bộ phận, kết hợp, phủ định
- ☺ Logic mệnh đề có tính cấu thành về ngữ nghĩa
 - ngữ nghĩa $B_{1,1} \wedge P_{1,2}$ dẫn được từ ngữ nghĩa $B_{1,1}$ và $P_{1,2}$
- ☺ Ngữ nghĩa của logic mệnh đề là độc lập ngữ cảnh.
 - (không giống như trong ngôn ngữ tự nhiên, ngữ nghĩa phụ thuộc ngữ cảnh)
- ☹ Năng lực biểu diễn tri thức của logic mệnh đề hạn chế

Logic Vị từ bậc 1

- Trong thế giới của Logic mệnh đề chỉ có các facts (mệnh đề bài trung),
- Thế giới của Logic vị từ:
 - đối tượng: người, nhà cửa, xe, máy tính,....
 - quan hệ: đỏ, xanh, cay, đắng, anh em, yêu,...
 - hàm: cha của, ban tốt nhất, nhiều hơn một, cộng, trừ, ...

Cú pháp của Logic vị từ

- hằng Hà nội, 2, HVKTQS,...
- vị từ Anh em, >,...
- Functions Sqrt, Điểm cao nhất của,...
- biến x, y, a, b,...
- phép toán \neg , \Rightarrow , \wedge , \vee , \Leftrightarrow
- đồng nhất =
- lượng từ \forall , \exists

Câu nguyên thuỷ

Câu nguyên thuỷ = $vịtù (term_1, \dots, term_n)$
hoặc $term_1 = term_2$

Term = $hàm (term_1, \dots, term_n)$
hoặc hằng hoặc biến

- VD: $Brother(KingJohn, RichardTheLionheart) > (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))$

Mệnh đề phức hợp

- tạo nên từ những câu (mệnh đề) nguyên thuỷ thông qua các phép toán logic:

$$\neg S, S_1 \wedge S_2, S_1 \vee S_2, S_1 \Rightarrow S_2, S_1 \Leftrightarrow S_2,$$

E.g. $\text{Sibling}(\text{KingJohn}, \text{Richard}) \Rightarrow \text{Sibling}(\text{Richard}, \text{KingJohn})$

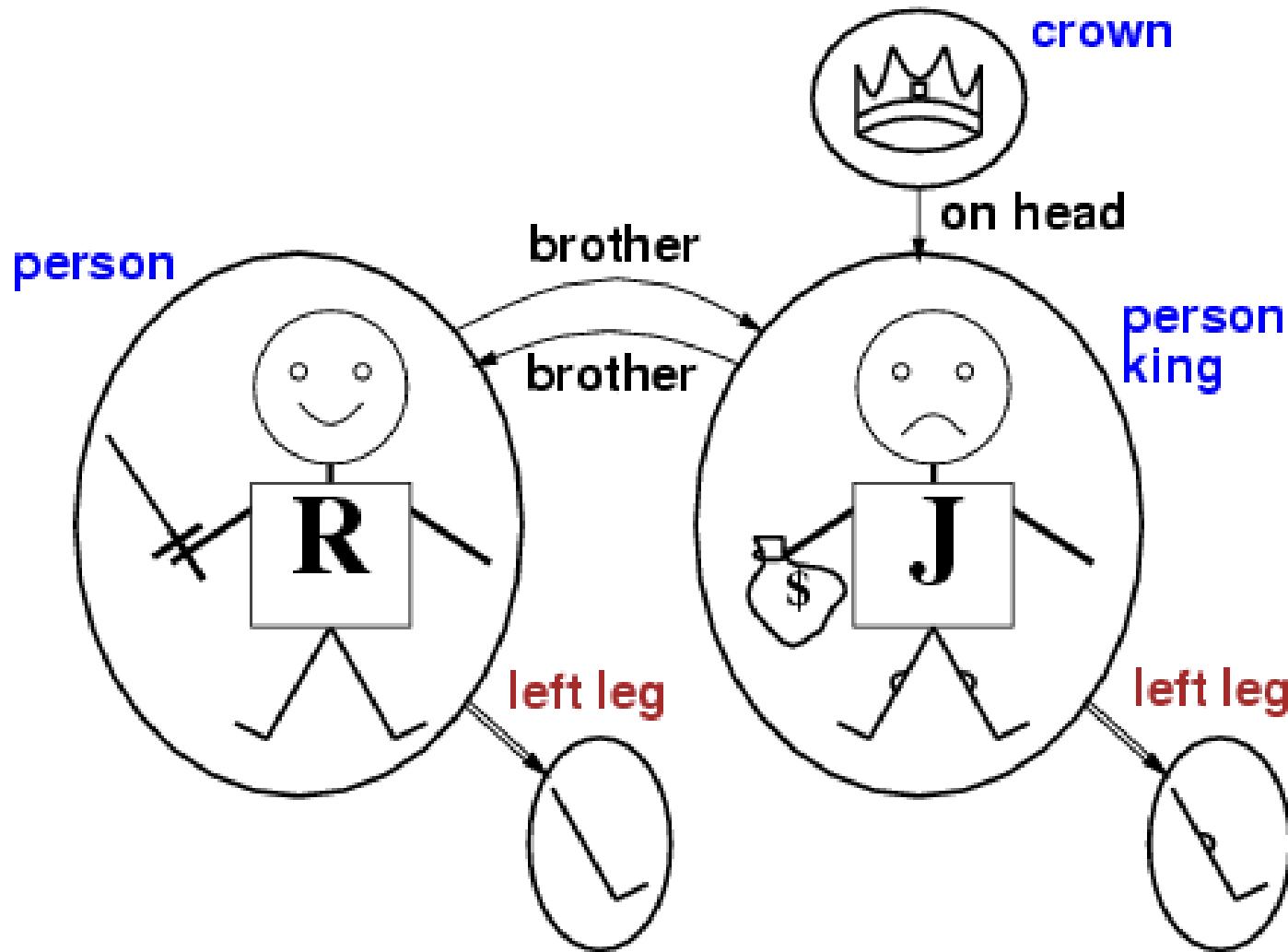
$$>(1,2) \vee \leq(1,2)$$
$$>(1,2) \wedge \neg >(1,2)$$

Luận lý trong logic vị từ

- mệnh đề được xem xét giá trị đúng/sai trên một model và một diễn dịch.
- Model gồm các đối tượng và quan hệ giữa chúng.
- Diễn dịch là ngữ nghĩa, tham chiếu, diễn giải trên model:

ký hiệu hằng	→	đối tượng
ký hiệu vị từ	→	quan hệ
ký hiệu hàm	→	quan hệ hàm
- Một câu (mệnh đề) nguyên thuỷ predicate(term₁, ..., term_n) là đúng khi và chỉ khi các đối tượng tham chiếu bởi term₁, term₂, ...term_n, có quan hệ predicate trên model.
- Ví dụ: ???

Models cho LGVT: Ví dụ



Lượng tử phổ dụng

- $\forall <biến> <mệnh đề>$

Mọi học viên tại HVKTQS đều thông minh:
 $\forall x At(x, HVKTQS) \Rightarrow Smart(x)$

- $\forall x P$ là đúng trên model m khi và chỉ khi P là đúng với x tham chiếu bằng bất cứ đối tượng nào thuộc m .
- Tương đương với việc kiểm tra mệnh đề trên từng đối tượng của m

$$\begin{aligned} & At(Nam, HVKTQS) \Rightarrow Smart(Nam) \\ \wedge \quad & At(Lan, HVKTQS) \Rightarrow Smart(Lan) \\ \wedge \quad & At(Minh, HVKTQS) \Rightarrow Smart(Minh) \\ \wedge \dots \end{aligned}$$

Lỗi thông dụng

- Thông thường, \Rightarrow là phép toán chính trong phạm vi ảnh hưởng của \forall
- Ví dụ lỗi (dùng \wedge):
$$\forall x \text{At}(x, \text{HVKTQS}) \wedge \text{Smart}(x)$$
có nghĩa là “Mọi người đều là học viên HVKTQS và mọi người đều thông minh”.

Lượng tử tồn tại

- $\exists <biến> <\text{mệnh đề}>$
- Tại học viện KTQS có học viên thông minh:
- $\exists x \text{At}(x, \text{HVKTQS}) \wedge \text{Smart}(x)$
- $\exists x P$ là đúng trên model m khi và chỉ khi P là đúng với x được tham chiếu bởi một (vài) đối tượng trong m .
- Tương đương với:
 - At(Nam, HVKTQS) \wedge Smart(Nam)
 - ✓ At(Lan, HVKTQS) \wedge Smart(Lan)
 - ✓ At(Minh, HVKTQS) \wedge Smart(Minh)
 - ✓ ...

Lỗi thường gặp

- Thông thường, \wedge là phép toán chính gắn với \exists
- Lỗi thông dụng: Sử dụng \Rightarrow làm phép toán chính trong \exists :
$$\exists x \text{At}(x, \text{HVKTQS}) \Rightarrow \text{Smart}(x)$$
đúng nếu có người không thuộc HVKTQS!

Tính chất của lượng từ

- $\forall x \forall y$ giống như $\forall y \forall x$
- $\exists x \exists y$ giống như $\exists y \exists x$
- $\exists x \forall y$ có thể khác $\forall y \exists x$
-
- $\exists x \forall y \text{ Loves}(x,y)$
 - “Tồn tại người yêu tất cả mọi người trên thế giới”
- $\forall y \exists x \text{ Loves}(x,y)$
 - “Mọi người trên thế giới này đều có ít nhất một người yêu”
- Đổi ngẫu lượng từ:
- $\forall x \text{ Likes}(x, \text{IceCream})$ $\neg \exists x \neg \text{Likes}(x, \text{IceCream})$
- $\exists x \text{ Likes}(x, \text{Broccoli})$ $\neg \forall x \neg \text{Likes}(x, \text{Broccoli})$

Phép Đồng Nhất

- $term_1 = term_2$ là đúng trong một diễn dịch khi và chỉ khi $term_1$ và $term_2$ cùng tham chiếu đến một đối tượng
- E.g., định nghĩa of *Sibling* dùng *Parent*:
$$\forall x, y \text{ } Sibling(x, y) \Leftrightarrow [\neg(x = y) \wedge \exists m, f \neg(m = f) \wedge \text{Parent}(m, x) \wedge \text{Parent}(f, x) \wedge \text{Parent}(m, y) \wedge \text{Parent}(f, y)]$$

Dùng LGVT cho biểu diễn tri thức

Biểu diễn quan hệ huyết thống: (tiếng anh)

- Brothers are sibling
 $\forall x,y \text{ } Brother(x,y) \Leftrightarrow Sibling(x,y)$
- One's mother is one's female parent
 $\forall m,c \text{ } Mother(c) = m \Leftrightarrow (\text{Female}(m) \wedge \text{Parent}(m,c))$
- “Sibling” is symmetric
 $\forall x,y \text{ } Sibling(x,y) \Leftrightarrow Sibling(y,x)$

Dùng LGVT cho biểu diễn tri thức

Tri thức về tập hợp

- $\forall s \text{ Set}(s) \Leftrightarrow (s = \{\}) \vee (\exists x, s_2 \text{ Set}(s_2) \wedge s = \{x|s_2\})$
- $\neg \exists x, s \{x|s\} = \{\}$
- $\forall x, s x \in s \Leftrightarrow s = \{x|s\}$
- $\forall x, s x \in s \Leftrightarrow [\exists y, s_2] (s = \{y|s_2\} \wedge (x = y \vee x \in s_2))$
- $\forall s_1, s_2 s_1 \subseteq s_2 \Leftrightarrow (\forall x x \in s_1 \Rightarrow x \in s_2)$
- $\forall s_1, s_2 (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$
- $\forall x, s_1, s_2 x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)$
- $\forall x, s_1, s_2 x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)$

Cơ sở tri thức trên Logic vị từ cho bài toán hang Wumpus

- Quan sát
 - $\forall t, s, b \text{ Percept}([s, b, \text{Ánh kim}], t) \Rightarrow \text{Có vàng } (t)$
- Hành động:
 - $\forall \text{ Ánh kim}(t) \Rightarrow \text{BestAction(Chộp}, t)$

Biểu diễn các tri thức ẩn

- $\forall x,y,a,b \text{ Liền_kè } ([x,y],[a,b]) \Leftrightarrow [a,b] \in \{[x+1,y], [x-1,y],[x,y+1],[x,y-1]\}$

Ô có tiếng gió nếu gần ô có hầm chông:

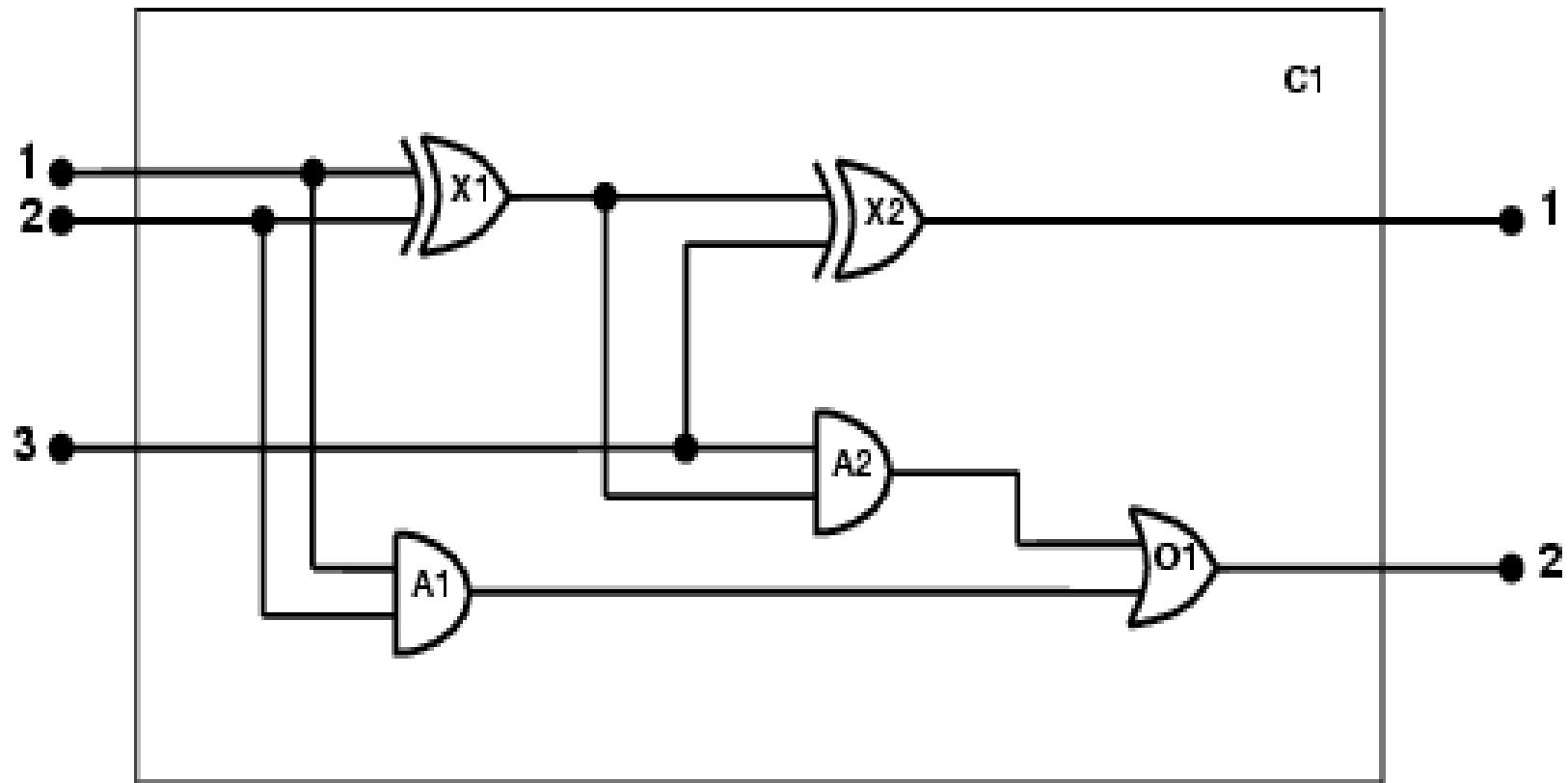
$$\forall s \text{ Breezy}(s) \Rightarrow \exists r \text{ Liền_Kè }(r,s) \wedge \text{Hầm_chông }(r)$$
$$\forall r \text{ Hầm_chông }(r) \Rightarrow [\forall s \text{ Liền_kè }(r,s) \Rightarrow \text{Có_tiếng_gió }(s)]$$

Biểu diễn tri thức trong LGVT

- 1. Xác định bài toán.**
- 2. Thu thập tri thức liên quan.**
- 3. Xác định vị từ, hàm, hằng**
- 4. Biểu diễn tri thức chung về miền giá trị của bài toán**
- 5. Biểu diễn tri thức cụ thể liên quan đến bài toán**
- 6. Đặt câu hỏi để máy suy diễn tự lập luận và trả lời**
- 7. Debug cơ sở tri thức.**

Ví dụ về mạch điện tử

Bộ cộng 1 bit (có nhớ):



Ví dụ về mạch điện tử

1. Xác định bài toán

- Mạch có công chính xác? (Kiểm tra mạch)

2. Thu thập tri thức liên quan

- Mạch gồm đường nối với các cổng logic; loại cổng (AND, OR, XOR, NOT)
- không liên quan: kích thước, hình dạng, màu sắc, chi phí của các loại cổng.

3. Xác định từ vựng (vị trí, hàm, hằng):

Type(X_1) = XOR

Type(X_1 , XOR)

XOR(X_1)

Ví dụ về mạch điện tử

4. Biểu diễn tri thức chung của miền giá trị bài toán

- $\forall t_1, t_2 \text{ Connected}(t_1, t_2) \Rightarrow \text{Signal}(t_1) = \text{Signal}(t_2)$
- $\forall t \text{ Signal}(t) = 1 \vee \text{Signal}(t) = 0$
- $1 \neq 0$
- $\forall t_1, t_2 \text{ Connected}(t_1, t_2) \Rightarrow \text{Connected}(t_2, t_1)$
- $\forall g \text{ Type}(g) = \text{OR} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 1 \Leftrightarrow \exists n \text{ Signal}(\text{In}(n,g)) = 1$
- $\forall g \text{ Type}(g) = \text{AND} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 0 \Leftrightarrow \exists n \text{ Signal}(\text{In}(n,g)) = 0$
- $\forall g \text{ Type}(g) = \text{XOR} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 1 \Leftrightarrow \text{Signal}(\text{In}(1,g)) \neq \text{Signal}(\text{In}(2,g))$
- $\forall g \text{ Type}(g) = \text{NOT} \Rightarrow \text{Signal}(\text{Out}(1,g)) \neq \text{Signal}(\text{In}(1,g))$

Ví dụ về mạch điện tử

5. Biểu diễn tri thức liên quan trực tiếp đến bài toán.

Type(X_1) = XOR

Type(X_2) = XOR

Type(A_1) = AND

Type(A_2) = AND

Type(O_1) = OR

Connected(Out(1, X_1),In(1, X_2))

Connected(In(1, C_1),In(1, X_1))

Connected(Out(1, X_1),In(2, A_2))

Connected(In(1, C_1),In(1, A_1))

Connected(Out(1, A_2),In(1, O_1))

Connected(In(2, C_1),In(2, X_1))

Connected(Out(1, A_1),In(2, O_1))

Connected(In(2, C_1),In(2, A_1))

Connected(Out(1, X_2),Out(1, C_1))

Connected(In(3, C_1),In(2, X_2))

Connected(Out(1, O_1),Out(2, C_1))

Connected(In(3, C_1),In(1, A_2))

Ví dụ về mạch điện tử

6. Đặt câu hỏi cho máy suy diễn:

Tập các giá trị đầu ra có thể với tập các giá trị vào có thể của mạch?

$$\exists i_1, i_2, i_3, o_1, o_2 \text{ Signal}(In(1, C_1)) = i_1 \wedge \text{Signal}(In(2, C_1)) = i_2 \wedge \\ \text{Signal}(In(3, C_1)) = i_3 \wedge \text{Signal}(Out(1, C_1)) = o_1 \wedge \\ \text{Signal}(Out(2, C_1)) = o_2$$

7. Debug cơ sở tri thức

Có thể quên biểu diễn $1 \neq 0$

Tóm tắt

- Logic vị từ (bậc 1):
 - Đối tượng ngữ nghĩa nguyên thuỷ là đối tượng và quan hệ.
 - Cú pháp: hằng, hàm, vị từ, đồng nhất, lượng tử.
- Khả năng diễn đạt mạnh hơn logic mệnh đề:
 - đủ để biểu diễn tri thức cho bài toán hang Wumpus.

Lập Luận Trên LGVT

- Đưa lập luận trên LGVT về lập luận trên logic mệnh đề.
- Phép hợp nhất (Unification)
- Lập luận tam đoạn luận tổng quát.
- Lập luận hướng tiến.
- Lập luận hướng lùi.
- Phép giải.

Khởi trị cho phép lượng từ phổ dụng (UI)

- Thế các biến bằng tất cả các hằng (đối tượng) trên model:

$$\forall v \alpha$$
$$\text{Subst}(\{v/g\}, \alpha)$$

với mọi biến v term g .

- E.g., $\forall x \text{King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$:

$$\text{King}(\text{John}) \wedge \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John})$$
$$\text{King}(\text{Richard}) \wedge \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard})$$
$$\text{King}(\text{Father}(\text{John})) \wedge \text{Greedy}(\text{Father}(\text{John})) \Rightarrow$$
$$\text{Evil}(\text{Father}(\text{John}))$$

.

Khởi trị lượng tử tồn tại

- Với mọi mệnh đề α , biến v , và ký hiệu hằng k không xuất hiện trong KB:

$$\frac{\exists v \alpha}{\text{Subst}(\{v/k\}, \alpha)}$$

- E.g., $\exists x \ Crown(x) \wedge \text{OnHead}(x, John)$ yields:

$$Crown(C_1) \wedge \text{OnHead}(C_1, John)$$

Với C_1 là một hằng, gọi là hằng Skolem

Đưa về lập luận trên Logic mệnh đề

Giải sử KB như sau:

$$\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$$

King(John)

Greedy(John)

Brother(Richard, John)

- Khởi trị cho lượng tử phổ dụng ta có:

King(John) \wedge Greedy(John) \Rightarrow Evil(John)

King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)

King(John)

Greedy(John)

Brother(Richard, John)

- Cơ sở tri thực đã được mệnh đề hóa:

King(John), Greedy(John), Evil(John), King(Richard), ...

Đưa về lập luận trên Logic mệnh đề

- Mọi cơ sở tt biểu diễn bằng logic vị từ đều có thể mệnh đề hoá
- ý tưởng: mệnh đề hoá cơ sở tri thức và câu hỏi, áp dụng phép giải, thu kết quả.
- Hạn chế: Với ký hiệu hàm, có thể có vô hạn term tương ứng
 - e.g., *Father(Father(Father(John)))*

Đưa về lập luận trên Logic mệnh đề

Định lý: Herbrand (1930). mệnh đề α là dẫn được từ cơ sở tri thức LGVT KB, nếu nó dẫn được từ một tập con hữu hạn của cơ sở tri thức mệnh đề hoá từ KB.

Do đó: For $n = 0$ to ∞ do

Tạo cơ sở tri thức mệnh đề hoá từ KB khởi trị với các term có độ sâu n .

Kiểm tra xem α có dẫn được từ cơ sở tri thức này không? (giống như làm trong logic mệnh đề)

Hạn chế: sẽ kết thúc nếu α dẫn được, lặp vô tận nếu α không dẫn được.

Định lý: Turing (1936), Church (1936) Bài toán dẫn được trong cơ sở tri thức dùng LGVT là nửa quyết định được (có thuật toán tồn tại để trả lời YES cho những mệnh đề dẫn được, nhưng không tồn tại thuật toán trả lời NO cho những mệnh đề không dẫn được.)

Hạn chế của việc mệnh đề hoá

- Mệnh đề hoá sinh ra rất nhiều những mệnh đề không liên quan đến quá trình suy diễn.
- Ví dụ:
 $\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$
 $\text{King}(\text{John})$
 $\forall y \text{ Greedy}(y)$
 $\text{Brother}(\text{Richard}, \text{John})$
- Dễ thấy KB dẫn ra *Evil(John)*, nhưng quá trình mệnh đề hoá dẫn tới nhiều mệnh đề chẳng liên quan như *Greedy(Richard)*....
- với p k-ary vị từ và n hằng số, có $p \cdot n^k$ mệnh đề tương ứng.

Phép hợp nhất

- Có thê có phép suy luận đơn giản nếu tìm được phép thê θ sao cho $King(x)$ và $Greedy(x)$ khớp với $King(John)$ và $Greedy(y)$

$$\theta = \{x/John, y/John\}$$

- $\text{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

p	q	θ
Knows(John,x)	Knows(John,Jane)	
Knows(John,x)	Knows(y,OJ)	
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

Phép hợp nhất

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}}
Knows(John,x)	Knows(y,OJ)	
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

Phép hợp nhất

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}}
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

Phép hợp nhất

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}
Knows(John,x)	Knows(x,OJ)	

Phép hợp nhất

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)})}
Knows(John,x)	Knows(x,OJ)	{fail}

Phép hợp nhất

- hợp nhất $Knows(John,x)$ và $Knows(y,z)$ cần có phép thế:
 $\theta = \{y/John, x/z\}$ or $\theta = \{y/John, x/John, z/John\}$
- Phép hợp nhất thứ nhất tổng quát hơn thứ hai.
- Chỉ có một phép hợp nhất tổng quát nhất (MGU) (chỉ sai khác tên biến).
 $MGU = \{y/John, x/z\}$

Thuật toán hợp nhất

```
function UNIFY( $x, y, \theta$ ) returns a substitution to make  $x$  and  $y$  identical  
inputs:  $x$ , a variable, constant, list, or compound  
         $y$ , a variable, constant, list, or compound  
         $\theta$ , the substitution built up so far  
  
if  $\theta$  = failure then return failure  
else if  $x = y$  then return  $\theta$   
else if VARIABLE?( $x$ ) then return UNIFY-VAR( $x, y, \theta$ )  
else if VARIABLE?( $y$ ) then return UNIFY-VAR( $y, x, \theta$ )  
else if COMPOUND?( $x$ ) and COMPOUND?( $y$ ) then  
    return UNIFY(ARGS[ $x$ ], ARGS[ $y$ ], UNIFY(OP[ $x$ ], OP[ $y$ ],  $\theta$ ))  
else if LIST?( $x$ ) and LIST?( $y$ ) then  
    return UNIFY(REST[ $x$ ], REST[ $y$ ], UNIFY(FIRST[ $x$ ], FIRST[ $y$ ],  $\theta$ ))  
else return failure
```

Thuật toán hợp nhất

function UNIFY-VAR(var, x, θ) returns a substitution

inputs: var , a variable

x , any expression

θ , the substitution built up so far

if $\{var/val\} \in \theta$ then return UNIFY(val, x, θ)

else if $\{x/val\} \in \theta$ then return UNIFY(var, val, θ)

else if OCCUR-CHECK?(var, x) then return failure

else return add $\{var/x\}$ to θ

Tam đoạn luận tổng quát (GMP)

$$p_1', p_2', \dots, p_n', (p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow q)$$
$$q\theta$$

where $p_i'\theta = p_i \theta$ for all i

p_1' is *King(John)* p_1 is *King(x)*

p_2' is *Greedy(y)* p_2 is *Greedy(x)*

θ is {x/John,y/John} q is *Evil(x)*

$q \theta$ is *Evil(John)*

- GMP sử dụng các câu có tối đa một literal dương (câu dạng Horn)
- Tất cả các biến chỉ chứa lượng tử phổ dụng.

Ví dụ

- Theo luật lệ nếu một người Mỹ bán vũ khí cho quốc gia thù địch với nước Mỹ thì bị coi là phạm tội. Nước Nono, là kẻ thù của nước Mỹ, Nono có tên lửa, và tất cả tên lửa của nước này đều mua của đại tá West, Một người Mỹ.
- Chứng minh đại tá West có tội.

Ví dụ

... một người Mỹ bán vũ khí cho quốc gia thù nghịch là có tội:

American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)

Nono ... có tên lửa, i.e., $\exists x \text{ Owns}(\text{Nono},x) \wedge \text{Missile}(x)$:

Owns(Nono,M₁) and Missile(M₁)

... Tất cả tên lửa đều do đại tá West bán:

Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)

Tên lửa là vũ khí:

Missile(x) \Rightarrow Weapon(x)

Kẻ thù của nước Mỹ được gọi là “thù nghịch”:

Enemy(x,America) \Rightarrow Hostile(x)

West là một người Mỹ ...

American(West)

Nước Nono là kẻ thù của Mỹ ...

Enemy(Nono,America)

Thuật toán lập luận hướng tiến

```
function FOL-FC-ASK( $KB, \alpha$ ) returns a substitution or false
repeat until  $new$  is empty
     $new \leftarrow \{ \}$ 
    for each sentence  $r$  in  $KB$  do
         $(p_1 \wedge \dots \wedge p_n \Rightarrow q) \leftarrow \text{STANDARDIZE-APART}(r)$ 
        for each  $\theta$  such that  $(p_1 \wedge \dots \wedge p_n)\theta = (p'_1 \wedge \dots \wedge p'_n)\theta$ 
            for some  $p'_1, \dots, p'_n$  in  $KB$ 
                 $q' \leftarrow \text{SUBST}(\theta, q)$ 
                if  $q'$  is not a renaming of a sentence already in  $KB$  or  $new$  then do
                    add  $q'$  to  $new$ 
                     $\phi \leftarrow \text{UNIFY}(q', \alpha)$ 
                    if  $\phi$  is not fail then return  $\phi$ 
    add  $new$  to  $KB$ 
return false
```

Chứng Minh Với Lập Luận Hướng Tiến

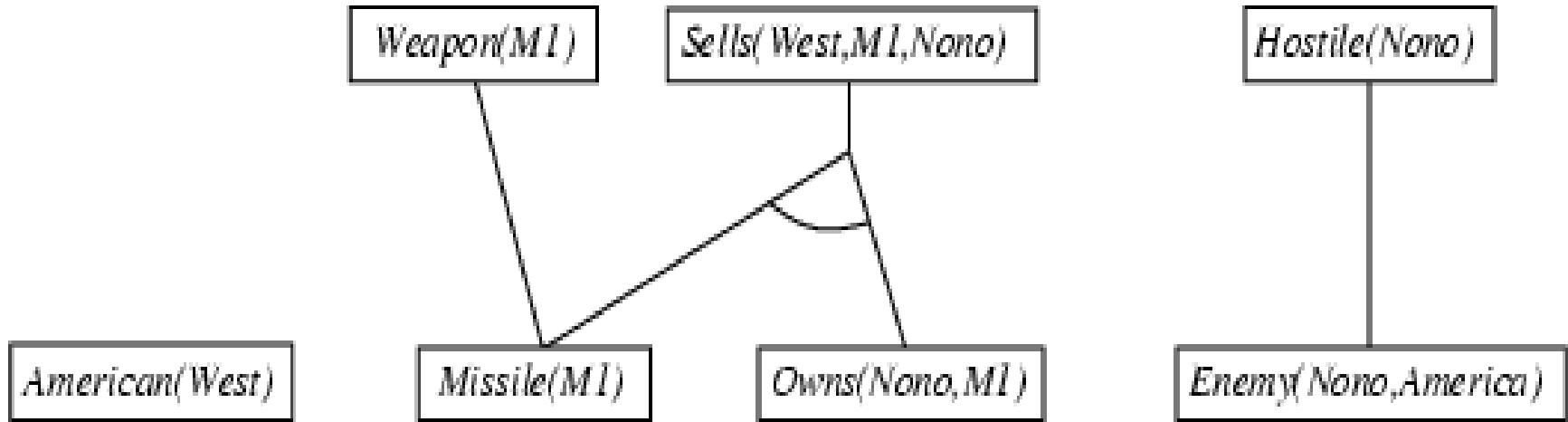
American(West)

Missile(MI)

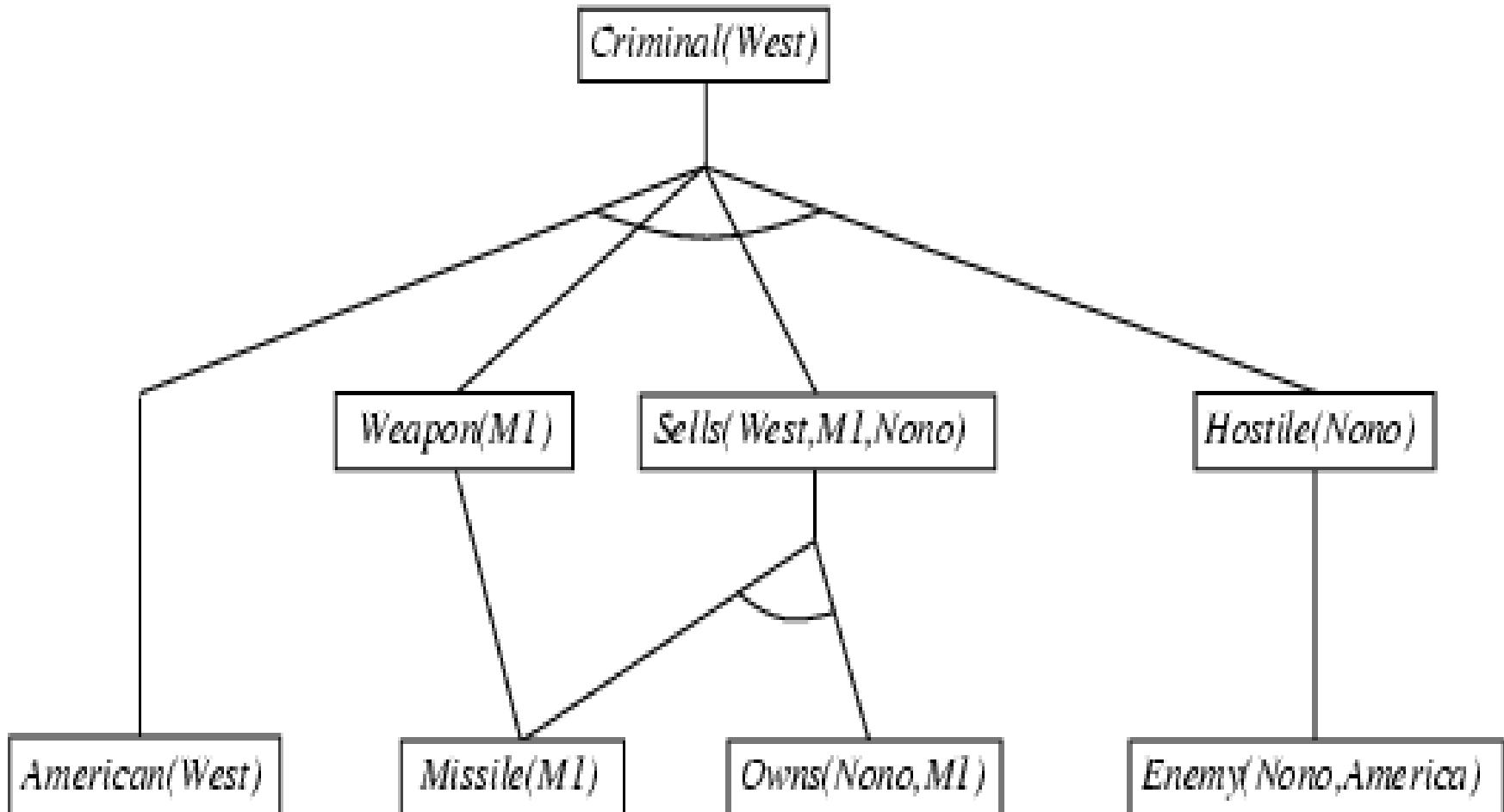
Owns(Nono,MI)

Enemy(Nono,America)

Chứng Minh Với Lập Luận Hướng Tiễn



Chứng Minh VỚI LẬP LUẬN HƯỚNG TIẾN



Đặc điểm của lập luận hướng tiến

- Chặt và đủ đối với các KB dùng logic vị từ bậc 1.
- Datalog = Các câu Horn bậc 1 + không dùng hàm
- FC kết thúc với Datalog sau một số hữu hạn bước lặp.
- Có thể không dùng nếu α không dẫn được.
- Thường dùng trong Deductive Database
- Nhắc lại: Bài toán suy dẫn trên câu dạng Horn là nửa quyết định được.

Thuật Toán Lập Luận Lùi

function FOL-BC-ASK(KB , $goals$, θ) returns a set of substitutions

inputs: KB , a knowledge base

$goals$, a list of conjuncts forming a query

θ , the current substitution, initially the empty substitution $\{ \}$

local variables: ans , a set of substitutions, initially empty

if $goals$ is empty then return $\{\theta\}$

$q' \leftarrow \text{SUBST}(\theta, \text{FIRST}(goals))$

for each r in KB where $\text{STANDARDIZE-APART}(r) = (p_1 \wedge \dots \wedge p_n \Rightarrow q)$

and $\theta' \leftarrow \text{UNIFY}(q, q')$ succeeds

$ans \leftarrow \text{FOL-BC-ASK}(KB, [p_1, \dots, p_n | REST(goals)], \text{COMPOSE}(\theta, \theta')) \cup ans$

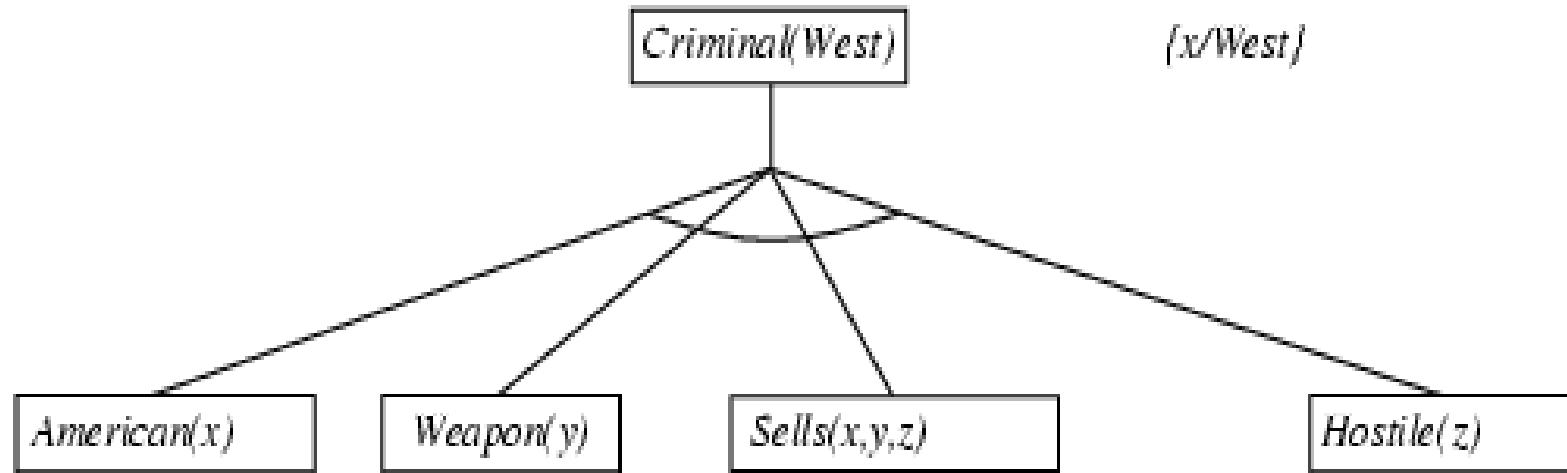
return ans

$$\text{SUBST}(\text{COMPOSE}(\theta_1, \theta_2), p) = \text{SUBST}(\theta_2, \text{SUBST}(\theta_1, p))$$

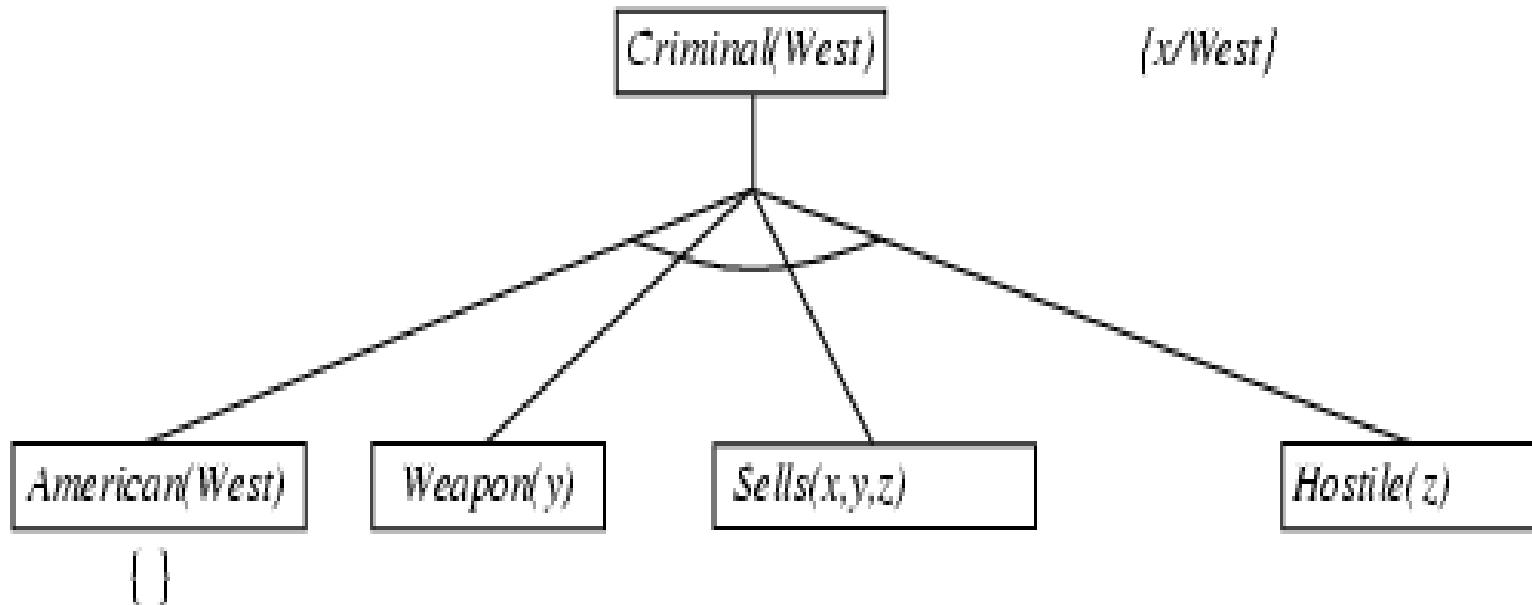
Ví dụ

Criminal (West)

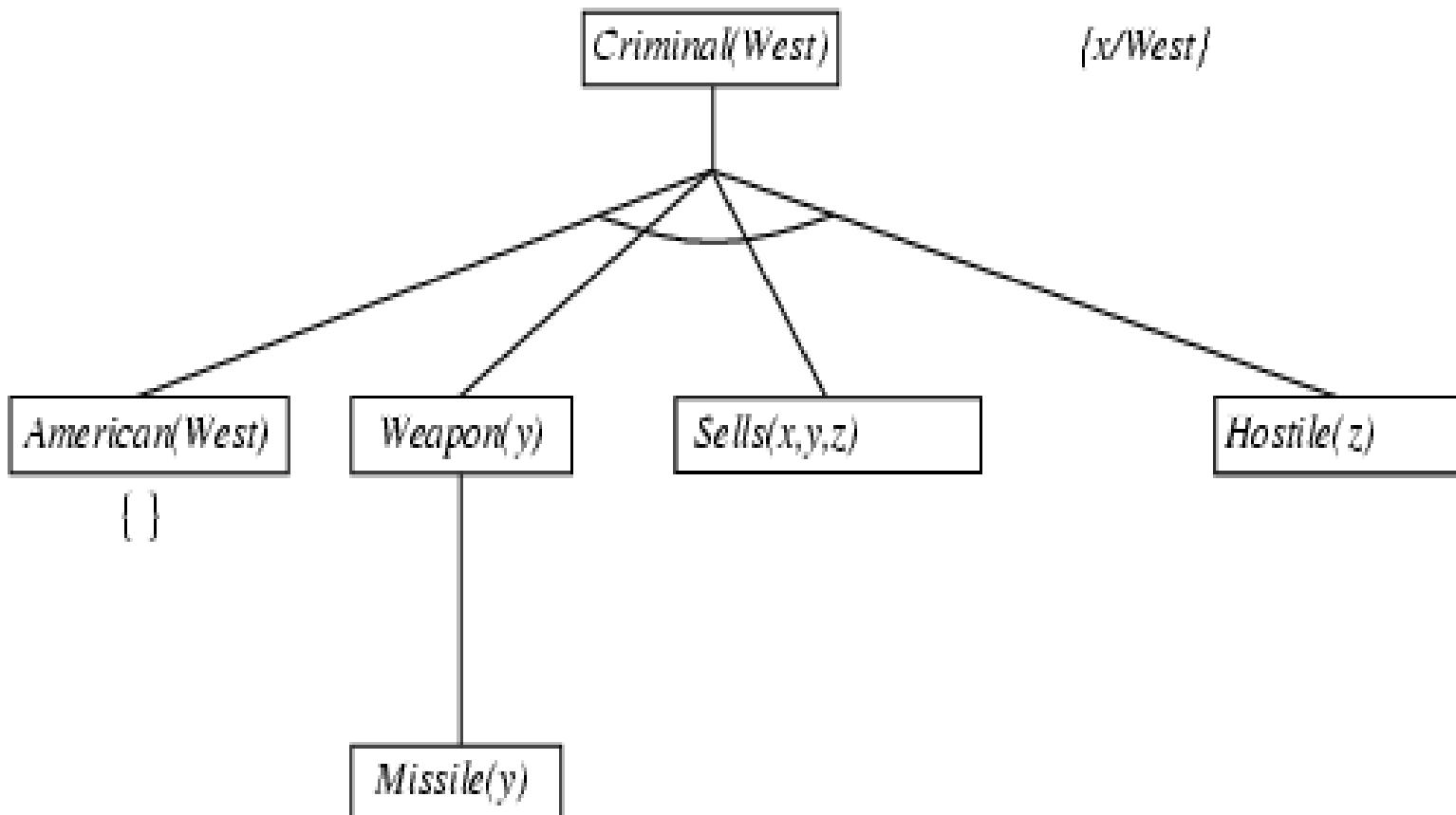
Ví dụ



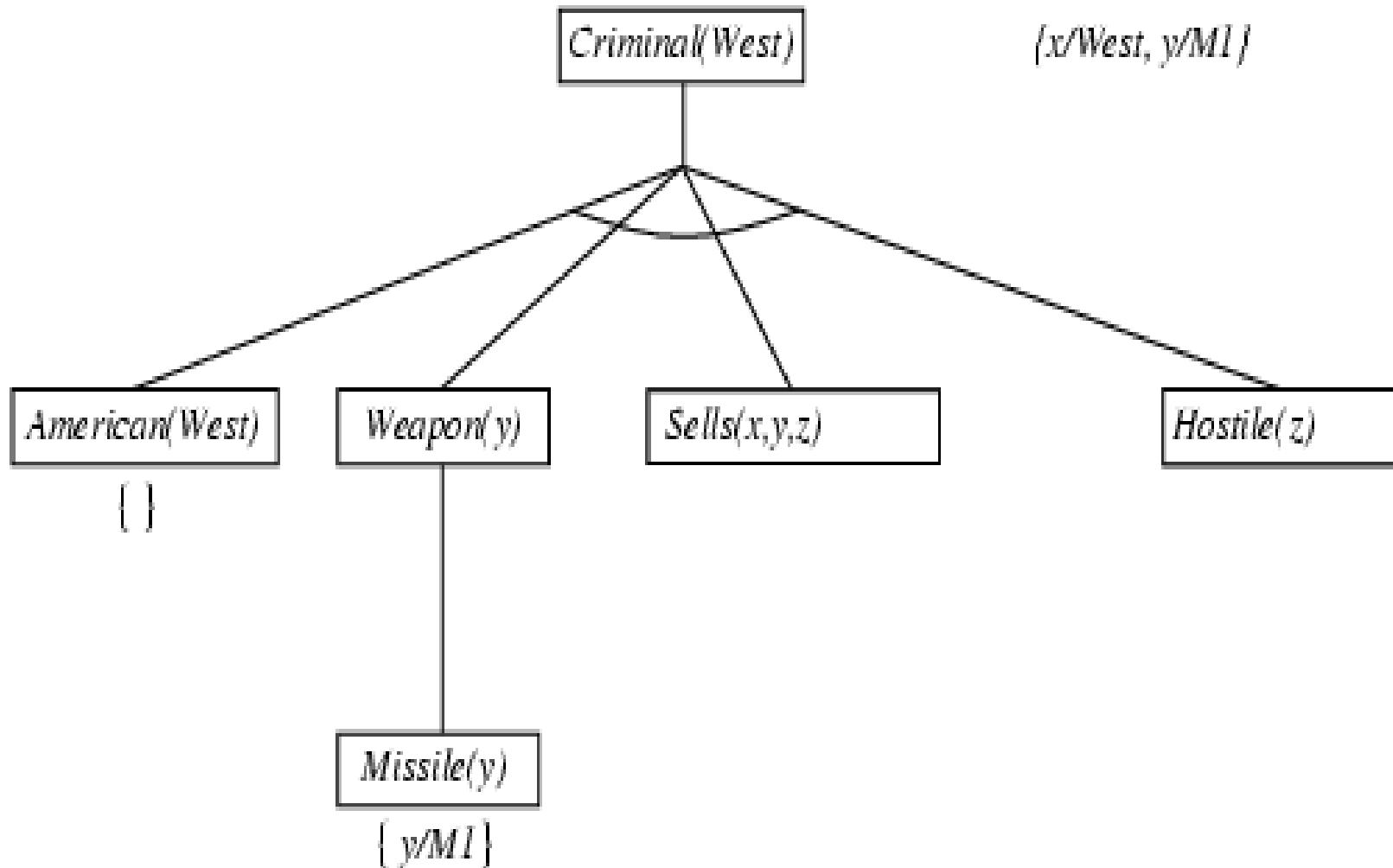
Ví dụ



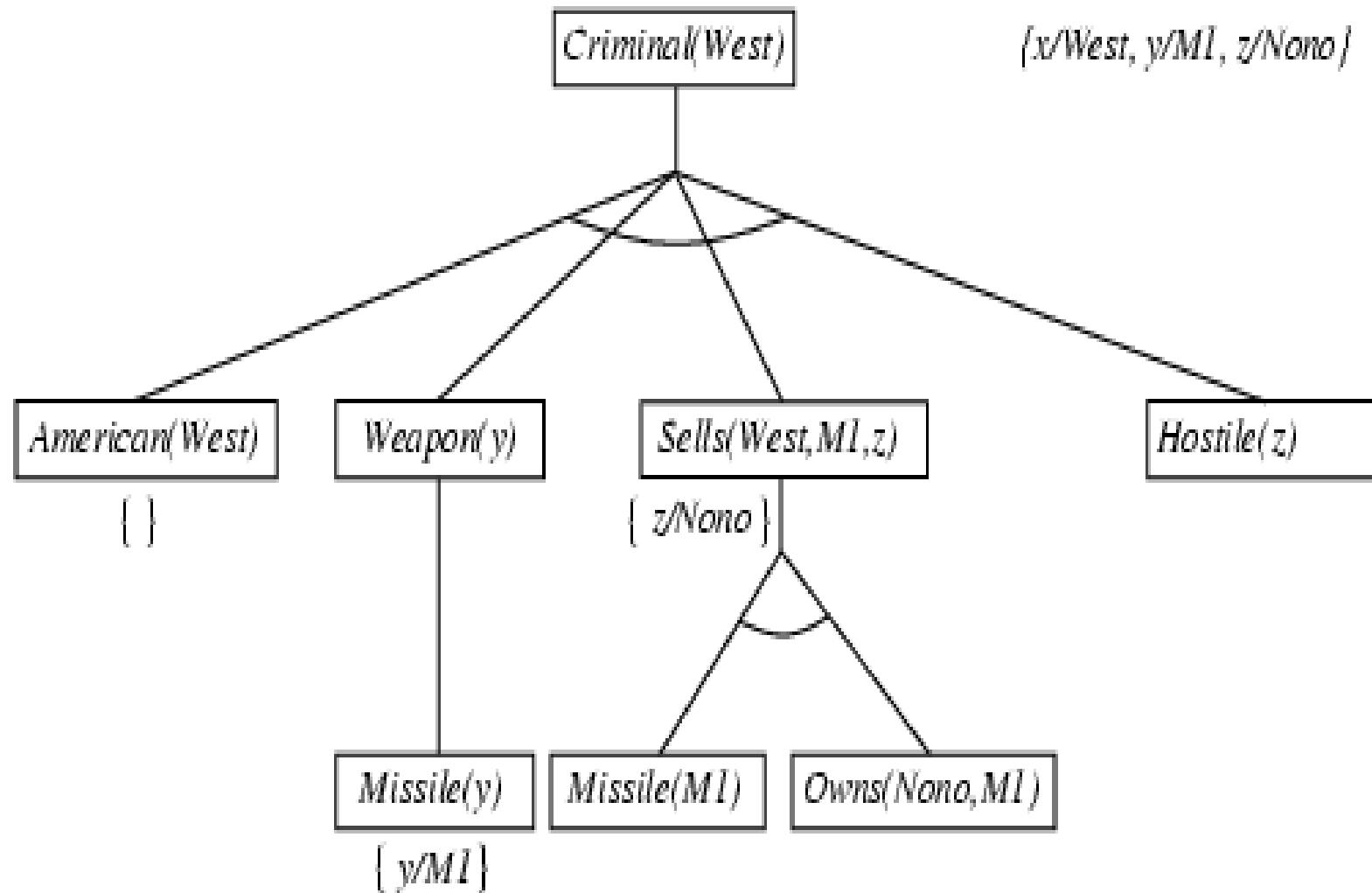
Ví dụ



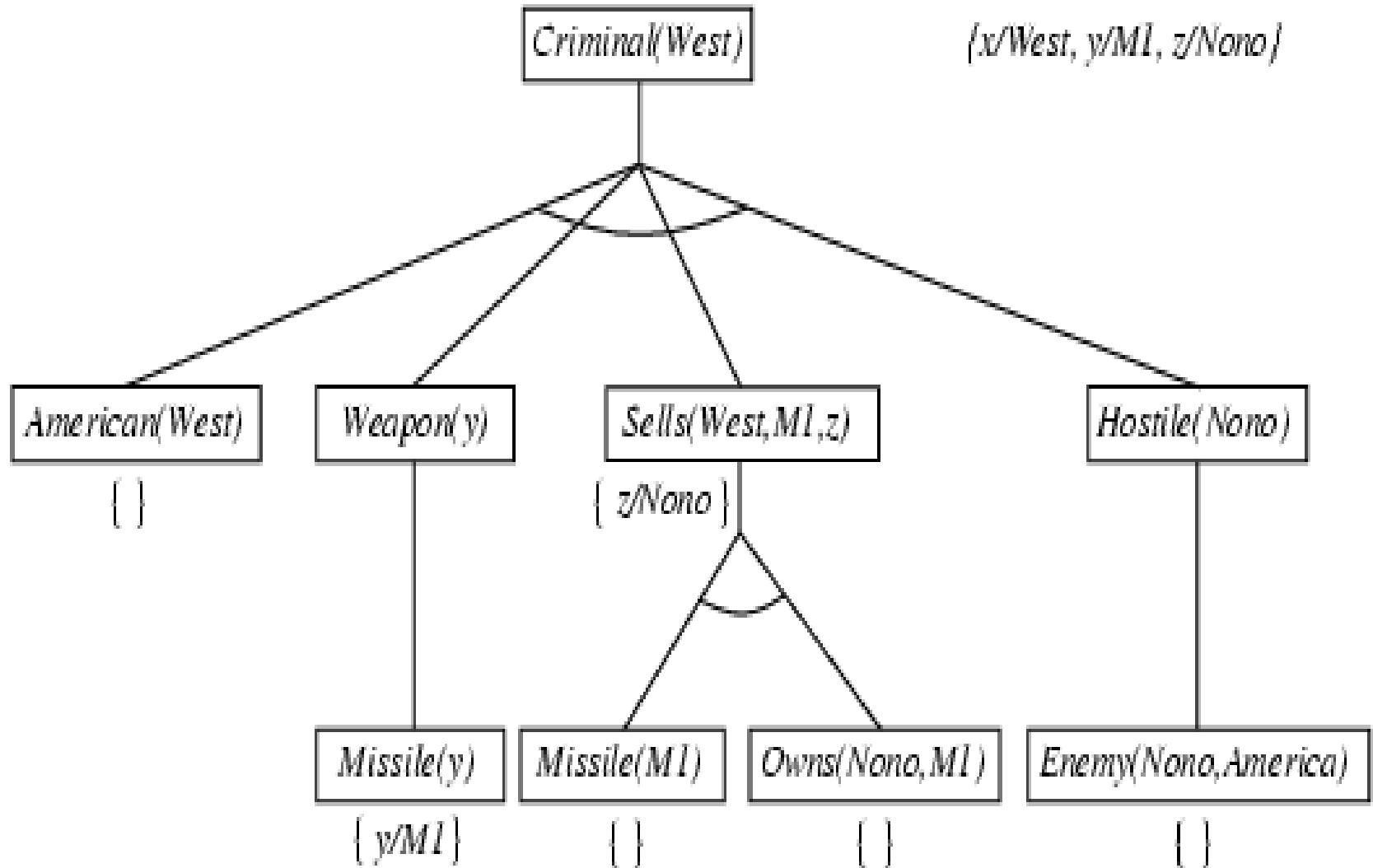
Ví dụ



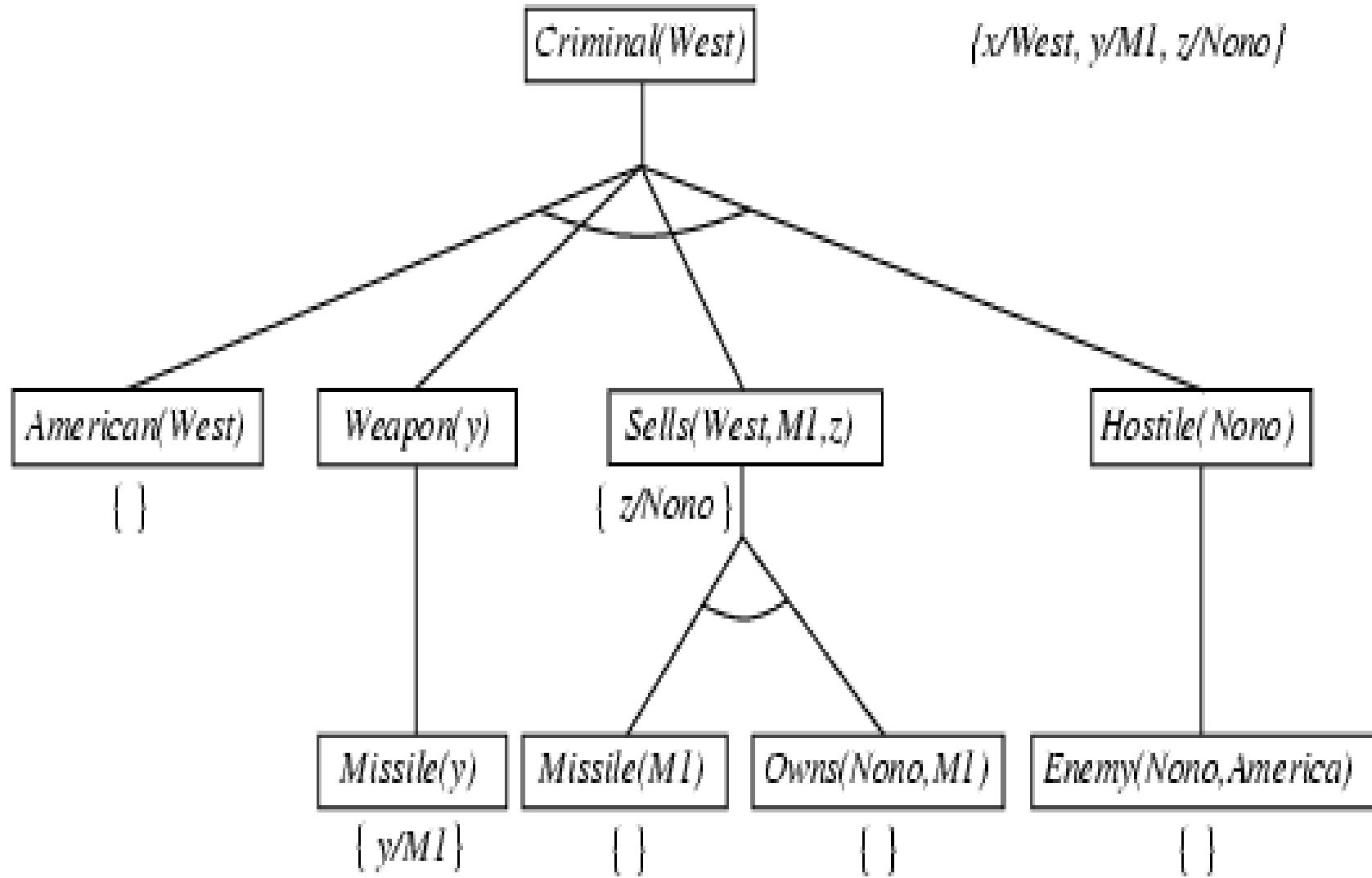
Ví dụ



Ví dụ



Ví dụ



Đặc điểm của lập luận lùi

- Tìm kiếm đệ quy theo chiều sâu. Không gian nhớ tuyến tính với độ dài chứng minh.
- Không đầy đủ do có thể rơi vào vòng lặp vô tận
- Không hiệu quả do có thể có các subgoals lặp đi lặp lại .
- Dùng trong lập trình Logic ([logic programming](#))

Lập Trình Logic: Prolog

- Chương trình = Logic + Control
- Cơ bản: Dùng lập luận lùi với các câu dạng Horn
- Dùng tại Châu âu, Nhật bản (Ngôn ngữ máy cho thế hệ thứ 5)
- Program = set of clauses = head :- literal₁, ... literal_n.
criminal(X) :- american(X), weapon(Y), sells(X, Y, Z), hostile(Z).
- Lập luận lùi với chiến lược tìm kiếm Depth-first, left-to-right
- Các vị từ có sẵn etc., e.g., X is Y*Z+3
- Các vị từ có sẵn tạo hiệu ứng lè (input and output predicates, assert/retract predicates)
- Giả thiết đóng (Phủ định được gán bằng thất bại - "negation as failure").
 - e.g., alive(X) :- not dead(X).
 - alive(joe) thành công nếu dead(joe) thất bại (không chứng minh được)

Prolog

- Ghép nối hai danh sách:

```
append( [ ] , Y , Y ) .
```

```
append( [ X | L ] , Y , [ X | Z ] ) :-
```

```
append( L , Y , Z ) .
```

- query: append(A, B, [1, 2]) ?

- answers: A= [] B= [1 , 2]

- A= [1] B= [2]

- A= [1 , 2] B= []

Phép Giải: Tóm tắt

- Trên LGVT bậc 1:

$$\frac{\ell_1 \vee \dots \vee \ell_k, \quad m_1 \vee \dots \vee m_n}{(\ell_1 \vee \dots \vee \ell_{i-1} \vee \ell_{i+1} \vee \dots \vee \ell_k \vee m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n)\theta}$$

trong đó $\text{Unify}(\ell_i, \neg m_j) = \theta$.

- Hai câu độc lập được đổi tên biến sao cho chúng không chung biến (tránh gây nhập nhằng)
- VD:

$$\frac{\neg Rich(x) \vee Unhappy(x)}{\frac{Rich(Ken)}{Unhappy(Ken)}}$$

với $\theta = \{x/Ken\}$

- Dùng phép giải trên công thức dạng CNF($KB \wedge \neg \alpha$);
- Đầy đủ đối với Logic vị từ.

Đổi sang dạng CNF

- Tất cả những ai yêu động vật thì được người khác yêu:

$$\forall x [\forall y Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y Loves(y,x)]$$

- 1. Khử dấu tương đương và kéo theo:

$$\forall x [\neg \forall y \neg Animal(y) \vee Loves(x,y)] \vee [\exists y Loves(y,x)]$$

- 2. Chuyển \neg vào trong: $\neg \forall x p \equiv \exists x \neg p$, $\neg \exists x p \equiv \forall x \neg p$

$$\forall x [\exists y \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y Loves(y,x)]$$

$$\forall x [\exists y \neg \neg Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y Loves(y,x)]$$

$$\forall x [\exists y Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y Loves(y,x)]$$

Đổi sang dạng CNF

3. Chuẩn hoá biến sao cho mỗi lượng từ gắn với 1 biến:

$$\forall x [\exists y Animal(y) \wedge \neg Loves(x,y)] \vee [\exists z Loves(z,x)]$$

4. Skolem hoá để khử lượng từ tồn tại
mỗi biến thuộc lượng từ tt được thay bằng một hàm
Skolem function của các biến gắn lượng từ phổ dụng
khác:

$$\forall x [Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee Loves(G(x),x)$$

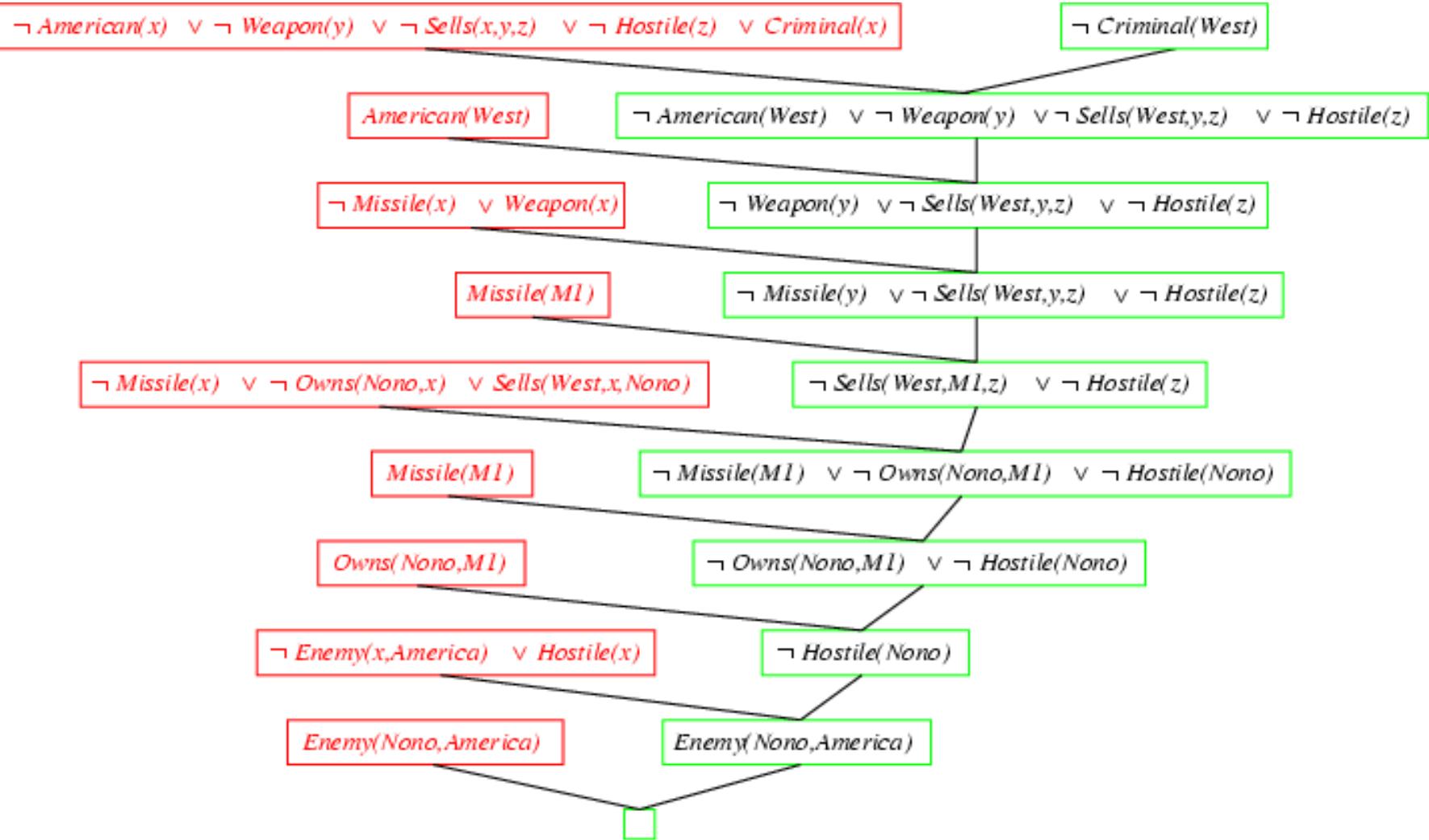
5. Xoá bỏ lượng từ phổ dụng:

$$[Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee Loves(G(x),x)$$

6. Áp dụng luật phân phối với \vee và \wedge :

$$[Animal(F(x)) \vee Loves(G(x),x)] \wedge [\neg Loves(x,F(x)) \vee$$

Chứng minh dựa vào phép giải (Proof Tree)



Đọc thêm

- **Sách giáo trình:**
 - Chương 8,9.
- **Open Courseware:**
 - Ch9, ch10, ch11
- **Logic toán:**
 - *A Mathematical Introduction to Logic*, Hebert B. Ederton;
 - *Introduction to Mathematical Logic*, Mendenson;
 - *Mathematical Logic*, Y.L. Ershov và E.A. Palyutin.
- **Phép giải và lập luận tự động:**
 - *Symbolic Logic and Mechanical Theorem Proving*, C.L. Chang and C.T. Lee.
 - *Automated Reasoning*, L. Wos et al.
- **LOGIC PROGRAMMING và PROLOG:**
 - *Foundations of Logic Programming*, J. W. Lloyd
 - *Logic, Programming and PROLOG*, U. Nilsson.
 - *PROLOG: Programming for Artificial Intelligence*, I. Bratko

Ôn tập

1. Nêu ưu/nhược điểm của logic mệnh đề.
2. Nêu cú pháp, ngũ nghĩa của Logic vị từ.
3. Nêu phương pháp mệnh đề hoá LGVT.
4. Định nghĩa phép hợp nhất và cài đặt thuật toán tìm phép hợp nhất khái quát nhất trong LGVT.
5. Cài đặt Lập luận tiến/lùi trên Logic vị từ.
6. Đưa công thức logic vị từ về dạng CNF.
7. Cài đặt phép giải trên Logic vị từ.
8. Xây dựng cơ sở tri thức cho bài toán hang wumpus dùng logic vị từ cài đặt chương trình.

Trí tuệ nhân tạo

Chương 3

Không gian trạng thái và Các phương pháp tìm kiếm mù

Biên soạn: TS Ngô Hữu Phúc

Bộ môn Khoa học máy tính

ĐT: 098 56 96 580

eMail: ngoihuuphuc76@gmail.com

Thông tin chung

- Thông tin về nhóm môn học:

TT	Họ tên giáo viên	Học hàm	Học vị	Đơn vị công tác (Bộ môn)
1	Ngô Hữu Phúc	GVC	TS	BM Khoa học máy tính
2	Trần Nguyên Ngọc	GVC	TS	BM Khoa học máy tính
3	Hà Chí Trung	GVC	TS	BM Khoa học máy tính
4	Trần Cao Trường	GV	ThS	BM Khoa học máy tính

- Thời gian, địa điểm làm việc: Bộ môn Khoa học máy tính Tầng 2, nhà A1.
- Địa chỉ liên hệ: Bộ môn Khoa học máy tính, khoa Công nghệ thông tin.
- Điện thoại, email: 069-515-329, ngohuuphuc76.mta@gmail.com.

Cấu trúc môn học

- Chương 1: Giới thiệu chung.
- Chương 2: Logic hình thức.
- Chương 3: Các phương pháp tìm kiếm mù.
- Chương 4: Các phương pháp tìm kiếm có sử dụng thông tin.
- Chương 5: Các chiến lược tìm kiếm có đối thủ.
- Chương 6: Các bài toán thỏa ràng buộc.
- Chương 7: Nhập môn học máy.

Bài 3: Tìm kiếm mù

Chương 3, mục: 3.1 – 3.6

Tiết: 1-3; Tuần thứ: 4.

Mục đích, yêu cầu:

1. Nắm được phương pháp giải quyết vấn đề.
2. Nắm được các khái niệm về không gian trạng thái.
3. Nắm được các phương pháp tìm kiếm yếu; qua đó nắm được ưu, nhược điểm của các phương pháp trên.

Hình thức tổ chức dạy học: Lý thuyết.

Thời gian: 3 tiết.

Địa điểm: Giảng đường do Phòng Đào tạo phân công

Nội dung chính: (Slides)

Nội dung bài học

1. Khái niệm về “Giải quyết một số vấn đề”.
2. Không gian trạng thái.
3. Phân loại vấn đề.
4. Các chiến lược tìm kiếm trên không gian trạng thái:
 - Tìm kiếm **hướng từ dữ liệu** (data – driven)
 - Tìm kiếm **hướng từ mục tiêu** (goal – driven).
5. Tìm kiếm trên không gian trạng thái:
 - **Tìm kiếm rộng** (breath – first search).
 - **Tìm kiếm sâu** (depth – first search).
 - **Tìm kiếm sâu bằng cách đào sâu nhiều lần** (depth – first search with iterative deepening).
6. Đồ thị and/or.

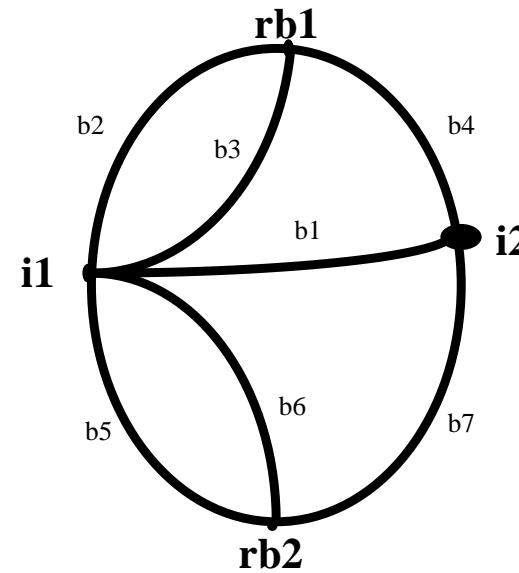
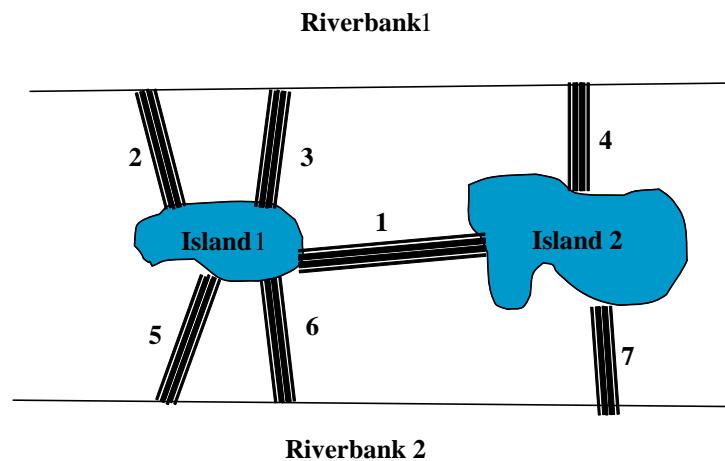
1. Khái niệm về “Giải quyết một số vấn đề”

Giải quyết vấn đề là gì?

- Để giải quyết vấn đề:
 1. Phát biểu chính xác bài toán
 - *Hiện trạng ban đầu,*
 - *Kết quả mong muốn,..*
 2. Phân tích bài toán.
 3. Thu thập và biểu diễn dữ liệu, tri thức cần thiết để giải bài toán.
 4. Lựa chọn kỹ thuật giải quyết thích hợp.

2. Không gian trạng thái - Mở đầu

- Khi biểu diễn một vấn đề như là một đồ thị không gian trạng thái, chúng ta có thể sử dụng lý thuyết đồ thị để phân tích cấu trúc và độ phức tạp của các vấn đề cũng như các thủ tục tìm kiếm.



Hệ thống cầu thành phố Konigsberg và biểu diễn đồ thị tương ứng (Leonhard Euler)

2. Không gian trạng thái

Khái niệm về Không gian trạng thái

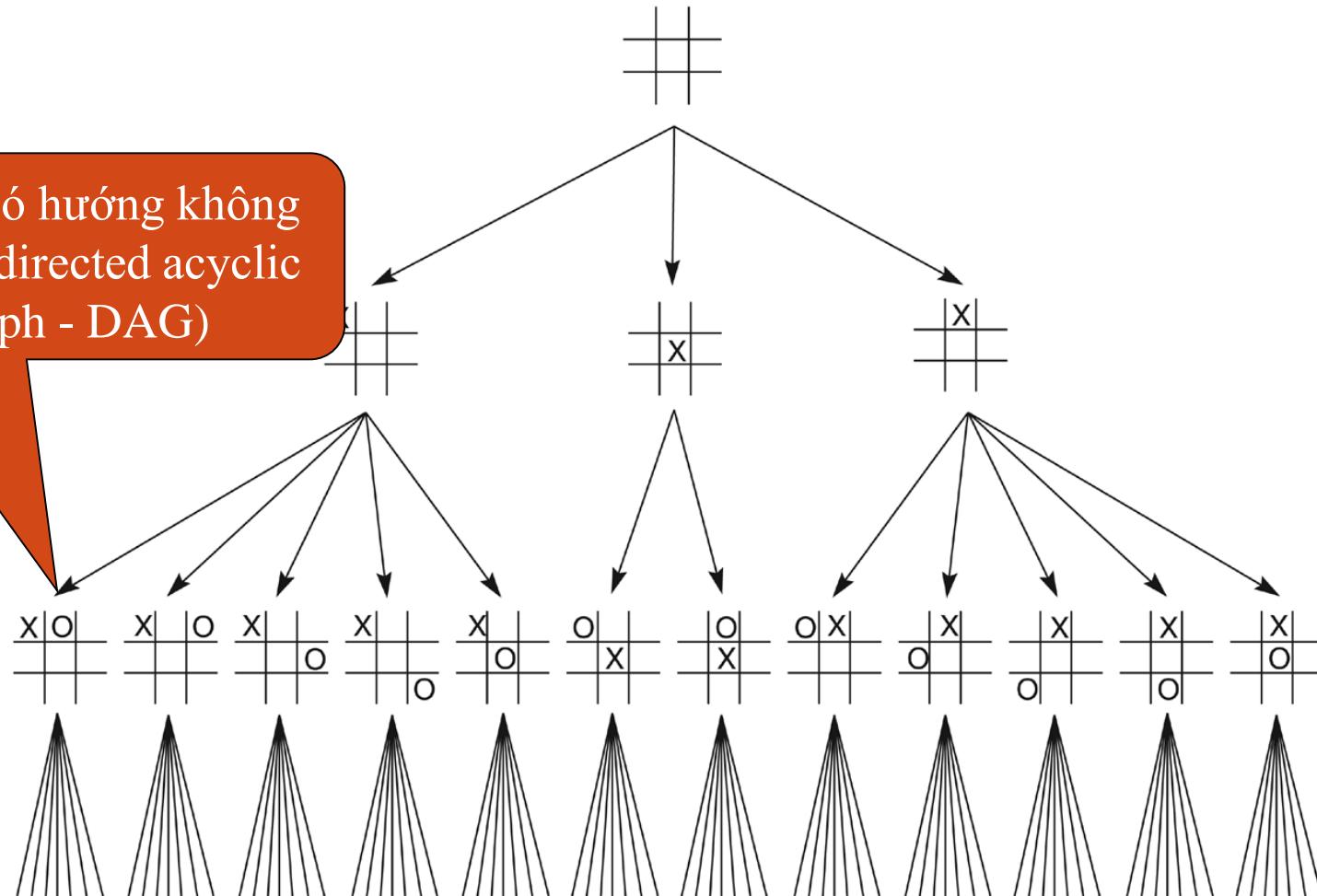
Một **KGTT** (state space) là 1 bộ $[N, A, S, GD]$ trong đó:

- **N** (node) là các *nút* hay các *trạng thái* của đồ thị.
- **A** (arc) là tập các *cung* (hay các *liên kết*) giữa các nút.
- **S** (start) là một tập chứa các *trạng thái ban đầu* của bài toán. ($S \subset N \wedge S \neq \emptyset$)
- **GD** (Goal Description) chứa các trạng thái đích của bài toán ($S \subset N \wedge S \neq \emptyset$). Các trạng thái trong **GD** được mô tả theo một trong hai đặc tính:
 - Đặc tính có thể đo lường được các trạng thái gặp trong quá trình tìm kiếm. VD: Tic-tac-toe, 8-puzzle,...
 - Đặc tính của đường đi được hình thành trong quá trình tìm kiếm. VD: TSP
- **Đường đi của lời giải** (solution path) là một con đường đi qua đồ thị này từ một nút thuộc S đến một nút thuộc GD.

2. Không gian trạng thái

Một phần KGTT triển khai trong Tic-tac-toe

Đồ thị có hướng không
lặp lại (directed acyclic
graph - DAG)



2. Không gian trạng thái

Trò đố 8 ô hay 15 ô

Trạng thái ban đầu

Trạng thái đích

- Trò đố 15 ô

11	14	4	7
10	6		5
1	2	13	15
9	12	8	3

1	2	3	4
12	13	14	5
11		15	6
10	9	8	7

- Trò đố 8 ô

	2	8
3	5	7
6	2	1

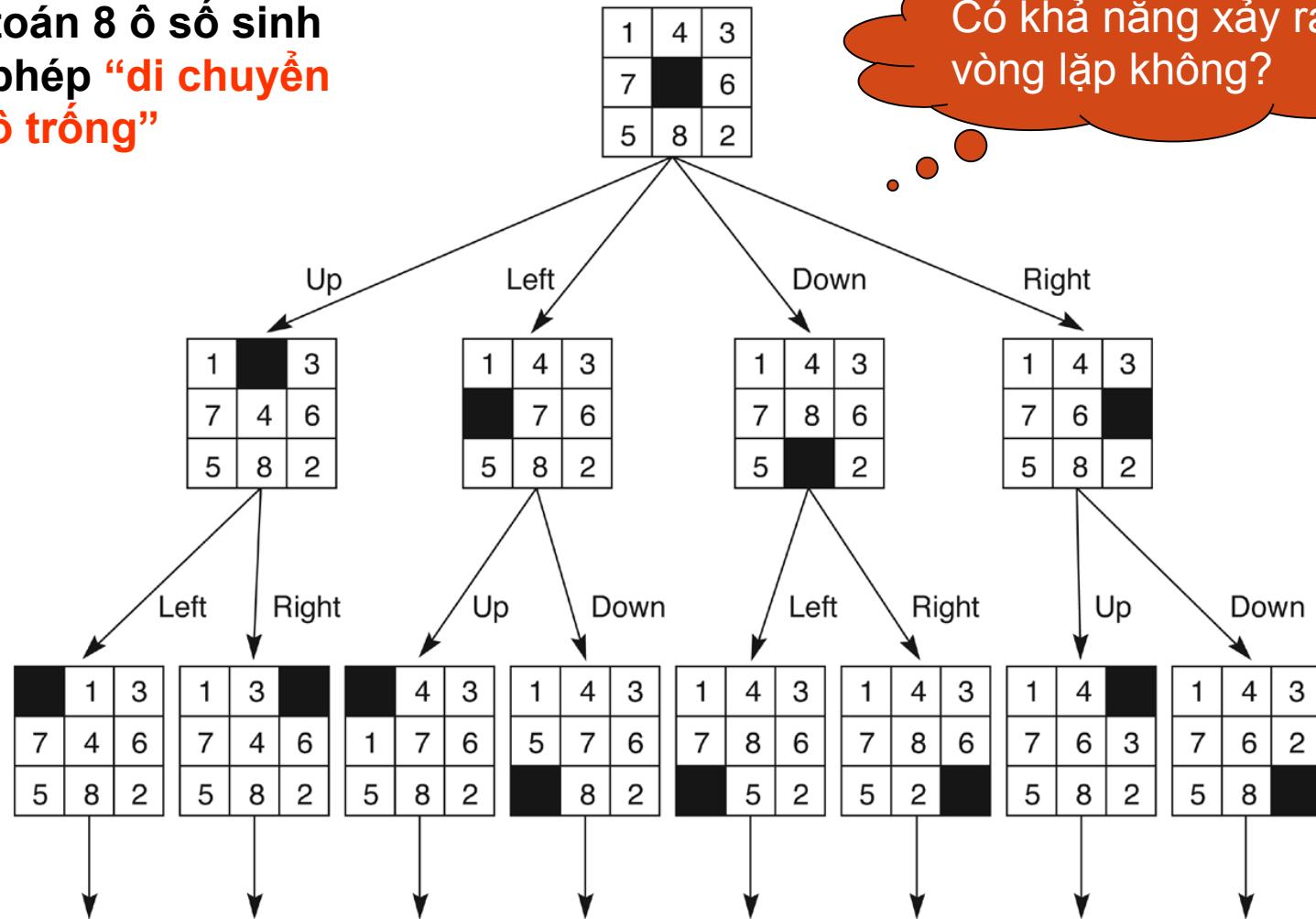
1	2	3
8		4
7	6	5

- Cần biểu diễn KGTT cho bài toán này như thế nào?

2. Không gian trạng thái

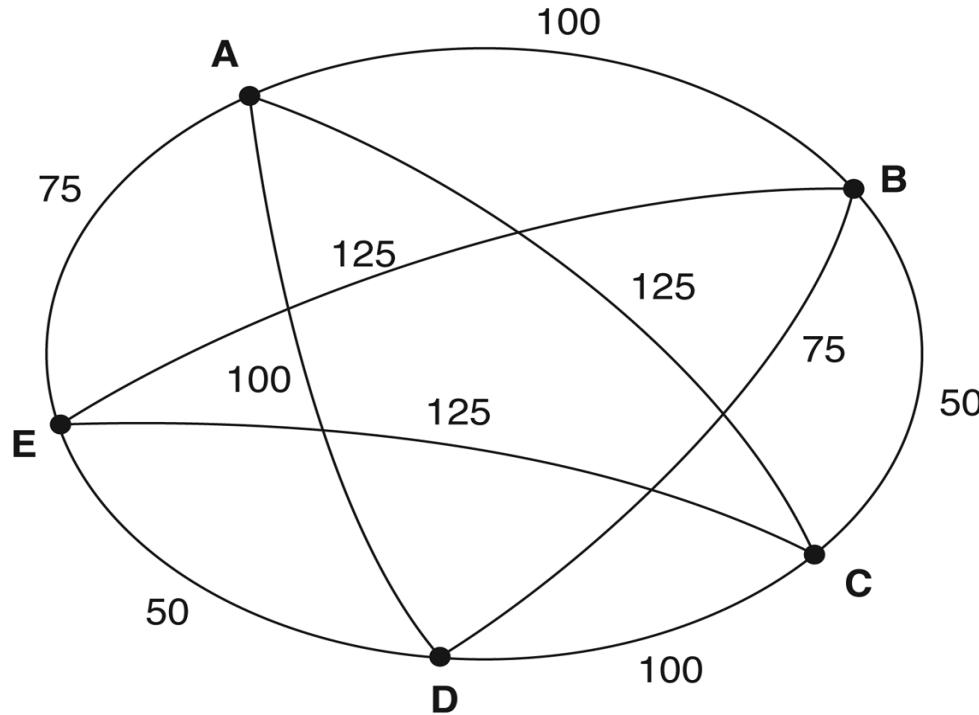
Không gian trạng thái
của bài toán 8 ô số sinh
ra bằng phép “di chuyển
ô trống”

Có khả năng xảy ra
vòng lặp không?



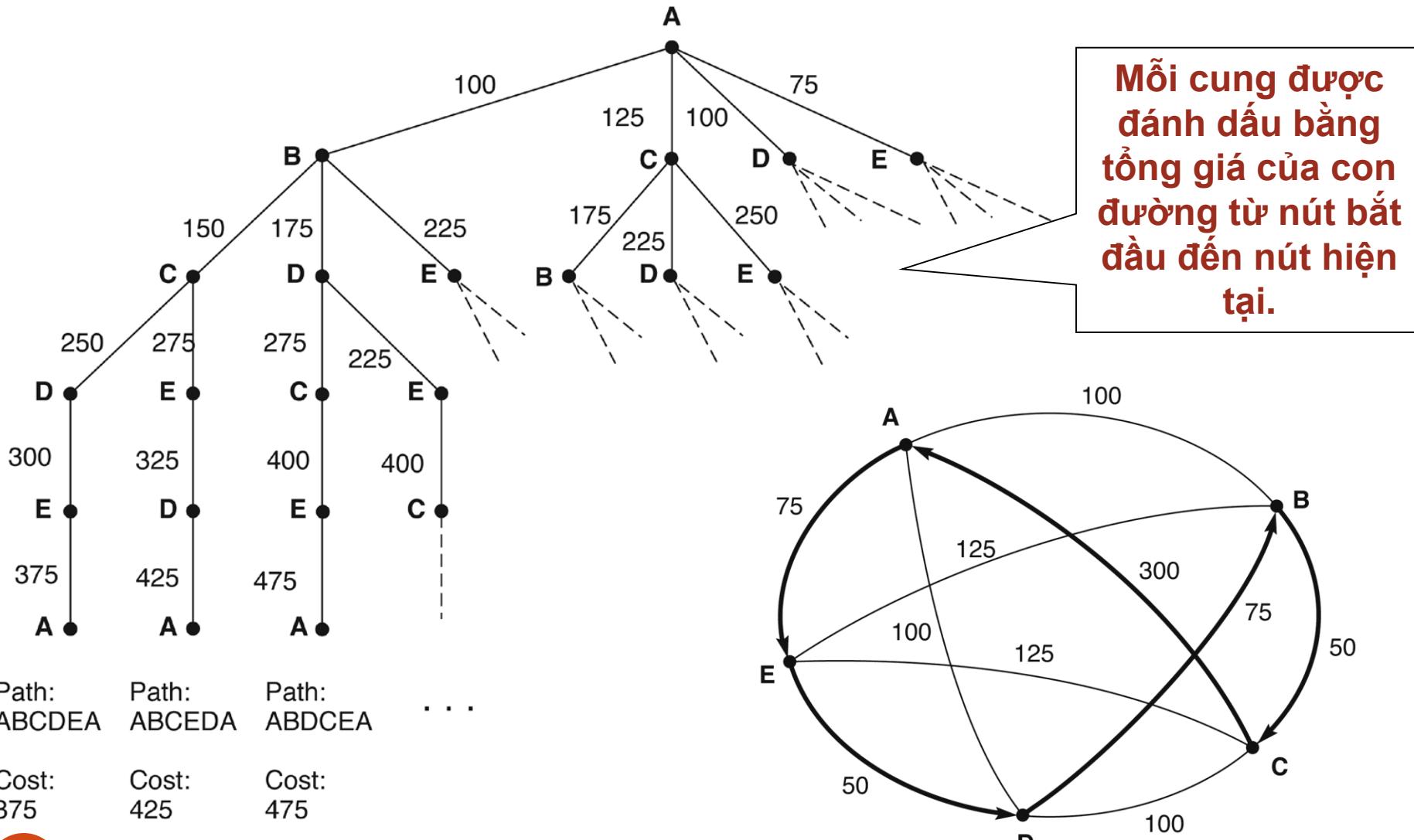
2. Không gian trạng thái

Biểu diễn không gian trạng thái bài toán người đưa thư



- Cần biểu diễn KGTT cho bài toán này như thế nào?

2. Không gian trạng thái



2. Không gian trạng thái

Các yếu tố xác định không gian trạng thái

1. Trạng thái.
2. Hành động.
3. Kiểm tra trạng thái thỏa đíc.
4. Chi phí cho mỗi bước chuyển trạng thái.

3. Phân loại vấn đề

Các đặc trưng của vấn đề

1. Tính khả tách.
2. Có thể huỷ bỏ hay lùn ngược bước giải?
3. Không gian bài toán có đoán định được trước? (sau mỗi bước giải).
4. Cần lời giải tốt hay tối ưu?
5. Lời giải là trạng thái hay dãy chuyển trạng thái?
6. Vai trò của tri thức?
7. Quá trình giải có cần tương tác người máy?

3. Phân loại vấn đề

Đơn định/nắm toán bộ không gian trạng thái	Đơn định/nắm được bộ phận trong không gian trạng thái
Không đơn định/nắm được một bộ phận của không gian trạng thái	Không đơn định/không nắm được bộ phận của không gian trạng thái

4. Các chiến lược cho TK-KGTT

1. TK hướng từ dữ liệu (Data-driven Search)

- Suy diễn tiến (forward chaining)

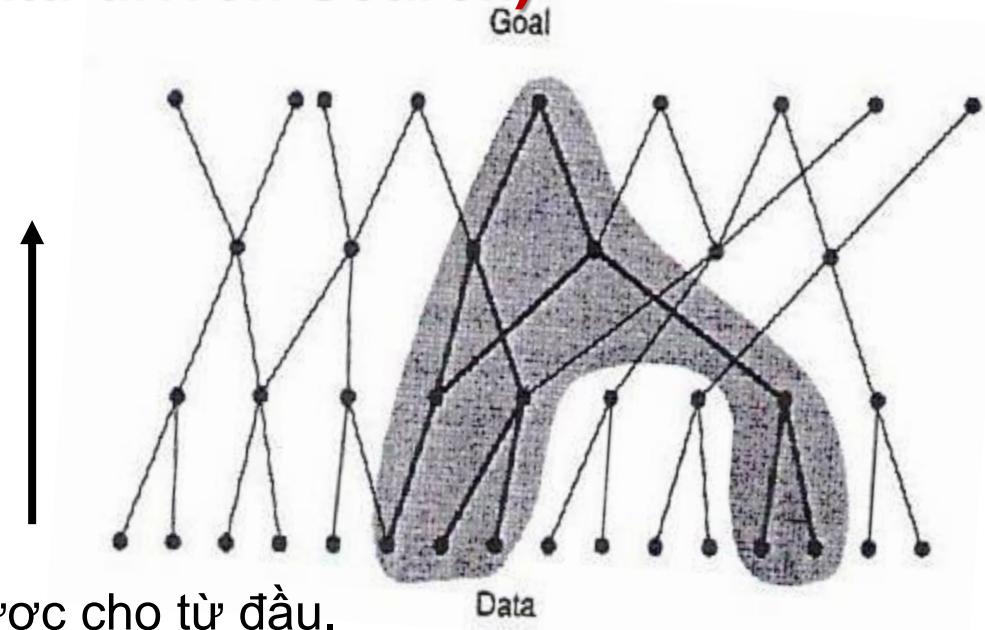
2. TK hướng từ mục tiêu (Goal-driven Search)

- Suy diễn lùi (backward chaining)

4. Các chiến lược cho TK-KGTT

1. TK hướng từ dữ liệu (Data-driven Search)

- Việc tìm kiếm đi từ dữ liệu đến mục tiêu



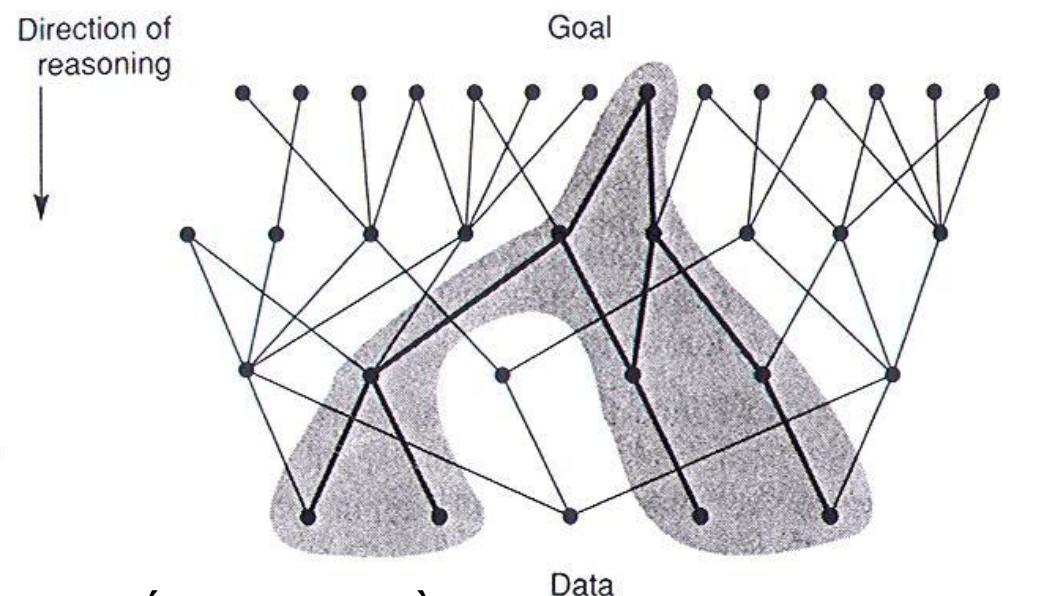
- Thích hợp khi:**

- Tất cả hoặc một phần dữ liệu được cho từ đầu.
- Có nhiều mục tiêu, nhưng chỉ có một số ít các phép toán có thể áp dụng cho một trạng thái bài toán.
- Rất khó đưa ra một mục tiêu hoặc giả thuyết ngay lúc đầu.

4. Các chiến lược cho TK-KGTT

2. TK hướng từ mục tiêu (Goal-driven Search)

- Việc tìm kiếm đi từ mục tiêu trở về dữ liệu.



- Thích hợp khi:**

- Có thể đưa ra mục tiêu hoặc giả thuyết ngay lúc đầu.
- Có nhiều phép toán có thể áp dụng trên 1 trạng thái của bài toán → sự bùng nổ số lượng các trạng thái.
- Các dữ liệu của bài toán không được cho trước, nhưng hệ thống phải đạt được trong quá trình tìm kiếm.

5. Chiến lược tìm kiếm trên đồ thị KGTT

Phương pháp đánh giá chiến lược tìm kiếm trên đồ thị KGTT:

- Chiến lược tìm kiếm là chiến lược lựa chọn thứ tự xét các nodes tạo ra.
- Các tiêu chuẩn để đánh giá chiến lược :
 - **đủ** : Liệu có tìm được lời giải (nếu có)?
 - **độ phức tạp thời gian**: số lượng node phải xét.
 - **độ phức tạp lưu trữ**: Tổng dung lượng bộ nhớ phải lưu trữ (các nodes trong quá trình tìm kiếm).
 - **tối ưu**: Có luôn cho lời giải tối ưu.
- Độ phức tạp thời gian và lưu trữ của bài toán có thể được đo bằng:
 - **b**: Độ phân nhánh của cây
 - **d**: Độ sâu của lời giải ngắn nhất
 - **m**: Độ sâu tối đa của không gian trạng thái (có thể vô hạn).

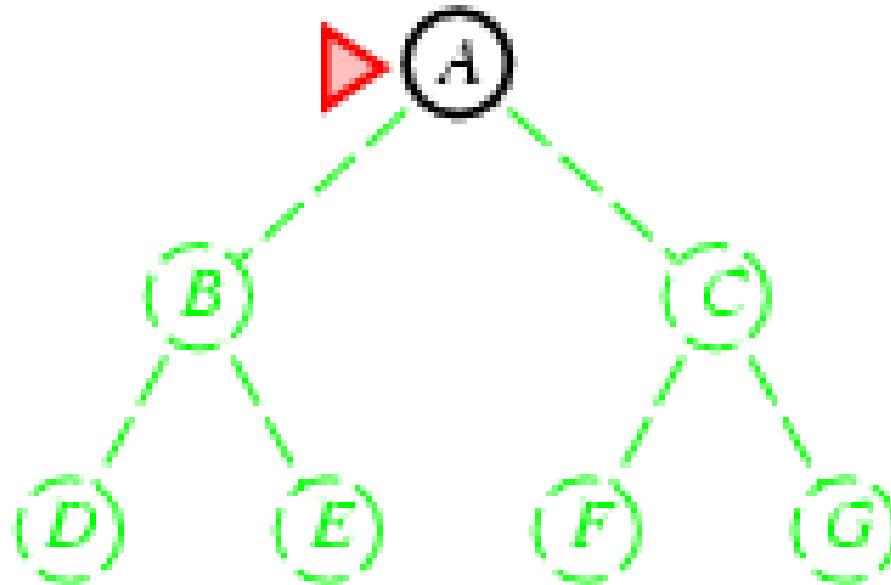
5. Chiến lược tìm kiếm trên đồ thị KGTT

Các chiến lược tìm kiếm mù (*weak/uninformed/blind search*)

- Những chiến thuật tìm kiếm chỉ sử dụng thông tin từ định nghĩa của bài toán:
 1. Tìm kiếm theo chiều rộng
 2. Tìm kiếm đều giá (uniform-cost search).
 3. Tìm kiếm theo chiều sâu.
 4. Tìm kiếm theo chiều sâu có hạn
 5. Tìm kiếm sâu dần.

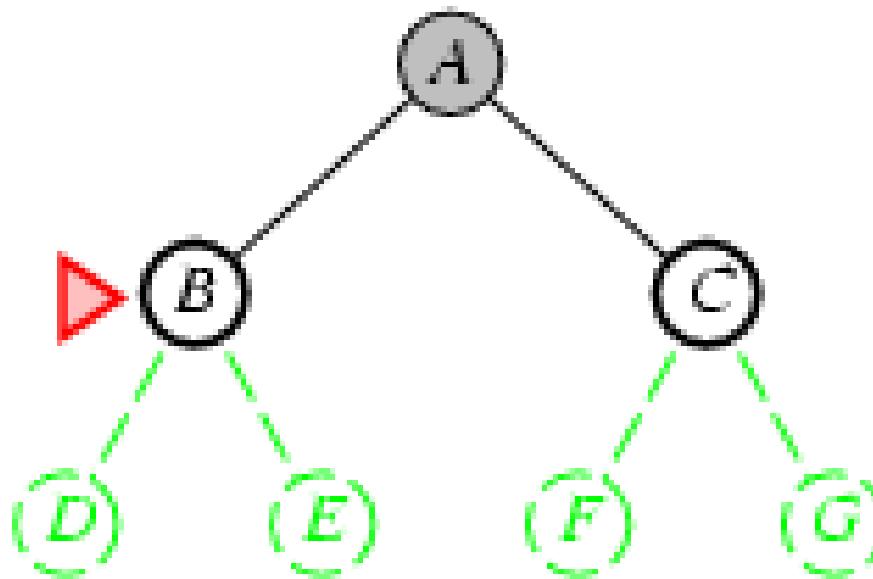
5.1. Tìm kiếm theo chiều rộng

- Tìm kiếm theo từng tầng. Phát triển các node gần nút nhất.
- Cài đặt:
 - L (danh sách các node đã được sinh ra và chờ được duyệt) được cài đặt dưới dạng danh sách FIFO, i.e., Các node con được sinh ra (bởi EXPAND) sẽ được đặt ở dưới cùng của L .



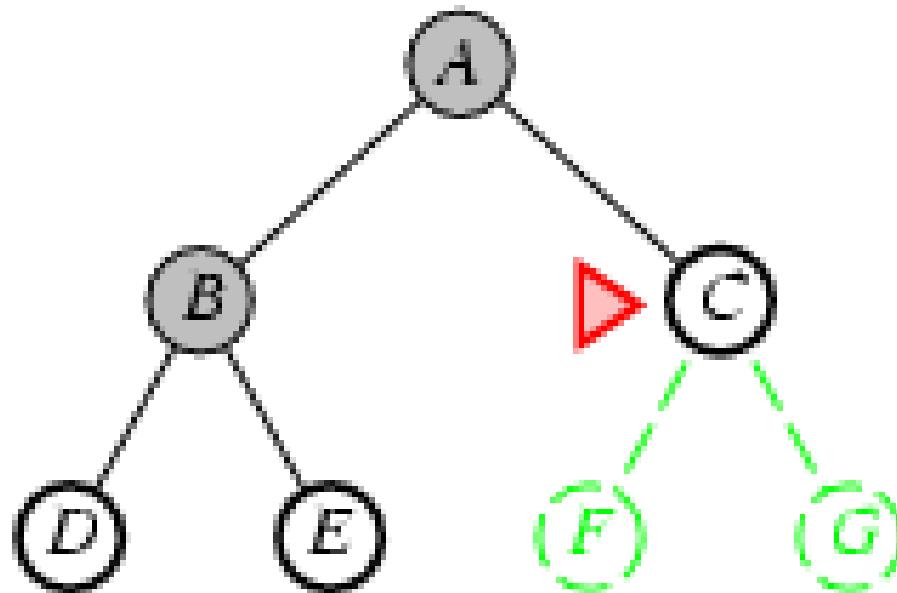
5.1. Tìm kiếm theo chiều rộng

- Tìm kiếm theo từng tầng. Phát triển các node gần nút nhất.
- Cài đặt:
 - L (danh sách các node đã được sinh ra và chờ được duyệt) được cài đặt dưới dạng danh sách FIFO, i.e., Các node con được sinh ra (bởi EXPAND) sẽ được đặt ở dưới cùng của L .



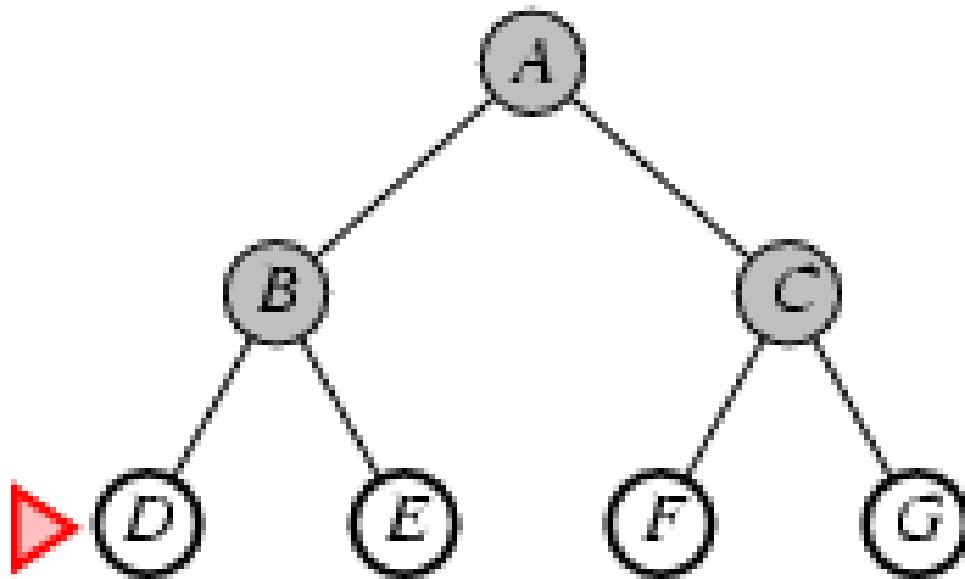
5.1. Tìm kiếm theo chiều rộng

- Tìm kiếm theo từng tầng. Phát triển các node gần nút nhất.
- Cài đặt:
 - L (danh sách các node đã được sinh ra và chờ được duyệt) được cài đặt dưới dạng danh sách FIFO, i.e., Các node con được sinh ra (bởi EXPAND) sẽ được đặt ở dưới cùng của L .



5.1. Tìm kiếm theo chiều rộng

- Tìm kiếm theo từng tầng. Phát triển các node gần nút nhất.
- Cài đặt:
 - L (danh sách các node đã được sinh ra và chờ được duyệt) được cài đặt dưới dạng danh sách FIFO, i.e., Các node con được sinh ra (bởi EXPAND) sẽ được đặt ở dưới cùng của L .



5.1. Tìm kiếm theo chiều rộng

Cài đặt thuật toán tìm kiếm theo chiều rộng

Thuật toán tìm kiếm theo bề rộng được mô tả bởi thủ tục sau:

procedure Breadth_First_Search;

begin

1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu;

2. **loop do**

 2.1 **if** L rỗng **then** {thông báo tìm kiếm thất bại; **stop**};

 2.2 Loại trạng thái u ở đầu danh sách L ;

 2.3 **if** u là trạng thái kết thúc **then** {thông báo tìm kiếm thành công; **stop**};

 2.4 **for** mỗi trạng thái v kề u **do** {

 Đặt v vào cuối danh sách L ;

$\text{father}(v) \leftarrow u$ }

end;

Trong thuật toán, sử dụng hàm $\text{father}(v)$ để lưu lại cha của mỗi đỉnh trên đường đi, $\text{father}(v)=u$ nếu u là cha của đỉnh v .

5.1. Tìm kiếm theo chiều rộng

Nhận xét về thuật toán tìm kiếm theo chiều rộng.

- Trong thuật toán tìm kiếm theo chiều rộng, trạng thái nào được sinh ra trước sẽ được phát triển trước, do đó danh sách **L** được sử dụng là hàng đợi. Trong bước 2.3, ta cần kiểm tra xem **u** có là trạng thái kết thúc không. Nói chung, các trạng thái kết thúc được xác định bởi điều kiện nào đó.
- Nếu bài toán có nghiệm (tồn tại đường đi từ trạng thái đầu tới trạng thái kết thúc), thì thuật toán sẽ tìm được nghiệm.

5.1. Tìm kiếm theo chiều rộng

Đánh giá tìm kiếm BFS

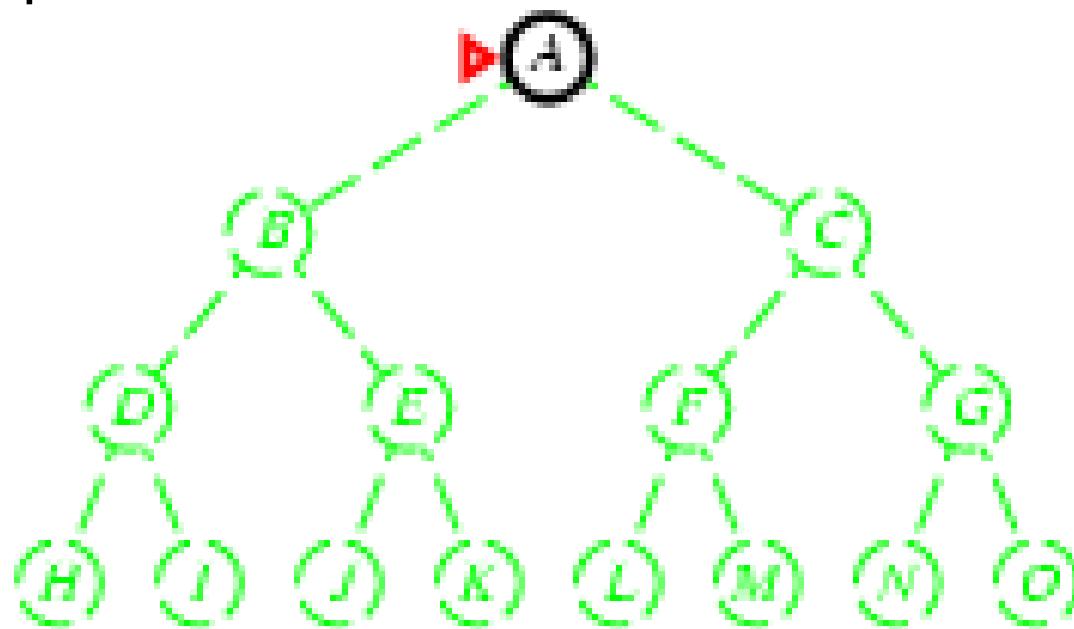
- **Tính đủ?**
 - có (nếu b là hữu hạn).
- **Thời gian?**
 - $1+b+b^2+b^3+\dots +b^d + b(b^d-1) = O(b^{d+1})$.
- **Không gian trạng thái?**
 - $O(b^{d+1})$ (lưu mọi node của cây).
- **Tính tối ưu?**
 - có (giải thiết giá của mỗi bước chuyển là 1)
- **Nhận xét chung:** Không gian lưu trữ hết sức tốn kém là vấn đề lớn nhất đối với phương pháp tìm kiếm theo chiều rộng!

5.2. Tìm kiếm đều giá (uniform-cost search)

- **Ý tưởng:** Xét node có giá tìm kiếm nhỏ nhất trước.
- **Cài đặt:**
 - L = Hàng đợi có ưu tiên (bằng nghịch dấu giá thành)
- **Nhận xét chung:** Tương đương với BFS nếu các node có giá như nhau.
- **Đánh giá phương pháp:**
 - **Tính đủ?**
 - Có nếu $\text{cost} \geq \varepsilon$
 - **Thời gian?**
 - Số lượng nodes với $g \leq \text{giá}$ của lời giải tối ưu, $O(b^{\text{ceiling}(C^*/ \varepsilon)})$ trong đó C^* là giá của lời giải tối ưu.
 - **Không gian trạng thái?**
 - Số lượng nodes với $g \leq \text{giá}$ của lời giải tối ưu, $O(b^{\text{ceiling}(C^*/ \varepsilon)})$.
 - **Tính tối ưu?**
 - Có – nodes được EXPAND theo thứ tự tăng dần của $g(n)$.

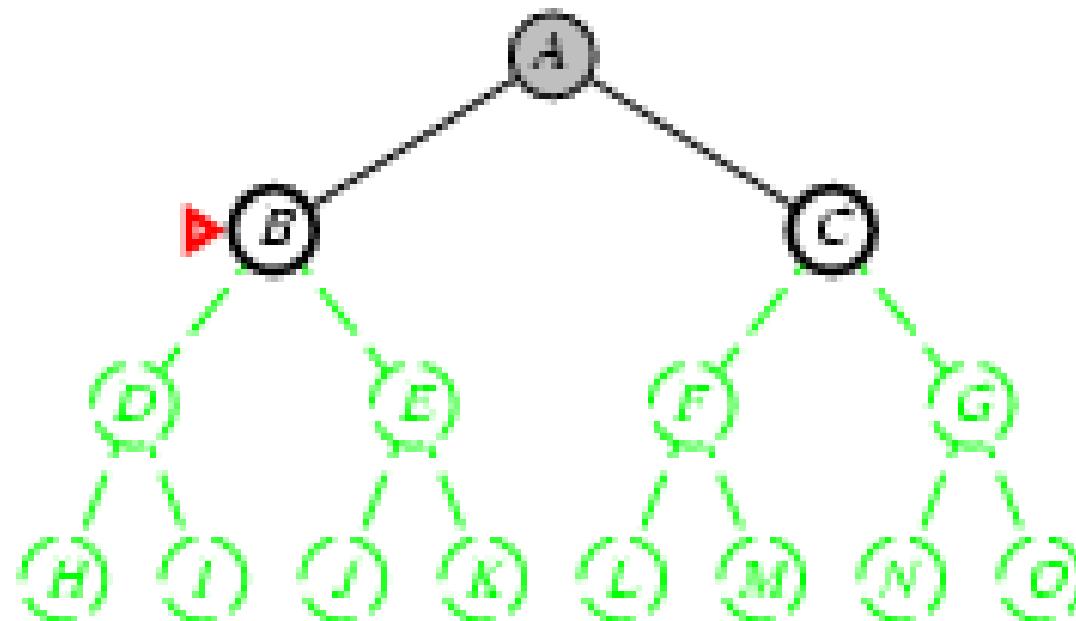
5.3. Tìm kiếm theo chiều sâu

- Phát triển các node chưa xét ở sâu nhất.
- **Cài đặt:**
 - L = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi quá trình phát triển vào đầu L .



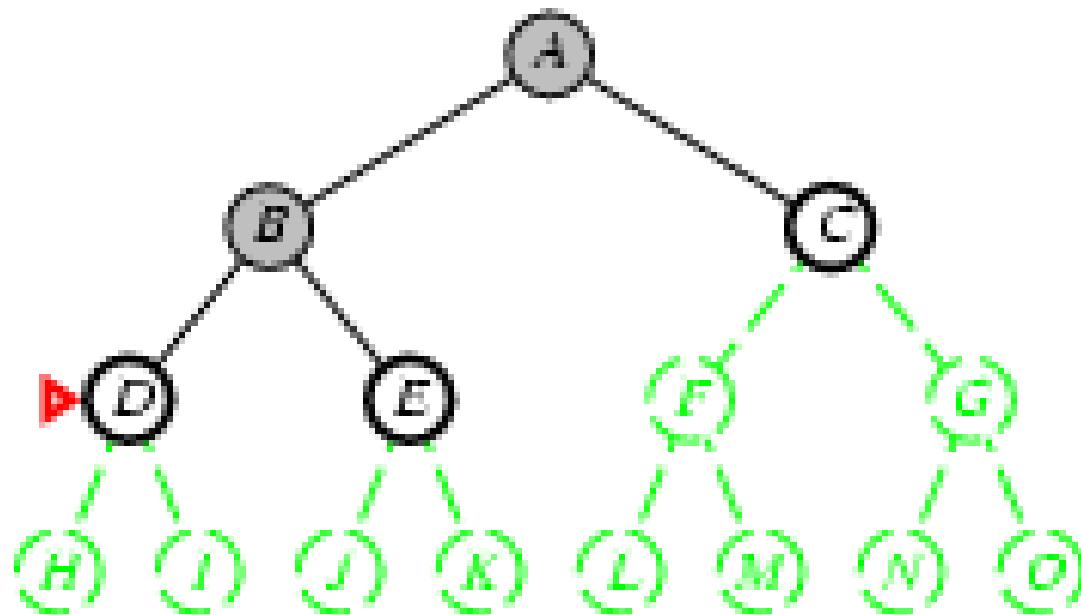
5.3. Tìm kiếm theo chiều sâu

- Phát triển các node chưa xét ở sâu nhất.
- **Cài đặt:**
 - L = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi quá trình phát triển vào đầu L .



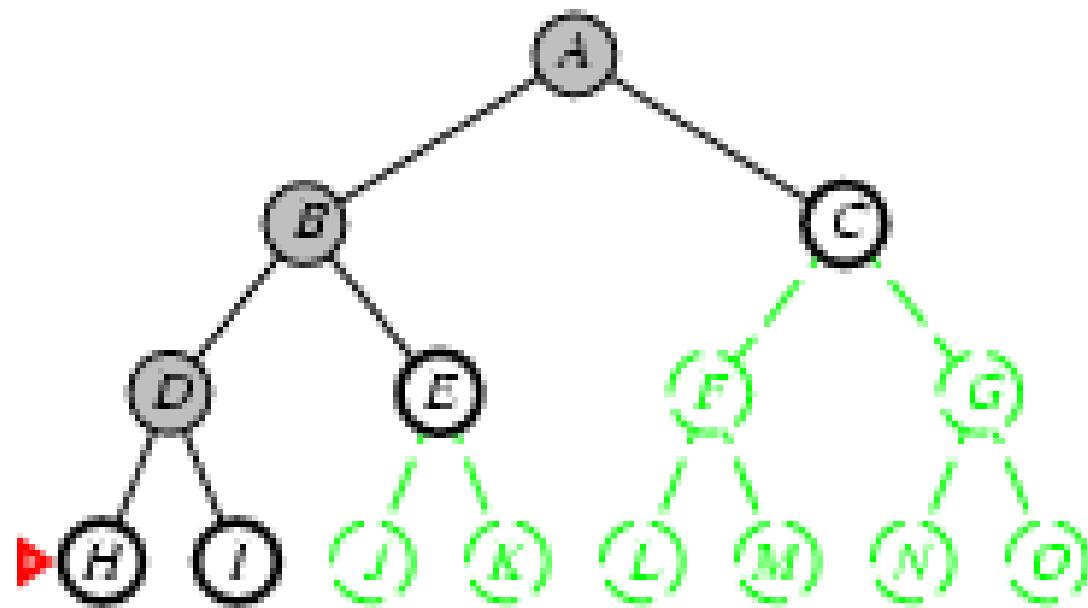
5.3. Tìm kiếm theo chiều sâu

- Phát triển các node chưa xét ở sâu nhất.
- **Cài đặt:**
 - L = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi quá trình phát triển vào đầu L .



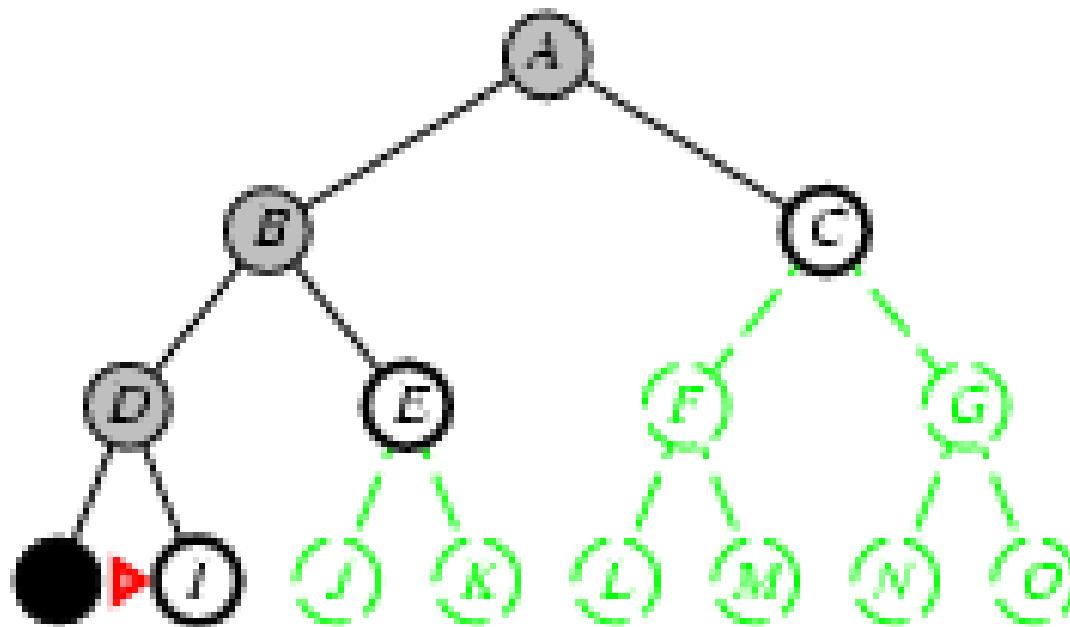
5.3. Tìm kiếm theo chiều sâu

- Phát triển các node chưa xét ở sâu nhất.
- **Cài đặt:**
 - L = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi quá trình phát triển vào đầu L .



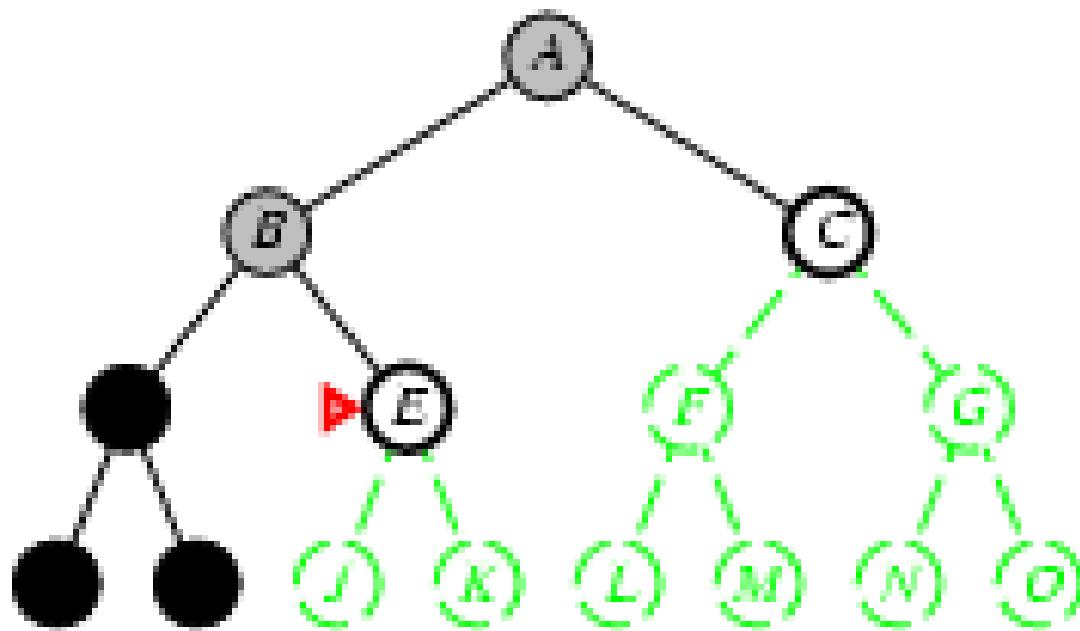
5.3. Tìm kiếm theo chiều sâu

- Phát triển các node chưa xét ở sâu nhất.
- **Cài đặt:**
 - L = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi quá trình phát triển vào đầu L .



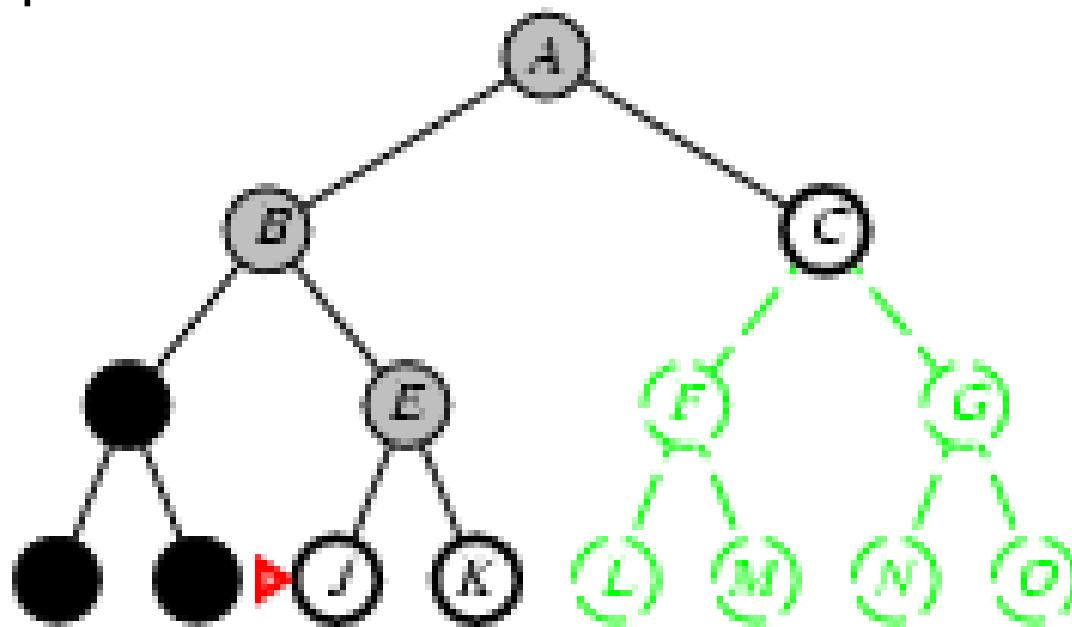
5.3. Tìm kiếm theo chiều sâu

- Phát triển các node chưa xét ở sâu nhất.
- **Cài đặt:**
 - L = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi quá trình phát triển vào đầu L .



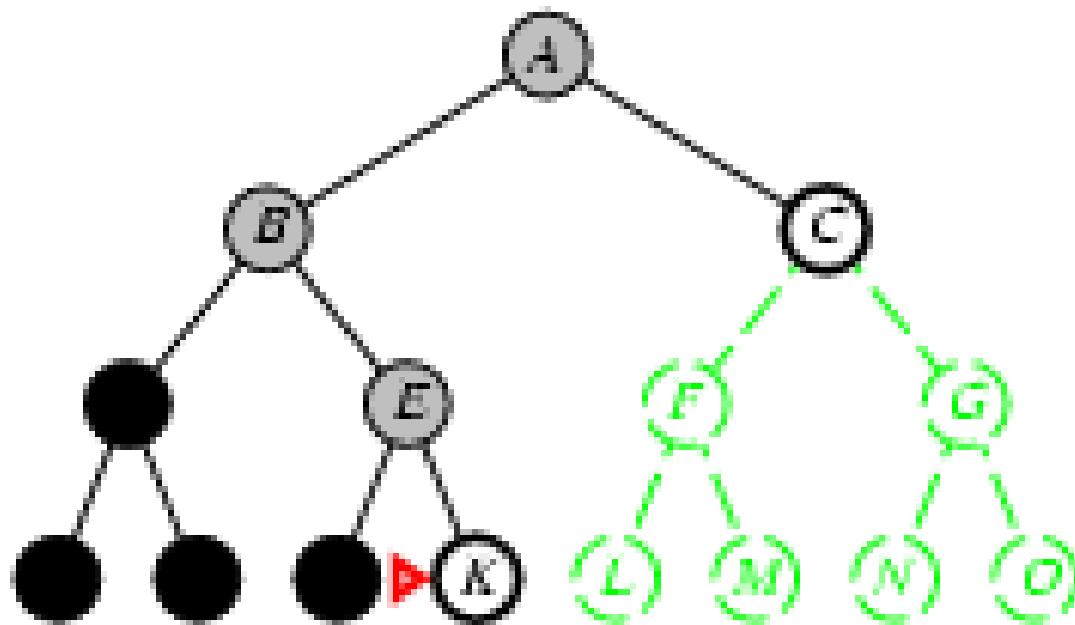
5.3. Tìm kiếm theo chiều sâu

- Phát triển các node chưa xét ở sâu nhất.
- **Cài đặt:**
 - L = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi quá trình phát triển vào đầu L .



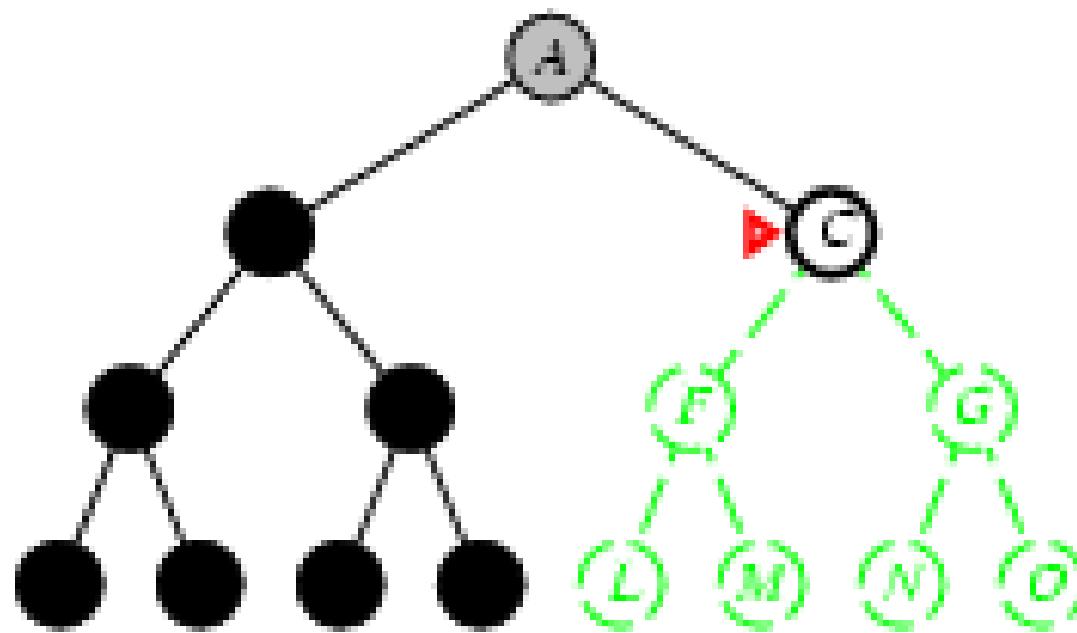
5.3. Tìm kiếm theo chiều sâu

- Phát triển các node chưa xét ở sâu nhất.
- **Cài đặt:**
 - L = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi quá trình phát triển vào đầu L .



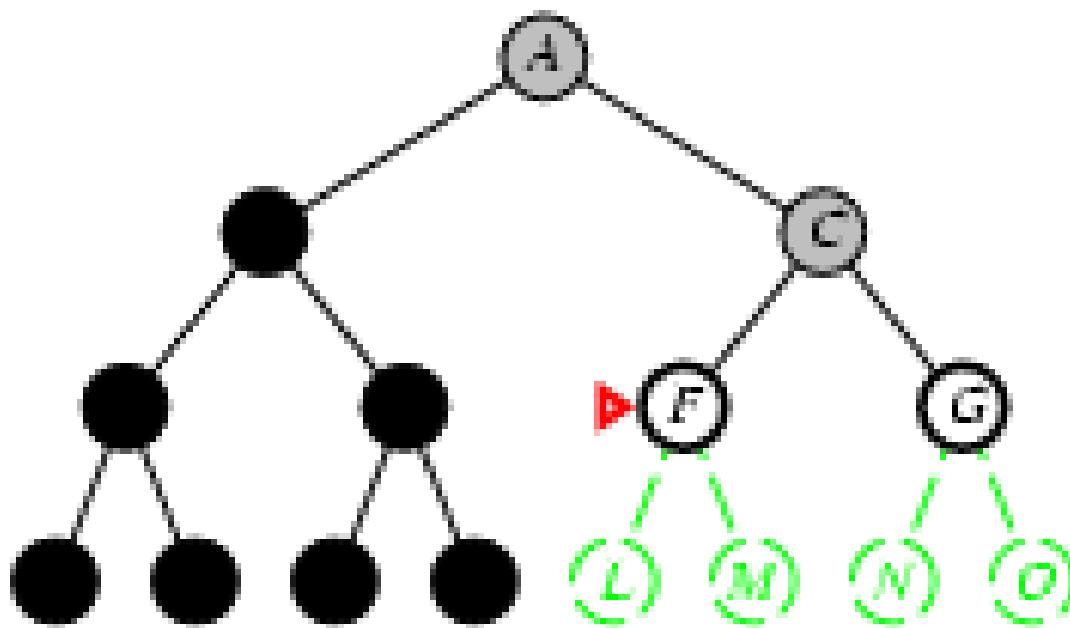
5.3. Tìm kiếm theo chiều sâu

- Phát triển các node chưa xét ở sâu nhất.
- **Cài đặt:**
 - L = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi quá trình phát triển vào đầu L .



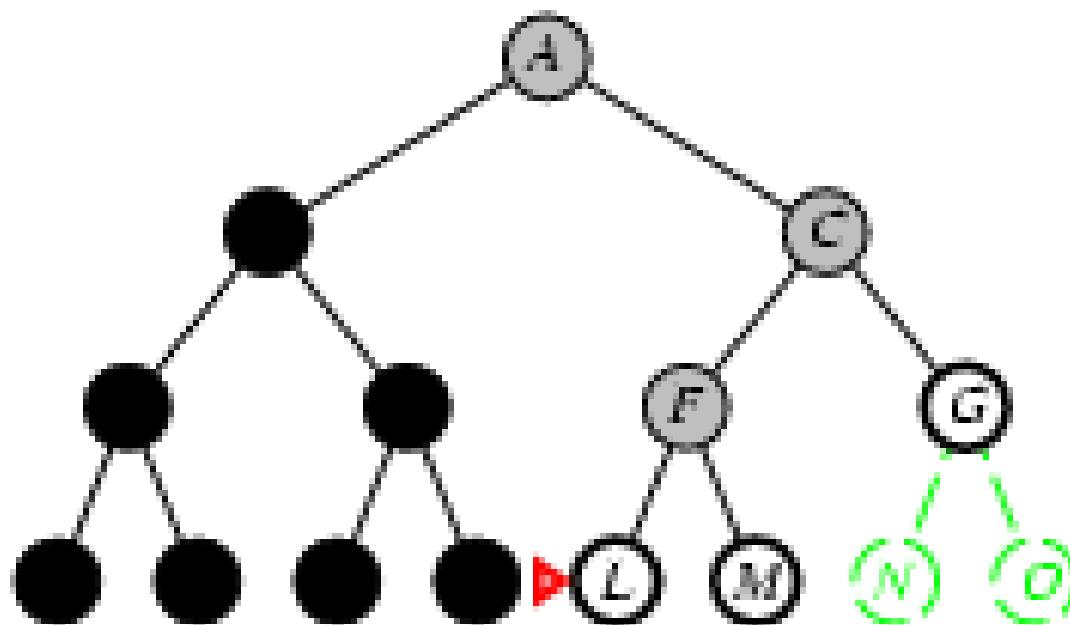
5.3. Tìm kiếm theo chiều sâu

- Phát triển các node chưa xét ở sâu nhất.
- **Cài đặt:**
 - L = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi quá trình phát triển vào đầu L .



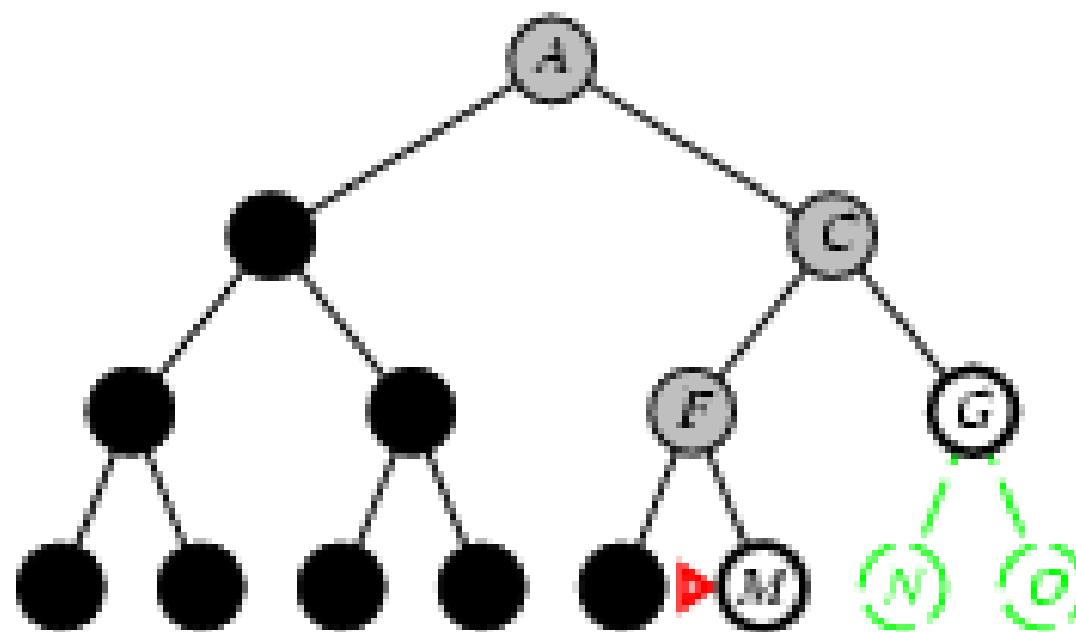
5.3. Tìm kiếm theo chiều sâu

- Phát triển các node chưa xét ở sâu nhất.
- **Cài đặt:**
 - L = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi quá trình phát triển vào đầu L .



5.3. Tìm kiếm theo chiều sâu

- Phát triển các node chưa xét ở sâu nhất.
- **Cài đặt:**
 - L = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi quá trình phát triển vào đầu L .



5.3. Tìm kiếm theo chiều sâu

Cài đặt thuật toán tìm kiếm theo chiều sâu

Procedure Depth_First_Search;

begin

1. Khởi tạo danh sách **L** chỉ chứa trạng thái ban đầu u_0 ;

2. **loop do**

 2.1. **if L** rỗng **then**

 {thông báo thất bại; stop};

 2.2. Loại trạng thái u ở đầu danh sách **L**;

 2.3. **if u** là trạng thái kết thúc **then**

 {thông báo thành công; stop};

 2.4. **for** mỗi trạng thái **v** kè **u do**

 {Đặt **v** vào đầu danh sách **L**;};

end;

5.3. Tìm kiếm theo chiều sâu

Nhận xét thuật toán tìm kiếm theo chiều sâu

- Đối BFS: luôn tìm ra nghiệm nếu bài toán có nghiệm.
- Đối với DFS: không phải với bất kỳ bài toán có nghiệm nào thuật toán tìm kiếm theo chiều sâu cũng tìm ra nghiệm. (với bài toán có nghiệm và không gian tìm kiếm hữu hạn mới tìm được nghiệm).
- **Lý do?**
 - Trong trường hợp không gian trạng thái là vô hạn, thì có thể nó không tìm ra nghiệm vì ta luôn đi sâu xuống, nếu ta đi theo nhánh vô hạn mà nghiệm không nằm trên nhánh đó thì thuật toán sẽ không dừng.

5.3. Tìm kiếm theo chiều sâu

Đánh giá tìm kiếm DFS

- **Tính đủ?**

- Không đủ (không gian vô hạn hoặc loop)
- Nếu sửa để tránh trùng lặp → đủ trong không gian hữu hạn.

- **Thời gian?**

- $O(b^m)$
- Rất xấu nếu m lớn hơn nhiều so với d .
- Nhưng nếu mật độ lời giải trong không gian lớn thì có thể nhanh hơn BFS.

- **Không gian trạng thái?**

- $O(bm)$, i.e., Độ phức tạp tuyến tính

- **Tính tối ưu?**

- Không

5.4. Tìm kiếm với độ sâu giới hạn

Là thuật toán DFS với độ sâu giới hạn là d , i.e., nodes tại độ sâu d không có node con (hàm successor trả về rỗng).

- Cài đặt:

```
function DEPTH-LIMITED-SEARCH( problem, limit) returns soln/fail/cutoff
    RECURSIVE-DLS(MAKE-NODE(INITIAL-STATE[problem]), problem, limit)

function RECURSIVE-DLS(node, problem, limit) returns soln/fail/cutoff
    cutoff-occurred? ← false
    if GOAL-TEST[problem](STATE[node]) then return SOLUTION(node)
    else if DEPTH[node] = limit then return cutoff
    else for each successor in EXPAND(node, problem) do
        result ← RECURSIVE-DLS(successor, problem, limit)
        if result = cutoff then cutoff-occurred? ← true
        else if result ≠ failure then return result
    if cutoff-occurred? then return cutoff else return failure
```

5.4. Tìm kiếm với độ sâu giới hạn

Procedure Depth_Limited_Search(d);

begin

1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu u_0 ;

depth(u_0) $\leftarrow 0$;

2. **loop do**

 2.1. **if** L rỗng **then**

 {thông báo thất bại; stop};

 2.2. Loại trạng thái u ở đầu danh sách L;

 2.3. **if** u là trạng thái kết thúc **then**

 {thông báo thành công; stop};

 2.4. **if** depth(u) $\leq d$ **then**

for mỗi trạng thái v kề u **do**

 {Đặt v vào đầu danh sách L;

 depth(v) $\leftarrow \text{depth}(u) + 1$ };

end;

5.5. Tìm kiếm sâu dần

- **Độ sâu giới hạn (depth bound):**

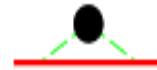
- Giải thuật TK sâu sẽ quay lui khi trạng thái đang xét đạt đến độ sâu giới hạn đã định.

- **TK Sâu bằng cách đào sâu nhiều lần:**

- TK sâu với độ sâu giới hạn là 1, nếu thất bại, nó sẽ lặp lại giải thuật TK sâu với độ sâu là 2,...
- Giải thuật tiếp tục cho đến khi tìm được mục tiêu, mỗi lần lặp lại tăng độ sâu lên 1.
- **Giải thuật này có độ phức tạp về thời gian cùng bậc với tìm kiếm theo chiều rộng và tìm kiếm theo chiều sâu.**

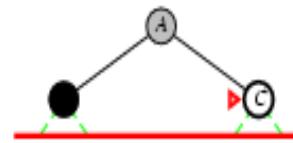
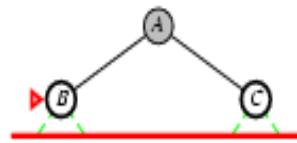
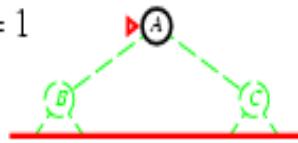
5.5. Tìm kiếm sâu dần $d=0$

Limit = 0



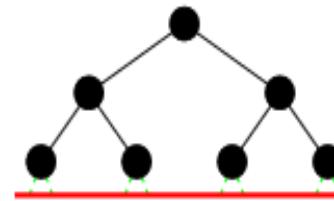
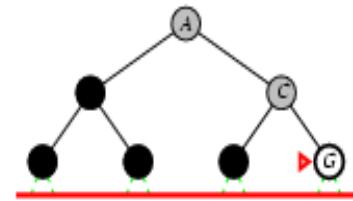
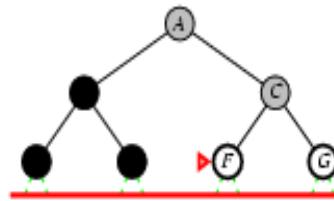
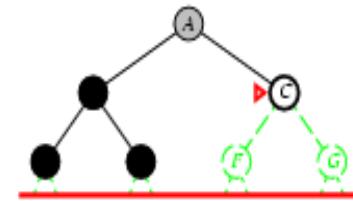
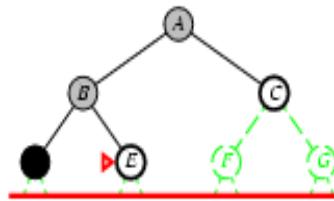
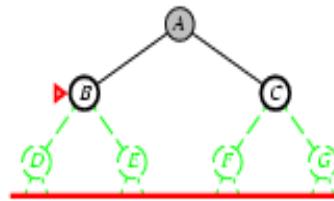
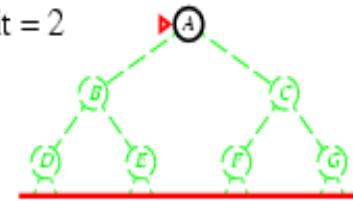
5.5. Tìm kiếm sâu dần $d=1$

Limit = 1



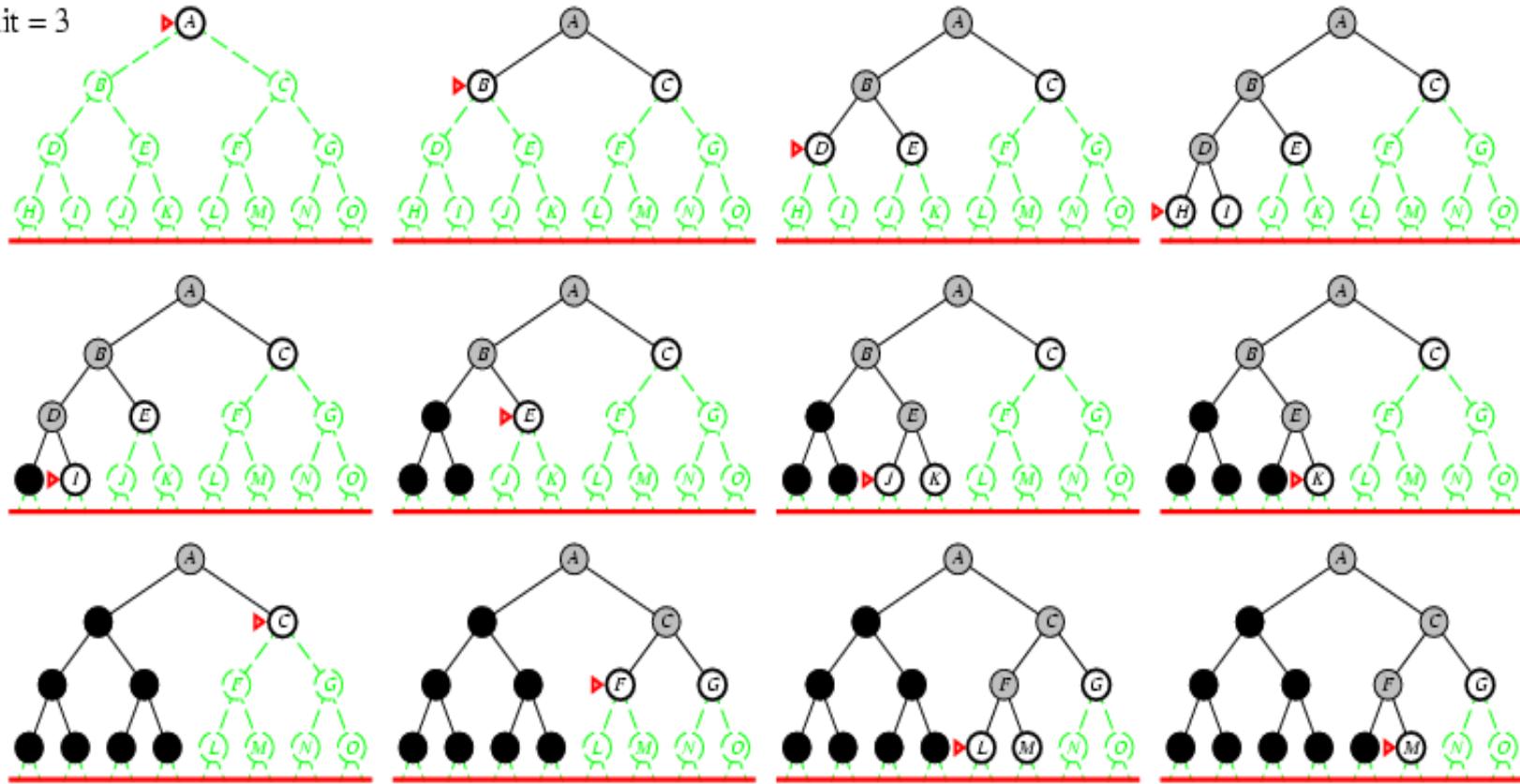
5.5. Tìm kiếm sâu dần $d = 2$

Limit = 2



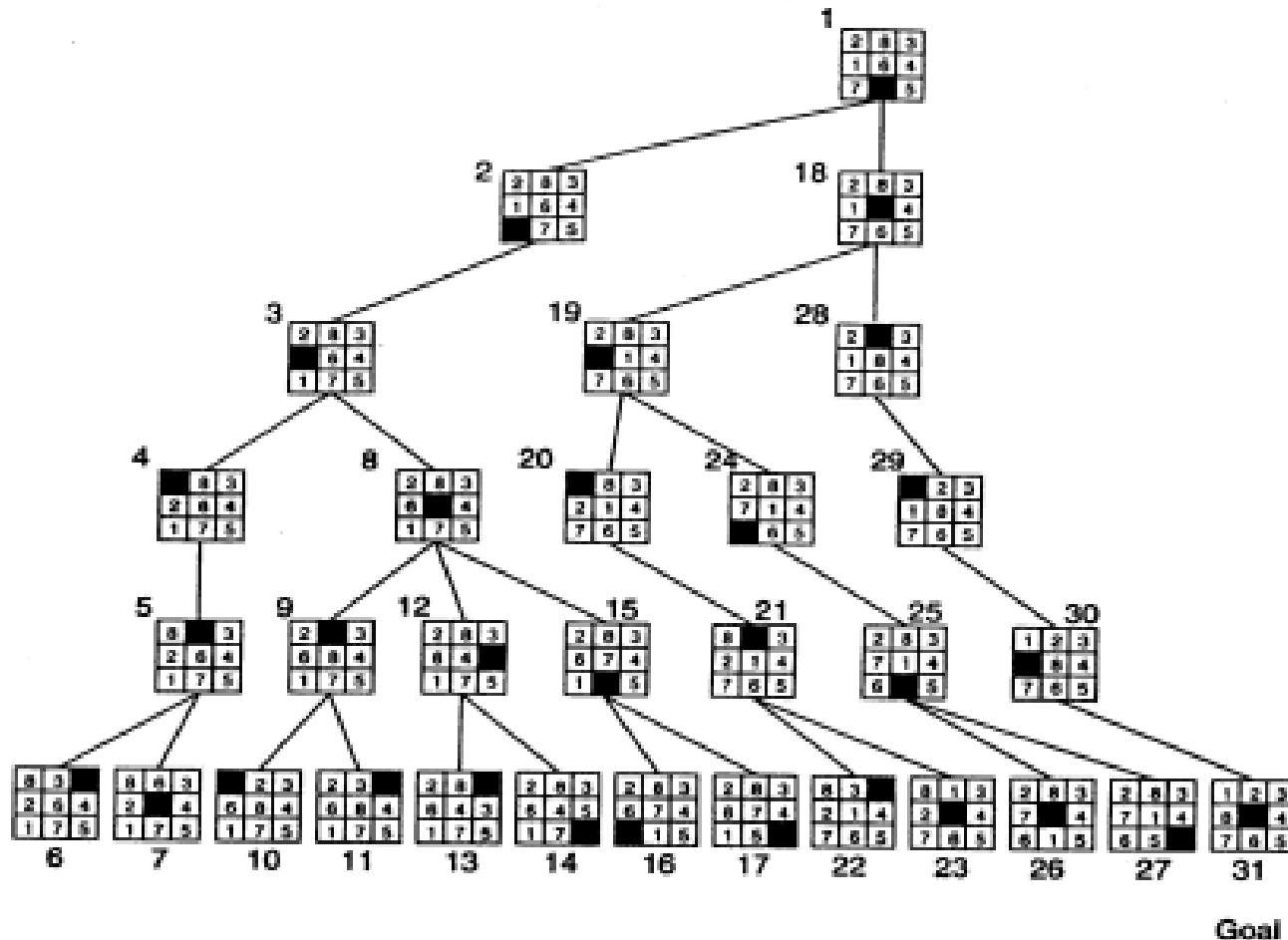
5.5. Tìm kiếm sâu dần $d = 3$

Limit = 3



5.5. Ví dụ: Trò chơi ô đố 8-puzzle

The 8-puzzle searched by a production system with loop detection and depth bound 5



5.5. Tìm kiếm sâu dần

Cài đặt thuật toán tìm kiếm sâu dần

Procedure Depth_Deepening_Search;

Begin

For d← 0 **to** max **do**

{Depth_Limited_Search(d);

If thành công **then exit**}

End;

5.5. Tìm kiếm sâu dần

Nhận xét về tìm kiếm sâu dần:

- Số lượng nodes được sinh trong giới hạn độ sâu d với độ phân nhánh b:

$$N_{DLS} = b^0 + b^1 + b^2 + \dots + b^{d-2} + b^{d-1} + b^d$$

- Số lượng nodes được sinh trong tìm kiếm sâu dần với độ sâu d độ phân nhánh b:

$$N_{IDS} = (d+1)b^0 + d b^1 + (d-1)b^2 + \dots + 3b^{d-2} + 2b^{d-1} + 1b^d$$

- Ví dụ b = 10, d = 5.,

- $N_{DLS} = 1 + 10 + 100 + 1,000 + 10,000 + 100,000 = 111,111.$
- $N_{IDS} = 6 + 50 + 400 + 3,000 + 20,000 + 100,000 = 123,456.$
- Tỷ lệ $(123,456 - 111,111)/111,111 = 11\%$

5.5. Tìm kiếm sâu dần

Đánh giá tìm kiếm sâu dần

- **Tính đủ ?**

- Có.

- **Thời gian?**

- $(d+1)b^0 + d b^1 + (d-1)b^2 + \dots + b^d = O(b^d)$.

- **Không gian trạng thái?**

- $O(bd)$.

- **Tính tối ưu?**

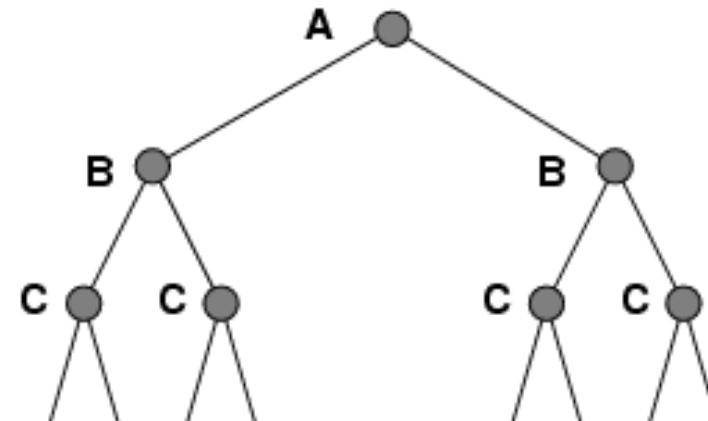
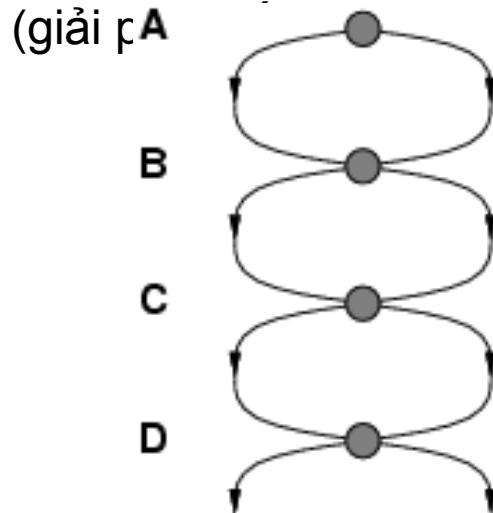
- Có, nếu giá mỗi bước = 1.

Tóm tắt các chiến lược tìm kiếm

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited Deepening	Iterative
Complete?	Yes	Yes	No	No	Yes
Time	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(b^m)$	$O(b^l)$	$O(b^d)$
Space	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(bm)$	$O(bl)$	$O(bd)$
Optimal?	Yes	Yes	No	No	Yes

5.6. Trạng thái bị trùng lắp

- Việc không xử lý tốt các trạng thái bị lặp nhiều lần có thể làm cho độ phức tạp (thời gian, không gian) bị bùng nổ tổ hợp.
- Giải pháp:
 - Khi phát triển đỉnh u, không sinh ra các đỉnh trùng với cha của u.
 - Khi phát triển đỉnh u, không sinh ra các đỉnh trùng với một đỉnh nào đó nằm trên đường dẫn tới u.
 - Không sinh ra các đỉnh mà nó đã được sinh ra, tức là chỉ sinh ra các đỉnh mới.



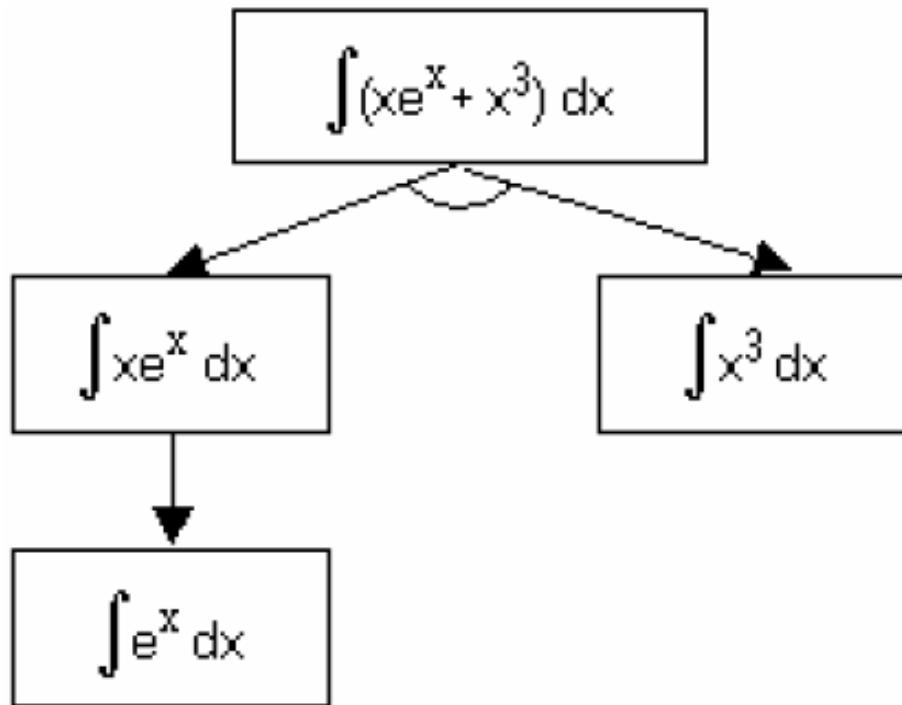
5.6. Chia để trị

- Thông thường, các bài toán được quy về việc tìm đường trong không gian trạng thái.
- Để giải quyết vấn đề, có thể chia nhỏ bài toán thành các vấn đề con. Việc này được thực hiện lặp lại nhiều lần đến khi các vấn đề này có thể giải quyết được.
- Một số ví dụ về phương pháp chia để trị.

5.6. Ví dụ về phương pháp: Tính tích phân

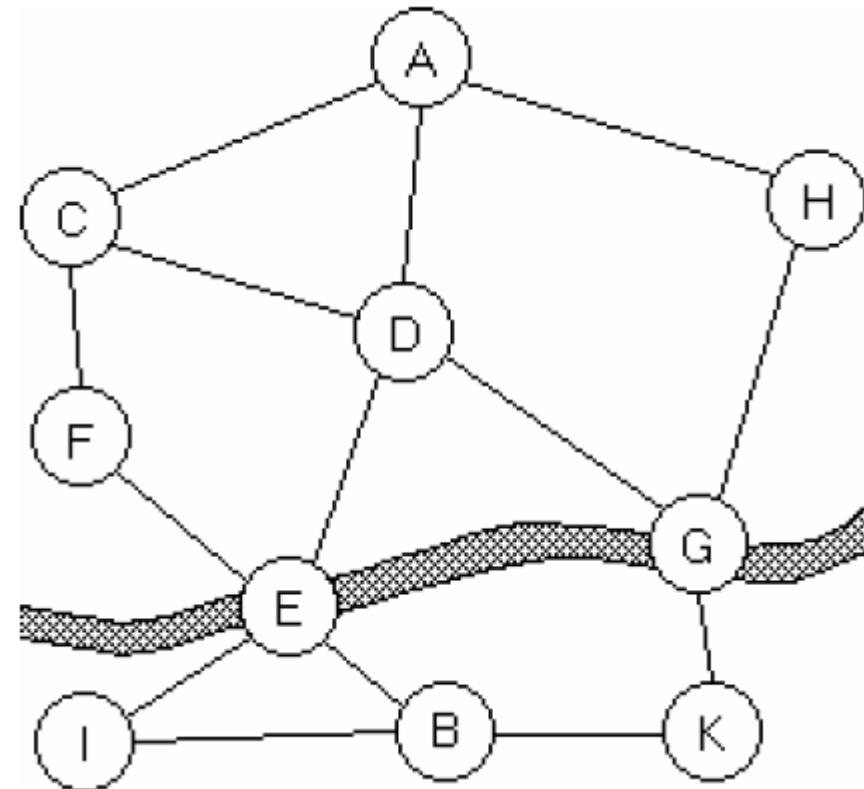
- Tính tích phân:
- Để tính được tích phân này có thể sử dụng mô hình sau:

$$\int (xe^x + x^3) dx$$



5.6. Ví dụ về phương pháp: Tìm đường

- Giả sử có bản đồ một thành phố như sau:
- Cần tìm đường đi từ A đến B. Như vậy, có thể có 2 trường hợp:
 - Đường đi từ A đến B qua E,
 - Đường đi từ A đến B qua G.

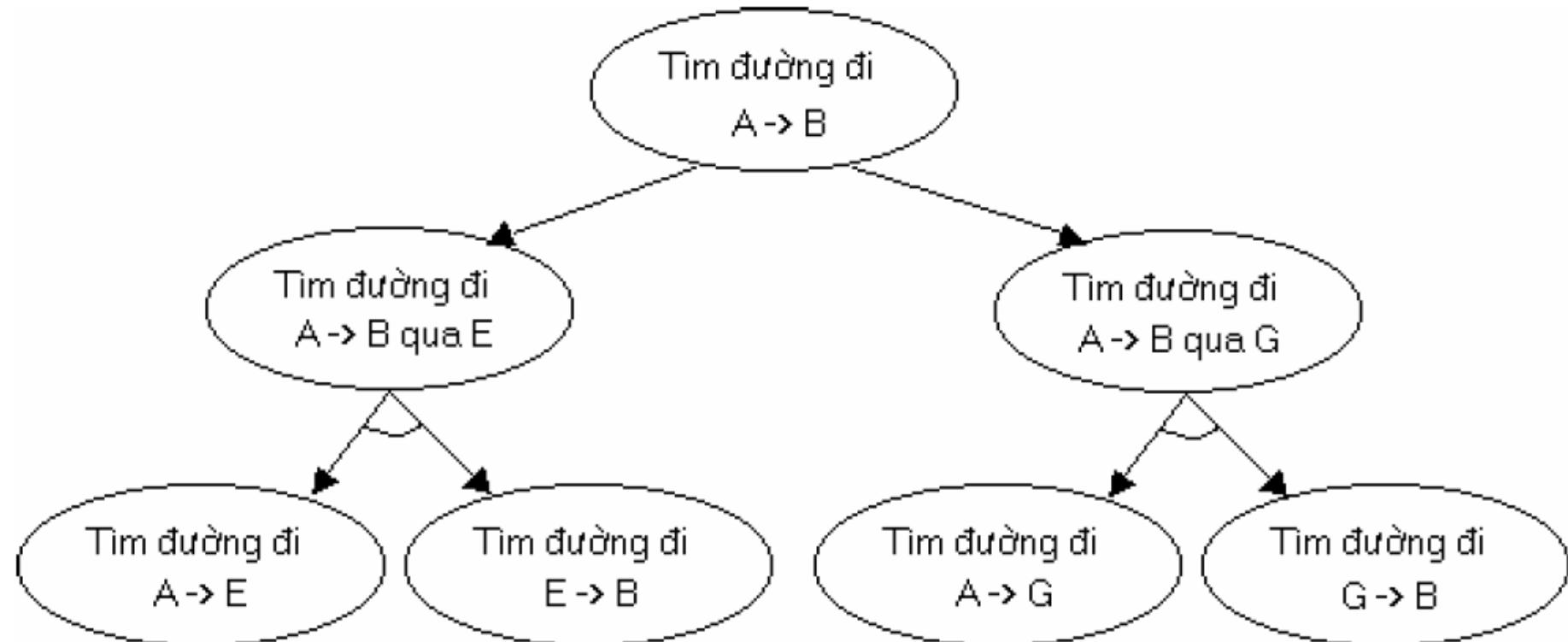


5.6. Ví dụ tìm đường (tiếp)

- Như vậy, bài toán tìm đường từ A đến B qua E có thể quy về các bài toán con:
 - Tìm đường từ A đến E (và),
 - Tìm đường từ E đến B.
- Bài toán tìm đường từ A đến B qua G có thể quy về các bài toán con:
 - Tìm đường từ A đến G (và),
 - Tìm đường từ G đến B.
- Các quá trình trên được minh họa bằng đồ thị (đồ thị và/hoặc) để giải quyết bài toán.

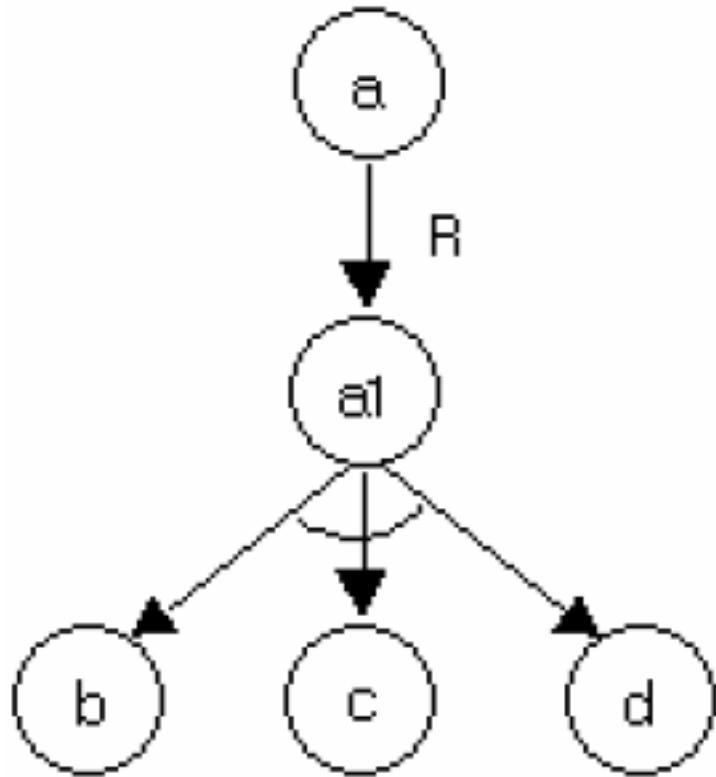
5.6. Đồ thị và/hoặc (and/or graph)

- Ví dụ về đồ thị và/hoặc cho bài toán tìm đường từ A đến B.



5.6. Quy tắc xây dựng đồ thị và/hoặc.

- Mỗi bài toán ứng với một đỉnh của đồ thị.
- Nếu có một toán tử quy một bài toán về một bài toán khác, ví dụ $R: a \rightarrow b$, thì trong đồ thị có cung gán nhãn đi từ đỉnh a tới đỉnh b .
- Đối với mỗi toán tử quy một bài toán về một số bài toán con, ví dụ $R: a \rightarrow b,c,d$, ta đưa một đỉnh mới a_1 , đỉnh này biểu diễn tập các bài toán con $\{b,c,d\}$ và bài toán $R: a \rightarrow b,c,d$ được xây dựng như sau:

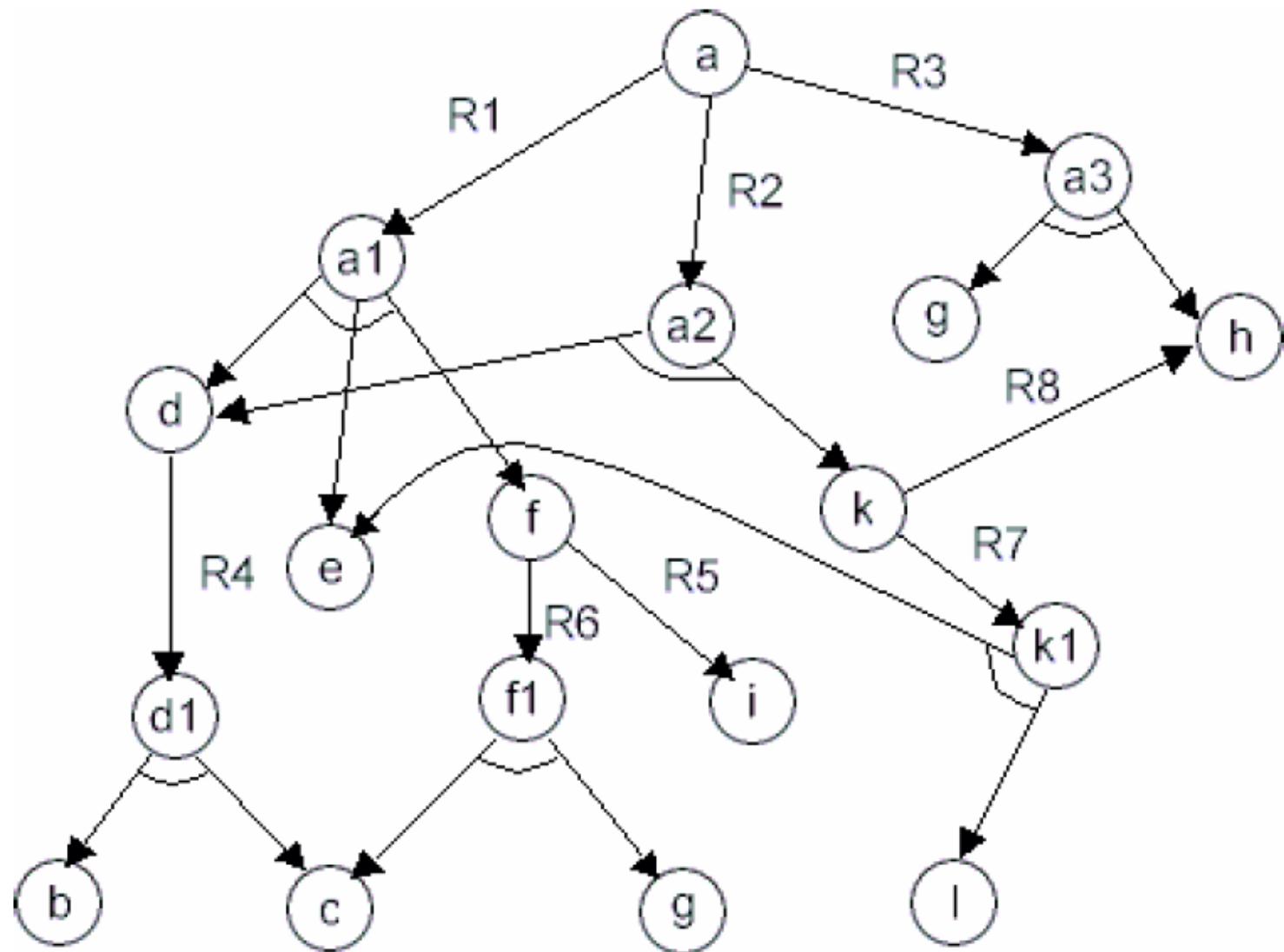


5.6. Ví dụ về đồ thị và/hoặc

Xét bài toán sau:

- Trạng thái ban đầu (bài toán cần giải) là a.
- Tập các toán tử quy gồm:
 - $R_1: a \rightarrow d,e,f$
 - $R_2: a \rightarrow d,k$
 - $R_3: a \rightarrow g,h$
 - $R_4: d \rightarrow b,c$
 - $R_5: f \rightarrow i$
 - $R_6: f \rightarrow c,j$
 - $R_7: k \rightarrow e,l$
 - $R_8: k \rightarrow h$
- Tập các trạng thái kết thúc (các bài toán sơ cấp) là $T=\{b,c,e,j,l\}$

5.6. Ví dụ về đồ thị và/hoặc



5.6. Tìm kiếm trên đồ thị và/hoặc

- Thông thường, sử dụng tìm kiếm theo chiều sâu để tìm lời giải cho bài toán.
- Tìm đến đỉnh u, đỉnh này có thể giải được hay không tùy thuộc nó thuộc lớp bài toán nào. Hàm Solvable sau sẽ trả về TRUE nếu giải được, nếu không là FALSE.

Function Solvable(u);

Begin

If u là đỉnh kết thúc **then** {Solvable(u) \leftarrow **true**; **stop** }

If u không là đỉnh kết thúc và không có đỉnh kè **then** {Solvable(u) \leftarrow **false**; **stop** }

For mỗi toán tử R áp dụng được tại u **do**

{ Ok \leftarrow **true**;

For mỗi v kè u theo R **do**

If Solvable(v) = **false** **then** {Ok \leftarrow **false**; **exit** }

If Ok **then** Solvable(u) \leftarrow **true**; Operator(u) \leftarrow R; **stop**}

Solvable(u) \leftarrow **false**;

End;

5.6. Tìm kiếm trên đồ thị và/hoặc(tiếp)

- Biến Ok: với mỗi toán tử R áp dụng được tại u, biến Ok nhận giá trị **true** nếu tất cả các đỉnh v kề u theo R đều giải được, và Ok nhận giá trị **false** nếu có một đỉnh v kề u theo R không giải được.
- Hàm Operator(u) ghi lại toán tử áp dụng thành công tại u, tức là $\text{Operator}(u) = R$ nếu mọi đỉnh v kề u theo R đều giải được.

5.7. Tóm tắt chương 2

- Để giải quyết vấn đề cần phân tích các đặc trưng và yêu cầu của vấn đề.
- Việc biểu diễn dùng không gian trạng thái giúp biến quá trình giải quyết vấn đề thành một quá trình tìm kiếm (trên không gian trạng thái).
- Các chiến lược tìm kiếm khác nhau: chiều rộng, đều giá, chiều sâu, sâu dần.
- Tìm kiếm sâu dần có độ phức tạp không gian tuyến tính và độ phức tạp thời gian không kém so với tìm kiếm chiều rộng, chiều sâu.

Nhập môn Trí tuệ nhân tạo

Chương 4-1 **Các phương pháp tìm kiếm có sử dụng thông tin**

Biên soạn: TS Ngô Hữu Phúc

Bộ môn Khoa học máy tính

ĐT: 098 56 96 580

eMail: ngohuuphuc76@gmail.com

Thông tin chung

- Thông tin về nhóm môn học:

TT	Họ tên giáo viên	Học hàm	Học vị	Đơn vị công tác (Bộ môn)
1	Ngô Hữu Phúc	GVC	TS	BM Khoa học máy tính
2	Trần Nguyên Ngọc	GVC	TS	BM Khoa học máy tính
3	Hà Chí Trung	GVC	TS	BM Khoa học máy tính
4	Trần Cao Trường	GV	ThS	BM Khoa học máy tính

- Thời gian, địa điểm làm việc: Bộ môn Khoa học máy tính Tầng 2, nhà A1.
- Địa chỉ liên hệ: Bộ môn Khoa học máy tính, khoa Công nghệ thông tin.
- Điện thoại, email: 069-515-329, ngohuuphuc76.mta@gmail.com.

Cấu trúc môn học

- Chương 1: Giới thiệu chung.
- Chương 2: Logic hình thức.
- Chương 3: Các phương pháp tìm kiếm mù.
- Chương 4: Các phương pháp tìm kiếm có sử dụng thông tin.
- Chương 5: Các chiến lược tìm kiếm có đối thủ.
- Chương 6: Các bài toán thỏa ràng buộc.
- Chương 7: Nhập môn học máy.

Bài 4: Các phương pháp tìm kiếm có kinh nghiệm

Chương 4, mục: 4.1 – 4.12

Tiết: 1-3; 4-6; Tuần thứ: 5,6.

Mục đích, yêu cầu:

1. Nắm được phương pháp xây dựng hàm đánh giá.
2. Nắm được một số phương pháp tìm kiếm dựa trên chiều sâu và chiều rộng có sử dụng thông tin.
3. Nắm được các phương pháp tìm kiếm phần tử tốt nhất.
4. Nắm được phương pháp tiếp cận giải thuật GEN.

Hình thức tổ chức dạy học: Lý thuyết.

Thời gian: 6 tiết.

Địa điểm: Giảng đường do Phòng Đào tạo phân công

Nội dung chính: (Slides)

Nội dung

1. Giới thiệu chung;
2. Hàm đánh giá trong tìm kiếm kinh nghiệm;
3. Tìm kiếm tốt nhất đầu tiên (Best-first search);
4. Tìm kiếm ăn tham tốt nhất đầu tiên (Greedy best-first search);
5. Thuật toán leo đồi (Hill-climbing search);
6. Tìm kiếm beam (Beam search);
7. Heuristic chấp nhận được;
8. Tìm kiếm A* (A* search)
9. Tìm kiếm nhánh và cận (Branch and Bound)
10. Các phương pháp tìm kiếm cục bộ (Local search algorithms)
11. Tìm kiếm mô phỏng luyện kim (Simulated annealing search)
12. Thuật toán gen (Genetic algorithms)

1. Giới thiệu chung

- Các kỹ thuật tìm kiếm mù rất kém hiệu quả, trong nhiều trường hợp không sử dụng được.
- Trong chương này, nghiên cứu:
 - Các phương pháp tìm kiếm kinh nghiệm (tìm kiếm heuristic).
 - Các phương pháp sử dụng hàm đánh giá.

2. Hàm đánh giá trong tìm kiếm kinh nghiệm

- Trong tìm kiếm sử dụng kinh nghiệm, với mỗi trạng thái **u**, xác định một hàm **h(u)**, **hàm đánh giá**, hàm này được dùng để đánh giá trạng thái “tốt”, sao cho hy vọng sẽ tới đích tốt nhất.
- Các kỹ thuật này được gọi chung là tìm kiếm kinh nghiệm (heuristic search).
- Các giai đoạn cơ bản của tìm kiếm kinh nghiệm:
 - Tìm biểu diễn thích hợp mô tả các trạng thái và các toán tử.
 - Xây dựng hàm đánh giá,
 - Thiết kế chiến lược chọn trạng thái để phát triển ở mỗi bước.

2.1. Hàm đánh giá

- Trong tìm kiếm có kinh nghiệm, hàm đánh giá đóng vai trò quan trọng.
- Nếu xây dựng hàm đánh giá đúng bản chất vấn đề → tìm kiếm có hiệu quả,
- Ngược lại, xây dựng không tốt có thể đi lệch hướng và tìm kiếm kém hiệu quả.
- Việc xây dựng hàm đánh giá tùy thuộc vào vấn đề cần giải quyết.

2.2. Một số ví dụ về hàm đánh giá

Ví dụ 1:

Trong bài toán tìm kiếm đường đi trên bản đồ, có thể xây dựng hàm đánh giá:

- Đường chim bay từ thành phố này sang thành phố khác, hoặc
- Sử dụng khoảng cách thực trên đường đi giữa các thành phố, hoặc
- Sử dụng cả khoảng cách và một số trọng số khác ảnh hưởng tới việc tìm kiếm (đóng vai trò làm tăng thời gian di chuyển giữa các thành phố),
-

2.2. Một số ví dụ về hàm đánh giá (tiếp)

Ví dụ 2:

Xét bài toán 8 số, ta có thể xây dựng hàm đánh giá như sau:

	2	8
3	5	7
6	4	1

Hàm H_1 :

- với một trạng thái u , $H_1(u)$ là số quân ở sai vị trí.
- trong ví dụ trên: $H_1(u) = 4$

1	2	3
8		4
7	6	5

Hàm H_2 :

- $H_2(u)$ là tổng khoảng cách giữa vị trí quân ở trạng thái u với vị trí ở trạng thái đích.
- trong ví dụ trên: $H_2(u) = 9$

3	2	8
	6	4
7	1	5

3. Tìm kiếm tốt nhất đầu tiên (Best-first search)

- **Ý tưởng:**
Tìm kiếm tốt nhất đầu tiên = Tìm kiếm theo chiều rộng + Hàm đánh giá.
- **Ý nghĩa:** khác với phương pháp tìm kiếm theo chiều rộng, các node không được phát triển lần lượt mà được lựa chọn dựa trên hàm đánh giá (tốt nhất), đỉnh này có thể ở mức hiện tại hoặc ở mức trên.
- **Cài đặt:** Dùng hàng đợi ưu tiên. Sắp xếp các node trong hàng theo thứ tự giảm dần của hàm đánh giá.
- **Một số dạng mở rộng:**
 - Tìm kiếm tham lam tốt nhất đầu tiên (Greedy best-first search).
 - Tìm kiếm A* (A* search).

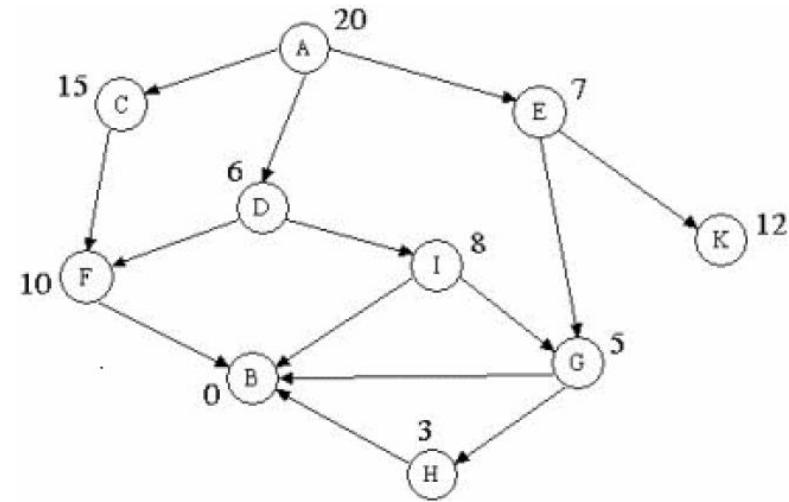
3.1. Ví dụ về Best First Search

Đầu vào:

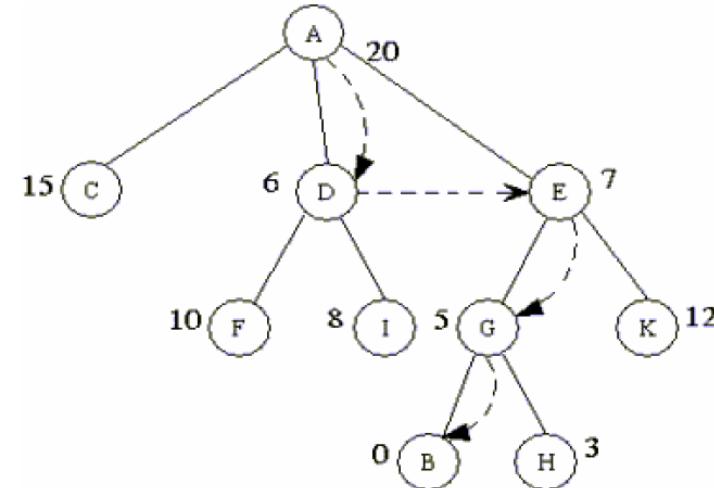
- Trạng thái đầu là A;
- Trạng thái kết thúc là B.

Thực hiện:

- A được xét \rightarrow C, D, E.
- Chọn D, vì $h(D) = 6$ (min), sinh ra F,I.
- Trong số các đỉnh chưa xét C,E,F,I; chọn E vì $h(E) = 7$; sinh ra G và K.
- Chọn G để phát triển; sinh ra B và H.
- B là trạng thái kết thúc.



Xét không gian trạng thái sau



3.2. Cài đặt Best First Search

Procedure Best_First_Search;

Begin

1. Khởi tạo danh sách **L** chỉ chứa trạng thái ban đầu;

2. Loop do

1. **If** L rỗng **then** { thông báo thất bại; **stop**; }
2. Loại trạng thái u ở đầu danh sách L;
3. **If** u là trạng thái kết thúc **then** { thông báo thành công; stop; }
4. **For** mỗi trạng thái v kè u **do**

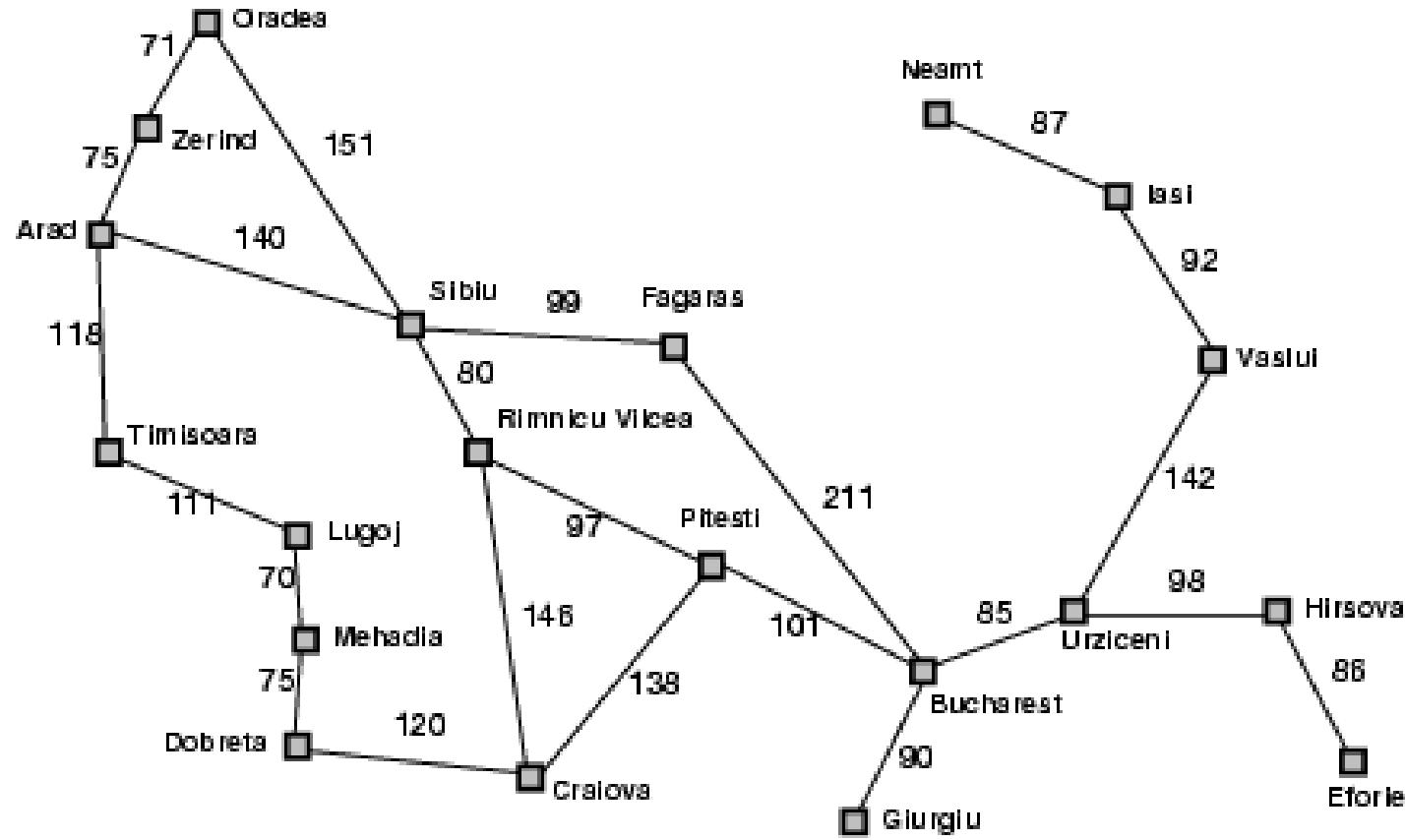
Xen v vào danh sách L sao cho L được sắp theo thứ tự tăng dần của hàm đánh giá;

3. End;

4. Tìm kiếm tham lam tốt nhất đầu tiên (Greedy best-first search)

- **Ý tưởng:**
 - ❖ Sử dụng hàm đánh giá $f(n) = h(n)$ (heuristic);
 - ❖ Hàm $f(n)$: ước lượng giá đến trạng thái đích.
- **Ý nghĩa:**
 - ❖ Khác với phương pháp tìm kiếm tốt nhất đầu tiên, phương pháp này sử dụng hàm đánh giá đến trạng thái đích.
- **Ví dụ:**
 - ❖ Trong tìm kiếm trên bản đồ, hàm $h(n)$ = Khoảng cách theo đường chim bay từ n to thành phố đích.
- **Nhận xét:**
 - ❖ GBFS chọn node “được cho là” gần với node đích để phát triển.

4.1. Tìm đường đi với giá tính theo km



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

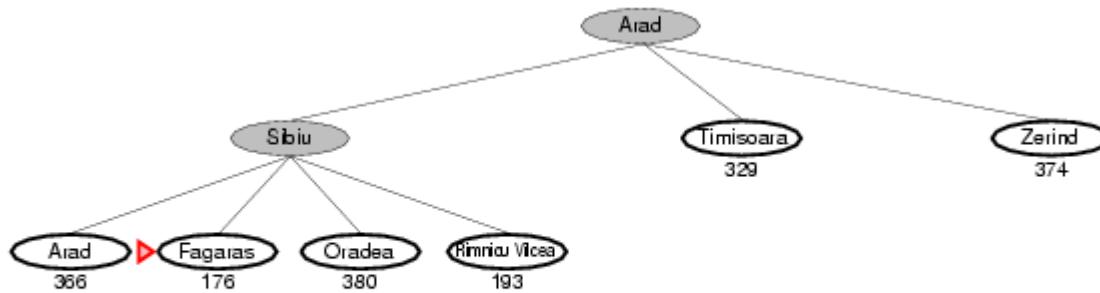
4.1. Ví dụ về GBFS



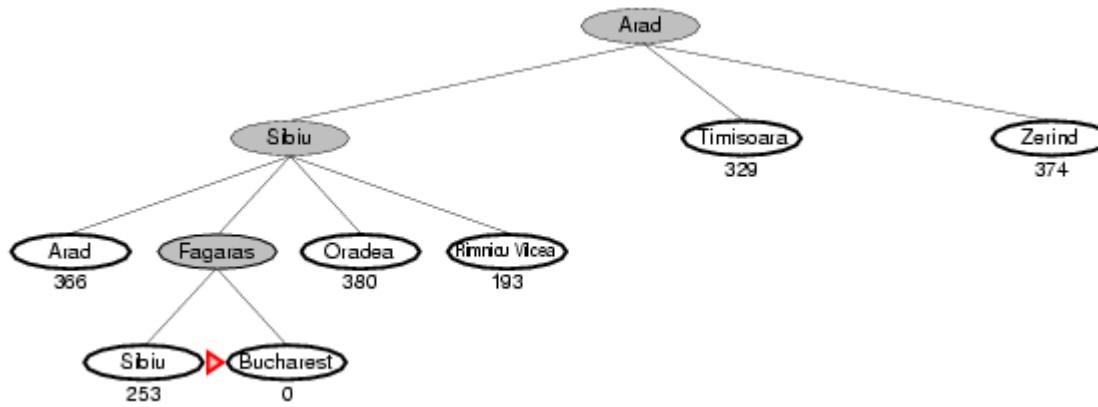
4.1. Ví dụ về GBFS



4.1. Ví dụ về GBFS



4.1. Ví dụ về GBFS



4.2. Đánh giá GBFS

- **Đủ?**

- ❖ Không – Có thể vào vòng lặp quẩn.

- **Độ phức tạp thời gian?**

- ❖ $O(b^m)$ – nếu hàm heuristic xấp xỉ tốt trong thực tế thì thời gian chạy sẽ giảm đi rất nhiều.

- **Độ phức tạp không gian?**

- ❖ $O(b^m)$ – lưu trữ tất cả các nodes.

- **Tối ưu?**

- ❖ Không.

5. Tìm kiếm leo đồi (hill-climbing search)

- **Ý tưởng:**

Tìm kiếm leo đồi = tìm kiếm theo chiều sâu + hàm đánh giá

- **Ý nghĩa:**

- ❖ Phương pháp này được thực hiện nhờ hàm đánh giá.
- ❖ Khác với phương pháp tìm kiếm theo chiều sâu, khi phát triển đỉnh **u**, chọn trong số các đỉnh con của u, đỉnh nào có nhiều hứa hẹn nhất thì phát triển.

- **Cài đặt:**

- ❖ Sử dụng ngăn xếp có ưu tiên.

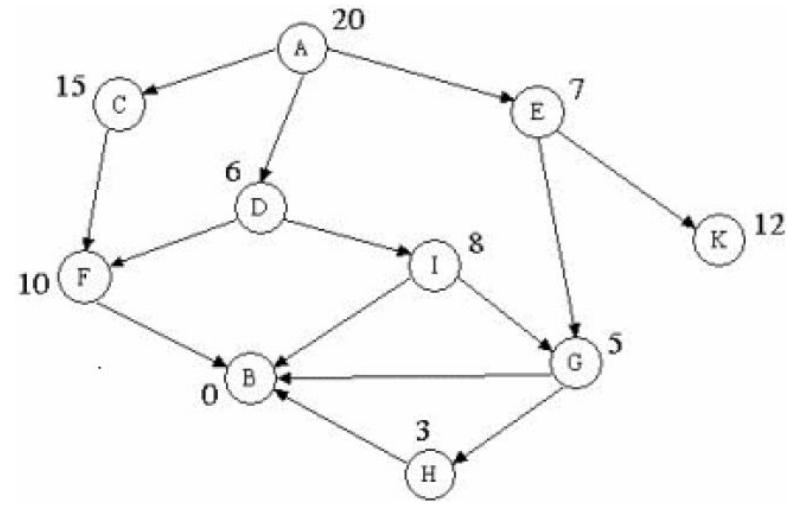
5.1. Ví dụ về Hill-climbing search

Đầu vào:

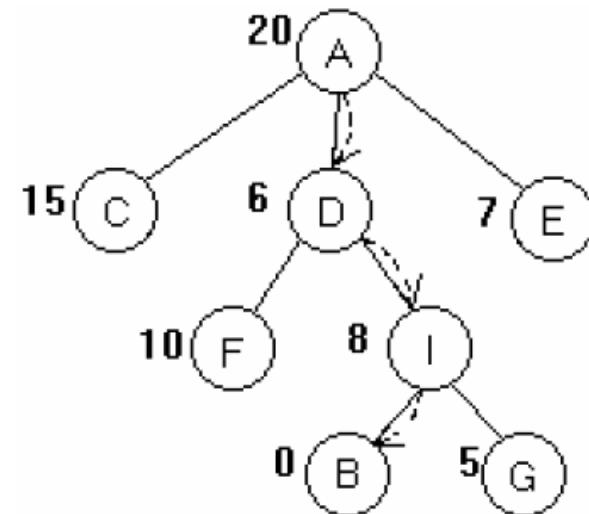
- ❖ Trạng thái đầu là A,
- ❖ Trạng thái kết thúc là B.

Thực hiện:

- ❖ A được xét → C, D, E.
- ❖ Chọn D, vì $h(D) = 6$ (min), sinh ra F,I.
- ❖ Trong số các đỉnh con của D, chọn I, vì $h(I) = 8$.
- ❖ Với I được chọn, sinh ra B và G.
- ❖ B là trạng thái kết thúc.



Xét không gian trạng thái sau



5.2. Cài đặt Hill-Climbing Search

Procedure Hill_Climbing_Search;

Begin

1. Khởi tạo danh sách **L** chỉ chứa trạng thái đầu;

2. **Loop do**

1. **If** L rỗng **then** { thông báo thất bại; **stop**; }

2. Loại trạng thái **u** đầu danh sách **L**;

3. **If** u là trạng thái kết thúc **then** { thông báo thành công; **stop**; }

4. **For** mỗi trạng thái **v** kề **u** đặt **v** vào **L** sao cho các phần tử được đưa vào đầu danh sách **L** có đánh giá giảm dần;

3. **End**;

6. Tìm kiếm chùm (Beam search)

- **Ý tưởng:**

Tìm kiếm theo chiều rộng + k node để phát triển + hàm đánh giá

- **Ý nghĩa:**

- ❖ Tìm kiếm beam giống tìm kiếm theo chiều rộng,
- ❖ Tuy nhiên trong tìm kiếm beam, hạn chế **k** đỉnh tốt nhất để phát triển.
- ❖ Như vậy, số đỉnh cần phát triển ở mức **d** là **k^d** .

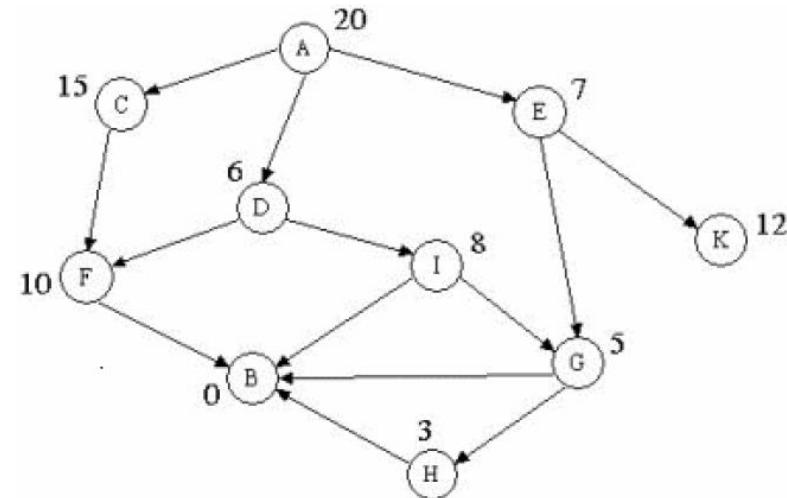
6.1. Ví dụ về Beam Search

Đầu vào:

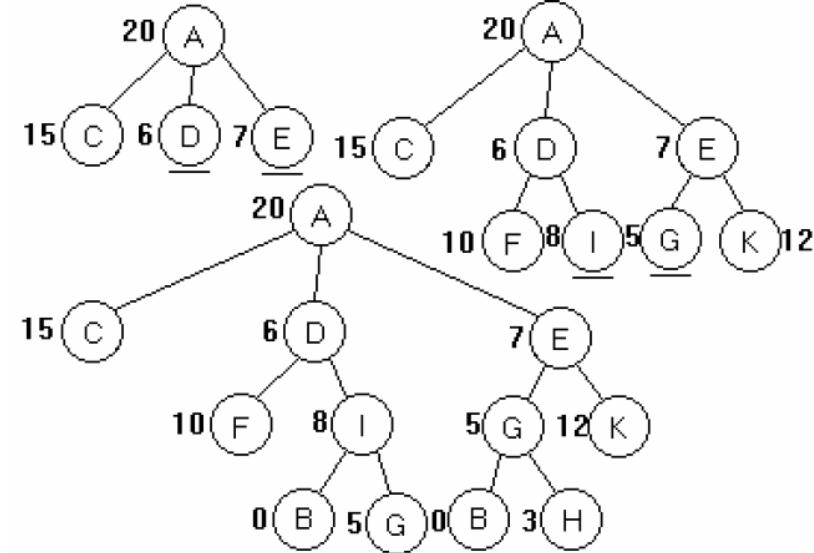
- ❖ Trạng thái đầu là A,
- ❖ Trạng thái kết thúc là B.

Thực hiện:

- Ví dụ lấy $k = 2$.
- A được xét \rightarrow C, D, E.
- Chọn D và E để phát triển, với D sinh ra F và I, với E sinh ra G và K.
- Chọn I và G để phát triển, sinh ra B, G, B, H.
- Chọn được B (qua I) và B (qua G).
- B là trạng thái kết thúc.



Xét không gian trạng thái sau



7. Heuristic chấp nhận được

Trong kỹ thuật tìm kiếm, để việc tìm kiếm có hiệu quả sẽ sử dụng hàm đánh giá để hướng dẫn tìm kiếm. Các kỹ thuật này thuộc nhóm tìm kiếm Heuristic.

- Giả sử u là một trạng thái đạt tới (có đường đi từ trạng thái đầu u_0 tới u); hàm đánh giá được xác định như sau:

➢ $g(u)$: đánh giá độ dài đường đi ngắn nhất từ u_0 tới u .

➢ $h(u)$: đánh giá độ dài đường đi ngắn nhất từ u tới trạng thái đích.

Hàm $h(u)$ được gọi là **chấp nhận được** nếu với mọi trạng thái u , thì

$h(u) \leq$ độ dài đường đi ngắn nhất thực tế từ u tới trạng thái đích.

➢ Để tăng hiệu quả của quá trình tìm kiếm:

$$f(u) = g(u) + h(u)$$

7.1. Heuristics chấp nhận được trong A*

- Hàm **heuristic $h(u)$** là chấp nhận được nếu với mọi node **u**, $h(u) \leq h^*(u)$,
 - ❖ Trong đó **$h^*(u)$** là chi phí thực để đi đến đích từ **u**.
- **Ý nghĩa:**
 - ❖ Heuristic chấp nhận được không bao giờ đánh giá chi phí cao quá thực tế.
- **Ví dụ:** $h_{SLD}(u)$ là Heuristic chấp nhận được.
- **Định lý:**
 - ❖ **Nếu $h(n)$ là chấp nhận được, A* là thuật toán cho lời giải tối ưu.**

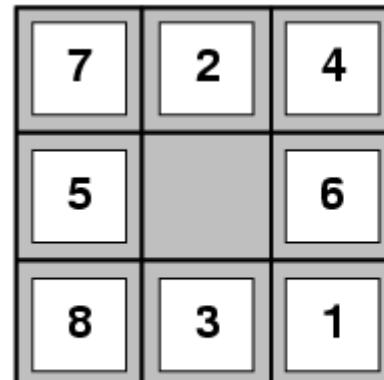
7.1. Ví dụ về heuristics chấp nhận được

Ví dụ, bài toán 8-số:

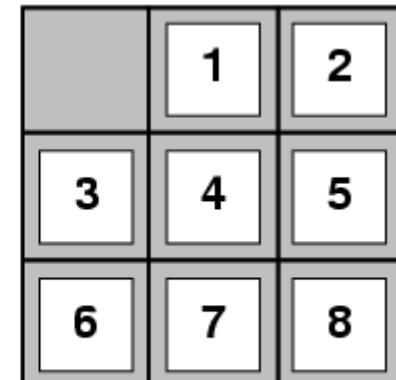
- $h_1(u)$ = Số lượng ô sai vị trí.
- $h_2(u)$ = Tổng khoảng cách theo Manhattan Metric

(i.e., Số lượng ô từ ô hiện tại đến vị trí mong muốn)

- $h_1(S) = ?$
- $h_2(S) = ?$



Start State



Goal State

7.1. Ví dụ về heuristics chấp nhận được (tiếp)

Ví dụ, bài toán 8-số:

- $h_1(u)$ = Số lượng ô sai vị trí.
- $h_2(u)$ = Tổng khoảng cách theo Manhattan Metric

(i.e., Số lượng ô từ ô hiện tại đến vị trí mong muốn)

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

- $h_1(S) = 8$

- $h_2(S) = 3+1+2+2+2+3+3+2 = 18$

7.2. So Sánh các Heuristics

- Nếu $h_2(u) \geq h_1(u)$ với mọi u (cả hai đều chấp nhận được)
 - ❖ thì h_2 được coi là mạnh hơn h_1
 - ❖ h_2 là heuristic tốt hơn
- Ví dụ tìm kiếm trên bài toán 8-số (số lượng node trung bình phải xét):
 - $d=12$ IDS = 3,644,035 nodes
 $A^*(h_1)$ = 227 nodes
 $A^*(h_2)$ = 73 nodes
 - $d=24$ IDS = too many nodes
 $A^*(h_1)$ = 39,135 nodes
 $A^*(h_2)$ = 1,641 nodes

7.3. Nói lỏng ràng buộc của bài toán

- Bài toán có thể nói lỏng bằng cách bớt các ràng buộc trên toán tử;
- Chi phí cho lời giải tối ưu của bài toán nói lỏng là một Heuristic chấp nhận được đối với bài toán gốc;
- Ví dụ:
 - Nếu bài toán 8-số được nói lỏng sao cho có thể dịch chuyển mỗi ô đến nơi tuỳ ý, ta có **$h_1(n)$** cho lời giải tối ưu.
 - Nếu bài toán được nói lỏng sao cho mỗi ô có thể dịch chuyển đến ô bất kỳ liền kề thì ta có **$h_2(n)$** cho lời giải tối ưu.

8. A* search

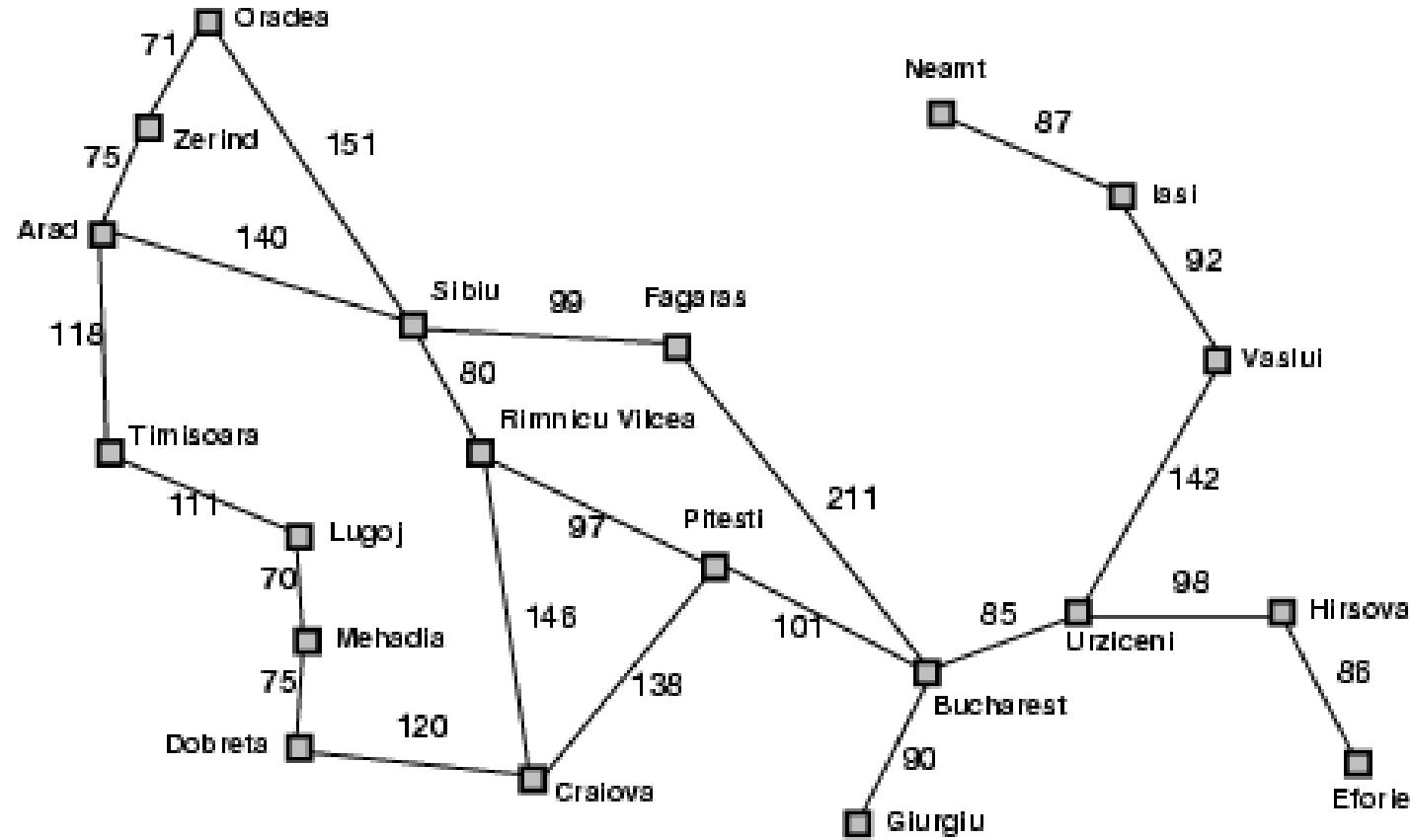
- **Ý tưởng:**

- ❖ Sử dụng hàm Heuristics chấp nhận được + tìm kiếm theo chiều rộng → Loại bỏ những đường đi có chi phí cao.

- **Hàm lượng giá: $f(u) = g(u) + h(u)$**

- ❖ $g(u)$ = Chi phí để đến u
- ❖ $h(u)$ = Lượng giá từ u đến đích
- ❖ $f(u)$ = Ước lượng tổng giá đến đích qua u .

8.1. Tìm đường đi với giá tính theo km

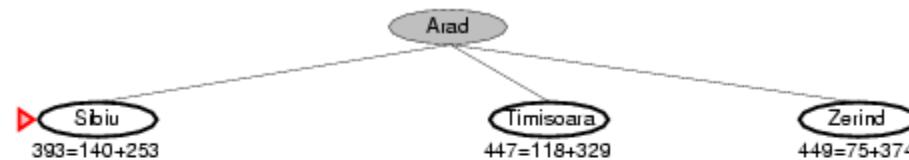


Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Lasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

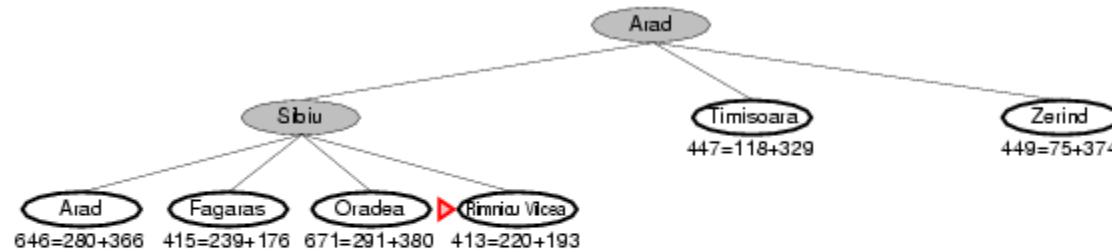
8.1. Ví dụ về A* search



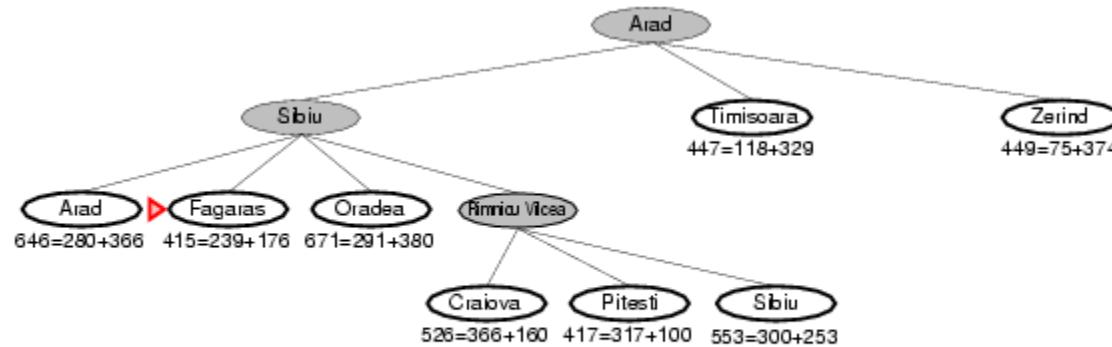
8.1. Ví dụ về A* search



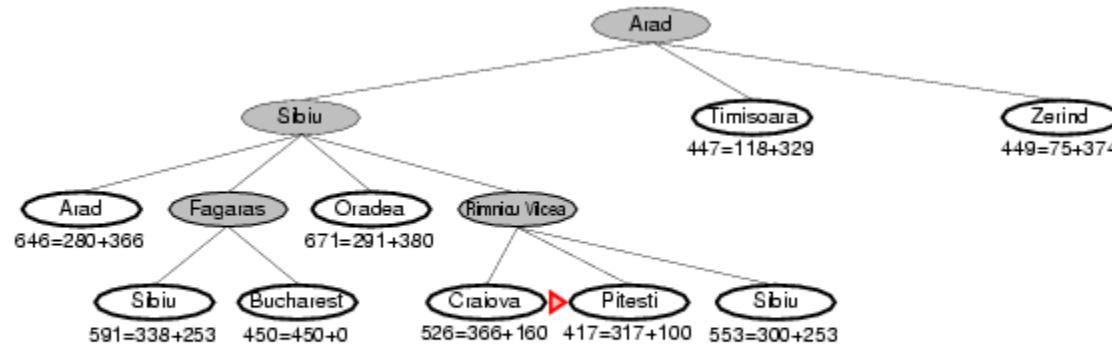
8.1. Ví dụ về A* search



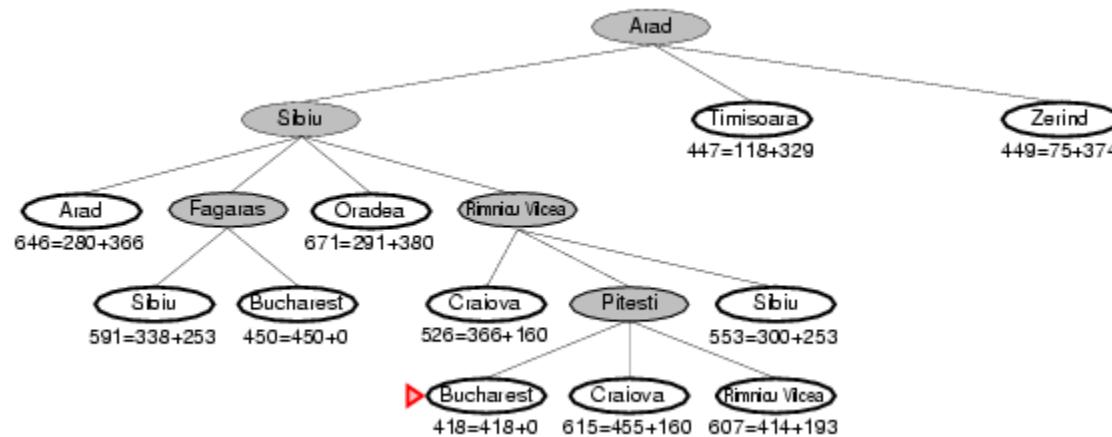
8.1. Ví dụ về A* search



8.1. Ví dụ về A* search



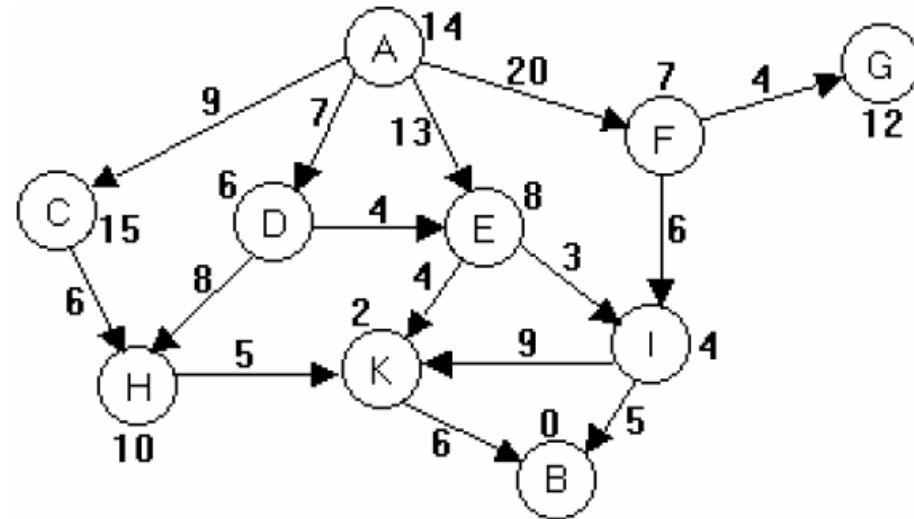
8.1. Ví dụ về A* search



8.1. Ví dụ về A* search (ví dụ 2)

Đầu vào:

- Trạng thái đầu A,
- Trạng thái đích B.
- Các giá trị ghi trên cạnh là độ dài đường đi;
- Các số cạnh các đỉnh là giá trị hàm h.



Yêu cầu:

- Tìm đường đi ngắn nhất từ A đến B bằng A*.

Không gian trạng thái với hàm đánh giá

8.1. Ví dụ về A* search (ví dụ 2)

Thực hiện:

- ❖ Phát triển đỉnh A sinh ra các đỉnh con C, D, E, F.
 - ❖ $g(C) = 9, h(C) = 15 \rightarrow f(C) = 9 + 15 = 24$
 - ❖ $g(D) = 7, h(D) = 6 \rightarrow f(D) = 7 + 6 = 13$
 - ❖ $g(E) = 13, h(E) = 8 \rightarrow f(E) = 13 + 8 = 21$
 - ❖ $g(F) = 20, h(F) = 7 \rightarrow f(F) = 20 + 7 = 27$
- Như vậy, đỉnh D được chọn để phát triển ($f(D) = 13$)

8.1. Ví dụ về A* search (ví dụ 2)

- Phát triển D, nhận được các đỉnh con H và E (mới). Trong đó:

➢ $g(H) = g(D) + \text{độ dài cung } (D,H) = 7 + 8 = 15$

➢ $h(H) = 10$

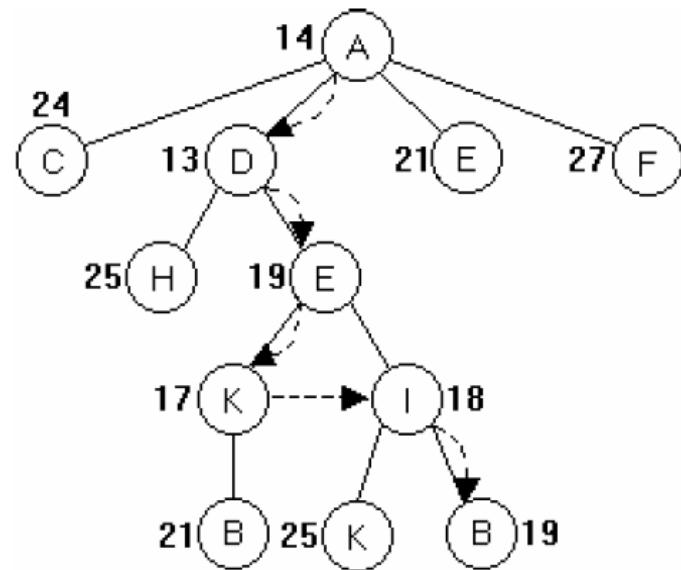
➢ $\rightarrow f(H) = g(H) + h(H) = 15 + 10 = 25.$

➢ $g(E) = g(D) + \text{độ dài cung } (D,E) = 7 + 4 = 11$

➢ $h(E) = 8$

$\rightarrow f(E) = g(E) + h(E) = 11 + 8 = 19.$

- Như vậy đỉnh E sẽ được dùng để phát triển tiếp
- Tương tự sẽ chọn được các đỉnh K, B (đích).
- Với $g(B) = 19$, $h(B) = 0$.
- Đường đi: A → D → E → I → B



8.2. Cài đặt thuật toán A*

Procedure A*;

Begin

1. Khởi tạo danh sách **L** chỉ chứa trạng thái đầu.

2. **Loop do**

2.1. **if** L rỗng **then** {thông báo thất bại; stop;}

2.2. Loại trạng thái **u** ở đầu danh sách **L**;

2.3. **if** **u** là trạng thái đích **then** {thông báo thành công; stop}

2.4. **for** mỗi trạng thái **v** kè **u** **do**

{ **g(v) ← g(u)+ k(u,v);**

f(v) ← g(v)+h(v);

Đặt v vào danh sách L;

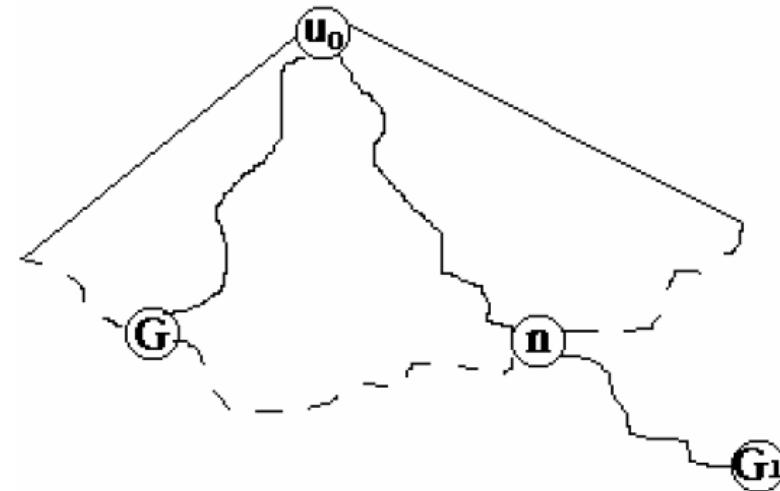
}

2.5. Sắp xếp **L** theo thứ tự giảm dần của hàm **f** sao cho trạng thái có giá trị của hàm **f** nhỏ nhất ở đầu danh sách;

3. **End;**

8.3. Chứng minh tính tối ưu của A*

- Giả sử thuật toán dừng ở G , với độ dài đường đi là $g(G)$ và ta có $h(G)=0$ nên $f(G)=g(G)$.
- Giả sử nghiệm tối ưu không phải là G , tối ưu là G_1 , với độ dài đường đi là S .
- Như vậy, đường đi này bị tách ra ở vị trí N nào đó. Khi đó có 2 khả năng:
 - N trùng G_1 , hoặc
 - N không trùng G_1 .



8.3. Chứng minh tính tối ưu của A* (tiếp)

➤ Nếu $N \equiv G_1$:

➤ G được chọn trước G_1 , nên

$$f(G) \leq f(G_1) \text{ vì } g(G) \leq g(G_1) = S$$

➤ Nếu $N \neq G_1$:

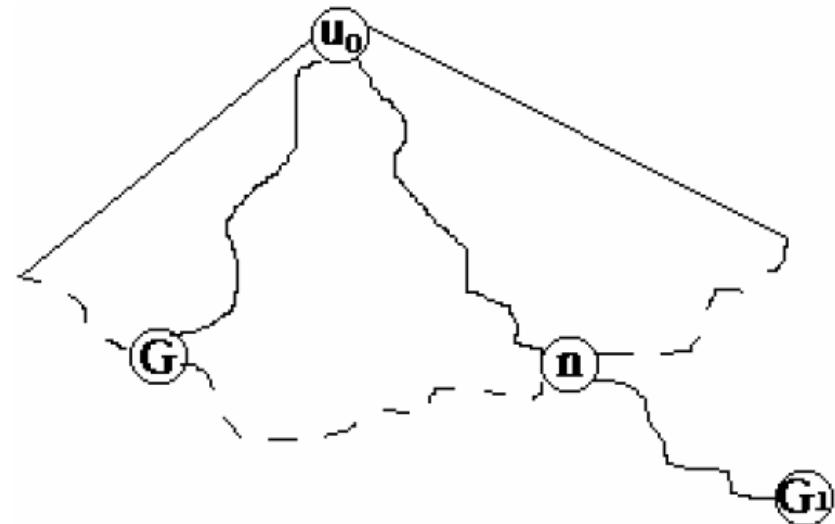
➤ Do $h(u)$ là đánh giá thấp \rightarrow

$$f(N) = g(N) + h(N) \leq S.$$

➤ Do G được chọn trước $\rightarrow f(G)$

$$\leq f(N) \leq S.$$

➤ Vậy G là đường đi tối ưu.



8.4. Nhận xét về A*

- Nếu hàm $h(u)$ đánh giá thấp nhất thì thuật toán A* là tối ưu.
- Nếu các cung không nhỏ hơn một số α nào đó thì A* là đầy đủ.
- Nếu $h(u)=0$ với mọi u , thuật toán A* chính là thuật toán tìm kiếm tốt nhất đầu tiên với hàm đánh giá $g(u)$.

8.5. Phân tích A*

- **Đủ?**
 - Có (Trừ phi có vô hạn node với $f \leq f(G)$).
- **Độ phức tạp thời gian?**
 - Hàm mũ
- **Không gian?**
 - Lưu trữ tất cả các node
- **Tối ưu?** Có

9. TÌM KIẾM NHÁNH VÀ CẬN (Branch and Bound)

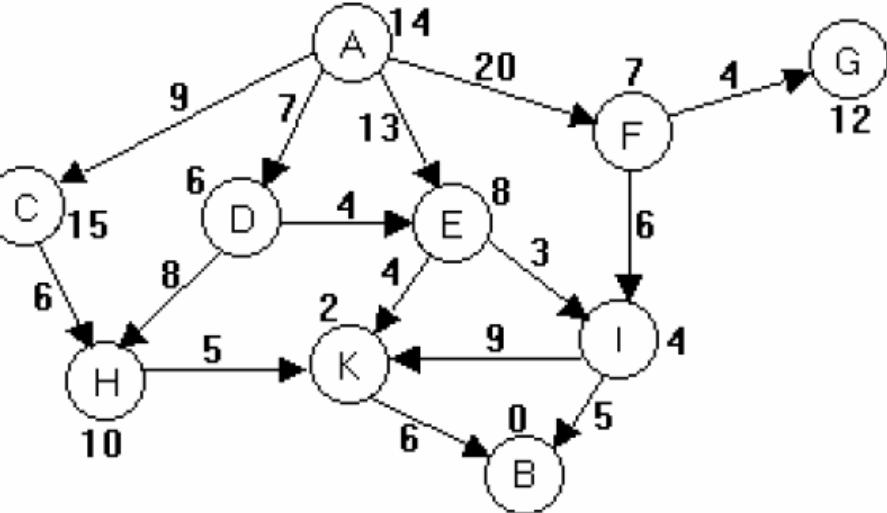
Ý TƯỞNG:

- Thuật toán nhánh và cận sử dụng tìm kiếm leo đồi với hàm đánh giá $f(u)$.
- Tại trạng thái u , chọn trạng thái v trong số các trạng thái kề với u , với $f(v)$ đạt min.
- Tương tự cho đến khi:
 - v là đích, hoặc
 - v không có đỉnh kề, hoặc
 - v có $f(v)$ lớn hơn độ dài đường đi tối ưu hiện thời.
 - Không phát triển v nữa, quay về cha của v để tìm trạng thái tốt nhất trong các trạng thái còn lại chưa xét.

9.1. Ví dụ về nhánh và cận

Xét không gian trạng thái bên.

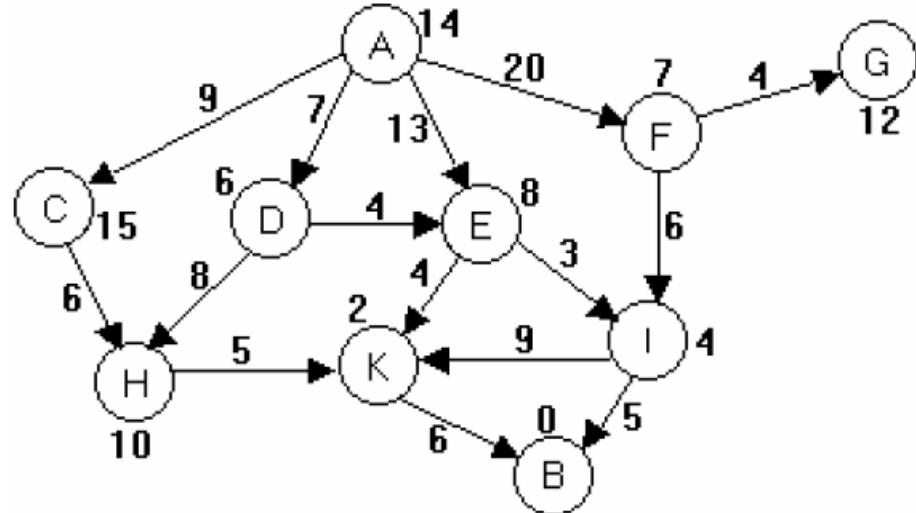
- Phát triển A, có các đỉnh con C, D, E, F với $f(C) = 24$; $f(D) = 13$; $f(E) = 21$; $f(F) = 27$.
- Chọn D, sinh các con H, E (mới) với $f(H) = 25$; $f(E) = 19$.
- Chọn E, sinh ra K, I với $f(K) = 17$; $f(I) = 18$.
- Chọn K, sinh ra B với $f(B) = g(B) = 21 \rightarrow$ **đường đi tạm thời là 21.**



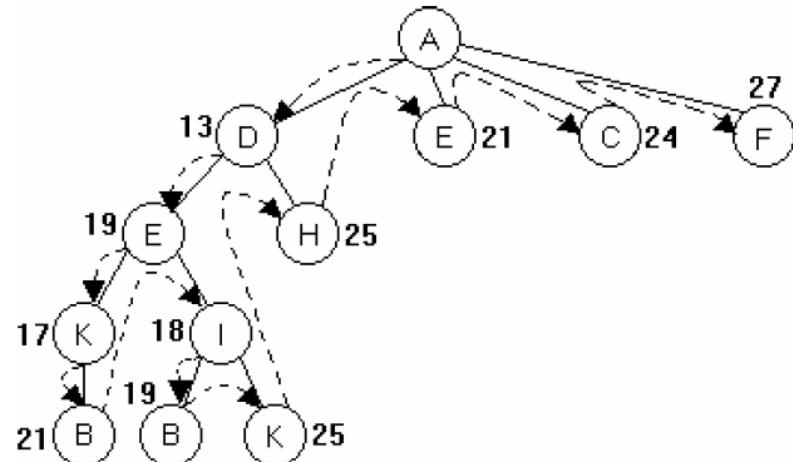
Không gian trạng thái với hàm đánh giá

9.1. Ví dụ về nhánh và cận (tiếp)

- Từ B, quay về K. Từ K quay về E.
- Từ E, sang I, $f(I) = 18 <$ độ dài tạm thời 21. Sinh ra K, B với $f(K) = 25$, $f(B) = g(B) = 19 \rightarrow$ **đường đi tạm thời là 19.**
- Với B, không tìm được điểm nào có chi phí tốt hơn nữa.
- Vậy đường đi tối ưu có độ dài 19.**



Không gian trạng thái với hàm đánh giá



9.2. Cài đặt thuật toán nhánh và cận

Procedure Branch_and_Bound;

Begin

1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu;
Gán giá trị ban đầu cho cost;

2. Loop do

- 2.1. **If** L rỗng **then stop**;
- 2.2. Loại trạng thái u ở đầu danh sách L;
- 2.3. **If** u là trạng thái kết thúc **then**
if $g(u) \leq cost$ **then** { $cost \leftarrow g(u)$; quay lại 2.1; }
- 2.4. **If** $f(u) > cost$ **then** quay về 2.1;
- 2.5. **For** mỗi trạng thái v kề u **do**
{ $g(v) \leftarrow g(u)+k(u,v)$; $f(v) \leftarrow g(v)+h(v)$; đặt v vào danh sách L1}
- 2.6. Sắp xếp L1 theo thứ tự tăng của hàm f;
- 2.7. Chuyển L1 vào đầu danh sách L sao cho L trạng thái đầu của L1 vào đầu L;

3. End;

9.3. Nhận xét thuật toán nhánh và cận

- Thuật toán nhánh và cận cũng là thuật toán đầy đủ và tối ưu nếu hàm đánh giá $h(u)$:
 - Đánh giá thấp và
 - Độ dài các cung không nhỏ hơn một số dương α nào đó.

10. Tìm đối tượng tốt nhất

- Khác với quá trình tìm kiếm trước, tìm đối tượng tốt nhất x , $x \in U$, chưa xác định được đích của quá trình tìm kiếm. Một số kỹ thuật được sử dụng trong phần này gồm:
 - Kỹ thuật tìm kiếm leo đồi để tìm đối tượng tốt nhất.
 - Kỹ thuật tìm kiếm gradient (gradient search).
 - Kỹ thuật tìm kiếm mô phỏng luyện kim (simulated annealing)

10.1. Kỹ thuật tìm kiếm leo đồi tìm đối tượng tốt nhất

- Kỹ thuật này không khác nhiều so với thuật toán tìm kiếm leo đồi trước.
- Với thuật toán leo đồi, từ trạng thái **u** chuyển sang trạng thái tốt nhất **v** (theo hàm lượng giá). Nếu không đến đích và không “leo” được nữa, “quay về” trạng thái trước và leo lên trạng thái tốt nhất còn lại.
- Với kỹ thuật tìm kiếm leo đồi tìm đối tượng tốt nhất, từ trạng thái **u**, chuyển sang trạng thái **v** tốt hơn. Nếu không tìm được thì dùng thuật toán.

10.1.1. Cài đặt thuật toán HC

- Trong thủ tục, **u** là đỉnh hiện thời, **v** là trạng thái tốt nhất của lân cận **u**.

Procedure Hill_Climbing;

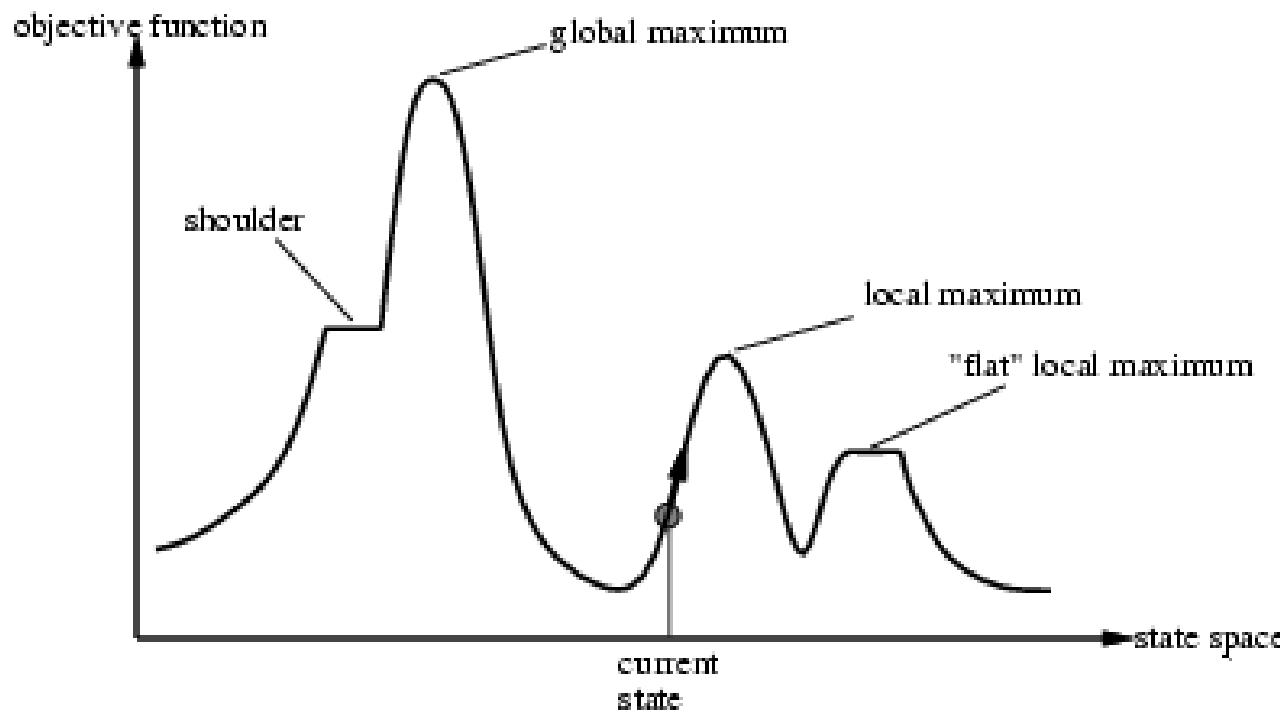
Begin

1. $u \leftarrow$ một đối tượng ban đầu nào đó;
2. **Loop do** với v là con tốt nhất của u
if $\text{cost}(v) > \text{cost}(u)$ **then** $u \leftarrow v$ **else stop**;

End;

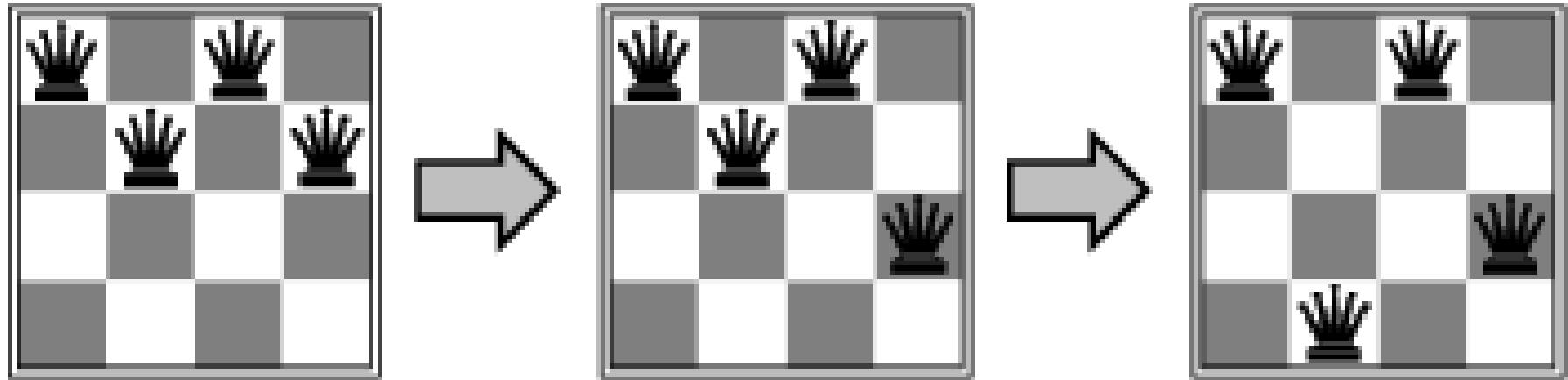
10.1.2. Nhận xét về Hill-Climbing

- Yếu điểm: phụ thuộc vào trạng thái khởi đầu, có thể bị kẹt tại cực trị địa phương. (Vấn đề ridge, valley).



- Khắc phục: Có thể lặp lại quá trình leo đồi với nhiều điểm xuất phát khác nhau

10.1.3. Ví dụ: *n*-queens

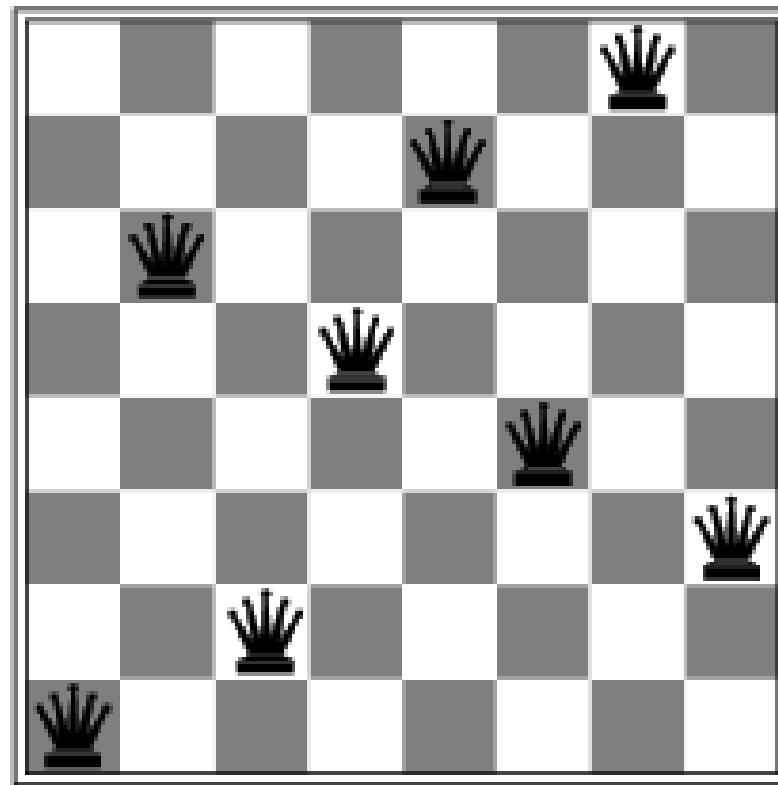


10.1.3. Ví dụ bài toán 8-queens

18	12	14	13	13	12	14	14
14	16	13	15	12	14	12	16
14	12	18	13	15	12	14	14
15	14	14	14	15	13	16	13
15	14	14	14	15	13	16	16
14	14	17	15	15	14	16	16
17	15	16	18	15	15	15	15
18	14	15	15	15	14	15	16
14	14	13	17	12	14	12	18

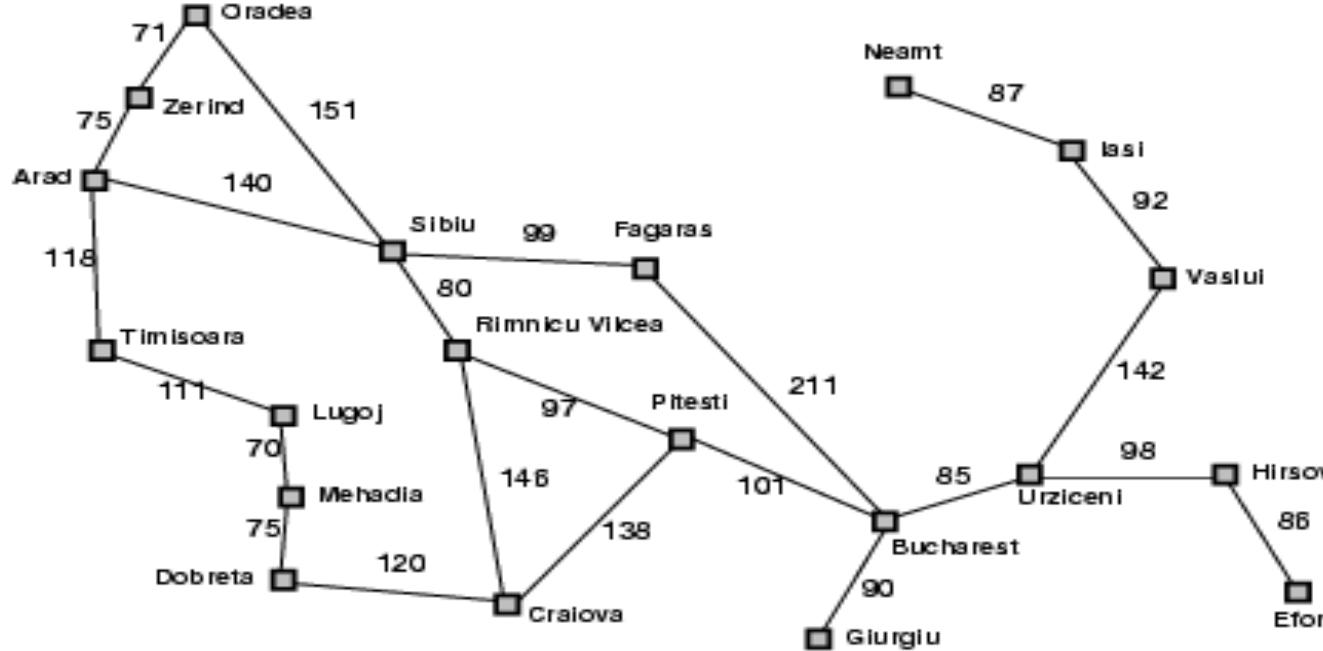
- $h = \text{số lượng con hậu tấn công lẫn nhau}$ (trực tiếp và gián tiếp)
- $h = 17$ trong hình trên

10.1.3. Ví dụ bài toán 8-queens



- Một cực trị địa phương với $h = 1$.

10.1.3. Bài toán người du lịch



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

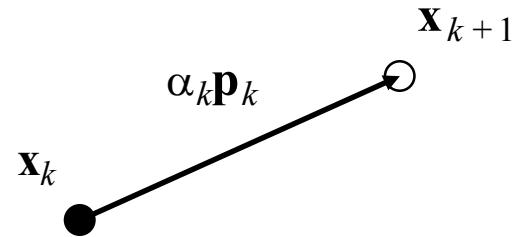
- biểu diễn: **dãy hoán vị.**
- $h =$ **tổng giá chi phí đường di trên dãy hoán vị.**
- Toán tử: **đổi chỗ hai đỉnh.**

10.2. Tìm kiếm địa phương trong tối ưu không ràng buộc

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$$

hoặc là

$$\Delta \mathbf{x}_k = (\mathbf{x}_{k+1} - \mathbf{x}_k) = \alpha_k \mathbf{p}_k$$



\mathbf{p}_k - hướng tìm kiếm

α_k - hệ số học

10.2.1. Tìm kiếm Gradient

- **Ý tưởng:** Tìm kiếm gradient là kỹ thuật tìm kiếm leo đồi để tìm giá trị lớn nhất (hoặc nhỏ nhất) của **hàm khả vi liên tục $f(x)$** trong không gian các vector thực n-chiều.
- Trong lân cận đủ nhỏ của điểm $x = (x_1, x_2, \dots, x_n)$, hàm f tăng nhanh nhất theo hướng gradient, với hướng được xác định:

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right)$$

10.2.2. Thuật toán giảm gradient

Chọn bước tiếp theo sao cho:

$$F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k)$$

với thay đổi nhỏ của \mathbf{x} ta có thể xấp xỉ hàm $F(\mathbf{x})$:

$$F(\mathbf{x}_{k+1}) = F(\mathbf{x}_k + \Delta\mathbf{x}_k) \approx F(\mathbf{x}_k) + \mathbf{g}_k^T \Delta\mathbf{x}_k$$

trong đó

$$\mathbf{g}_k \equiv \nabla F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}_k}$$

Muốn hàm giảm thì:

$$\mathbf{g}_k^T \Delta\mathbf{x}_k = \alpha_k \mathbf{g}_k^T \mathbf{p}_k < 0$$

Giảm nhiều nhất:

$$\mathbf{p}_k = -\mathbf{g}_k$$

$$\boxed{\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k}$$

10.2.3. Cài đặt tìm kiếm Gradient

Procedure Gradient_Search;

Begin

$x \leftarrow$ điểm xuất phát nào đó;

repeat

$x \leftarrow x + \alpha \cdot \nabla f(x);$

until $|\nabla f(x)| < \varepsilon$

End;

Với α nhỏ, xác định bước chuyển, ε xác định tiêu chuẩn dừng.

10.2.4. Ví dụ về tìm kiếm gradient

$$F(\mathbf{x}) = x_1^2 + 2x_1x_2 + 2x_2^2 + x_1$$

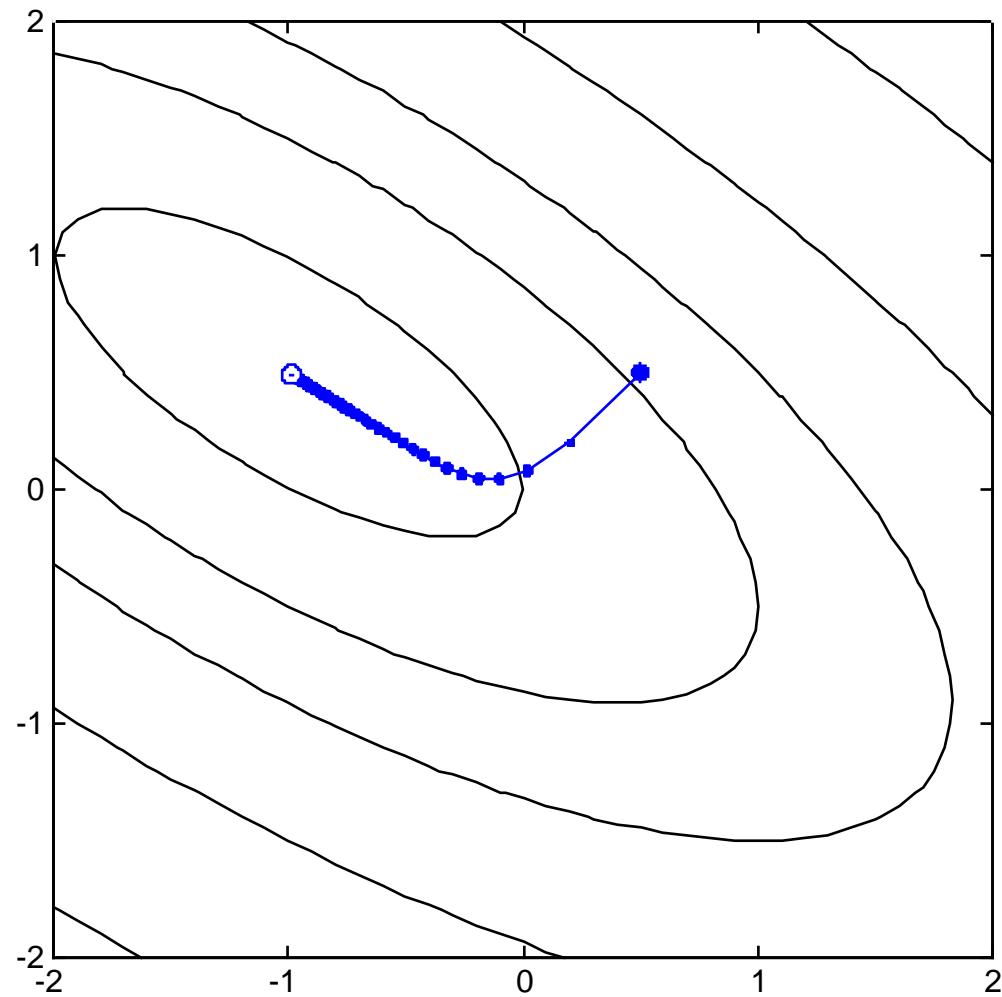
$$\mathbf{x}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \quad \alpha = 0.1$$

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 2x_1 + 2x_2 + 1 \\ 2x_1 + 4x_2 \end{bmatrix} \quad \mathbf{g}_0 = \nabla F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}_0} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$\mathbf{x}_1 = \mathbf{x}_0 - \alpha \mathbf{g}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - 0.1 \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix}$$

$$\mathbf{x}_2 = \mathbf{x}_1 - \alpha \mathbf{g}_1 = \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix} - 0.1 \begin{bmatrix} 1.8 \\ 1.2 \end{bmatrix} = \begin{bmatrix} 0.02 \\ 0.08 \end{bmatrix}$$

10.2.5. Hình minh họa



10.3. Tìm kiếm mô phỏng luyện kim

Ý tưởng:

- Với phương pháp tìm kiếm leo đồi không đảm bảo cho việc tìm kiếm nghiêm tối ưu toàn cục. Để có được nghiêm tối ưu toàn cục, kỹ thuật leo đồi sử dụng việc xuất phát từ lựa chọn ngẫu nhiên, lặp nào đó.
- Tư tưởng chính của mô phỏng luyện kim là cho phép chọn cả lựa chọn “xấu” với xác suất nào đó.

10.3.1. Mô tả tìm kiếm mô phỏng luyện kim

- Giả sử đang ở trạng thái **u** nào đó. Chọn ngẫu nhiên trạng thái **v** trong lân cận **u**.
 - Nếu **v** tốt hơn **u**: sang **v**.
 - Nếu **v** không tốt hơn **u**: chỉ sang **v** với xác suất nào đó.
- Xác suất này giảm theo hàm mũ của “độ tồi” của trạng thái **v**. Xác suất phụ thuộc vào tham số **T** (nhiệt độ), **T** càng lớn khả năng sang trạng thái tồi càng cao.
- Xác suất được xác định:

$$e^{\frac{\Delta}{T}}$$

$$\text{với } \Delta = \text{cost}(v) - \text{cost}(u)$$

10.3.2. Cài đặt thuật toán SA

Procedure Simulated_Annealing;

Begin

$t \leftarrow 0;$

$u \leftarrow$ trạng thái ban đầu nào đó;

$T \leftarrow$ nhiệt độ ban đầu;

repeat

$v \leftarrow$ trạng thái được chọn ngẫu nhiên trong lân cận u ;

if $\text{cost}(v) > \text{cost}(u)$ **then** $u \leftarrow v$;

else $u \leftarrow v$ với xác suất $e^{\Delta/T}$;

$T \leftarrow g(T,t);$

$t \leftarrow t + 1;$

until T đủ nhỏ;

End;

Chú ý:

$g(T,t)$ thỏa mãn

điều kiện

$g(T,t) < T$

với mọi t .

Hàm này xác định tốc độ giảm nhiệt độ.

10.3.3. Nhận xét về SA

- Đã chứng minh được bằng lý thuyết, nếu T giảm đủ chậm thì thuật toán sẽ tìm được **nghiệm tối ưu toàn cục**.
- Thuật toán mô phỏng luyện kim (SA) đã áp dụng thành công cho các **bài toán tối ưu cỡ lớn**.

Nhập môn Trí tuệ nhân tạo

Chương 4-2 **Các phương pháp tìm kiếm có sử dụng thông tin**

Biên soạn: TS Ngô Hữu Phúc

Bộ môn Khoa học máy tính

ĐT: 098 56 96 580

eMail: ngohuuphuc76@gmail.com

Nội dung

- Thuật toán Gen
- Các thành phần cơ bản của thuật toán gen
- Các khuyến cáo khi sử dụng thuật toán gen
- Ưu và nhược điểm của thuật toán gen

10.4.1. Thuật Toán Gene (GAs)

- **GAs** (John Holland, 1975) mô phỏng tiến hóa tự nhiên (Darwinian Evolution) ở mức gen sử dụng tư tưởng của **chọn lọc tự nhiên** (survival of the fittest)
- **Một cá thể (nhiễm sắc thể)** (chromosome) mô tả một lời giải ứng viên của bài toán.
- Một tập các cá thể “alive”, gọi là **quần thể** (population) được tiến hóa từ thế hệ này tới thế hệ khác phụ thuộc vào sự thích nghi của các cá thể.
- **Kỳ vọng** (Hope): Thế hệ mới sinh ra sẽ chứa lời giải tốt của bài toán.

10.4.2. Mô tả thuật toán Gene

- Ban đầu, sinh ra thế hệ khởi tạo với quần thể $P(0)$, chỉ số i chỉ ra thế hệ thứ i .
- Lặp cho đến khi **quần thể hội tụ** hoặc **tiêu chuẩn kết thúc** đạt được.
 - Đánh giá độ thích nghi của mỗi cá thể trong $P(i)$.
 - Lựa chọn các cha từ $P(i)$ dựa trên độ thích nghi của chúng trong $P(i)$.
 - Áp dụng các toán tử Gen (crossover, mutation) từ các cha đã chọn để sinh ra các con (offspring)
 - Đạt được thế hệ tiếp theo $P(i + 1)$ từ các con và các cá thể ở thế hệ $P(i)$

10.4.3. Cài đặt GA

Procedure GA

begin

$t := 0 ;$

 initialize $P(t) ;$

 evaluate $P(t) ;$

while (not termination-condition) **do**

begin

$t := t + 1 ;$

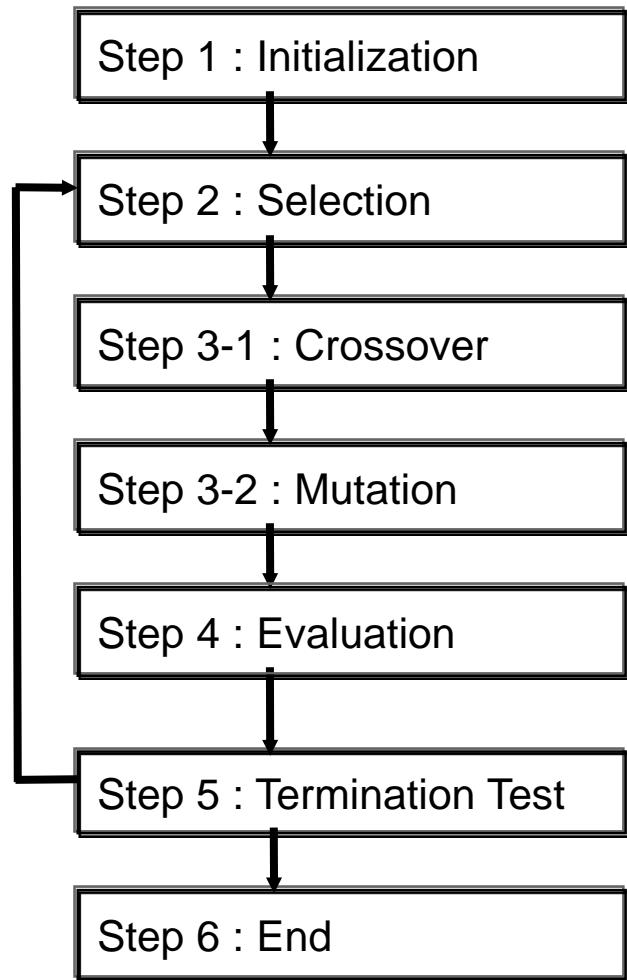
 select $P(t)$ from $P(t-1) ;$

 alter $P(t) ;$

 evaluate $P(t) ;$

end;

end;



10.4.4. Các thành phần cơ bản của GAs

1. Giới thiệu một số bài toán
2. Mã hóa (encoding)
3. Khởi tạo quần thể (initial population generation)
4. Hàm thích nghi (fitness function)
5. Phương pháp lựa chọn (Selection for recombination)
6. Lai ghép (Crossover)
7. Đột biến (Mutation)
8. Chiến lược thay thế (Replacement Strategy)
9. Tiêu chuẩn kết thúc (Termination Criteria)

10.4.4.1. Một số bài toán áp dụng

Bài toán Knapsack 01:

Mô tả bài toán:

- Bạn chuẩn bị đi picnic .
- Và bạn có một số các vật mà bạn có thể cầm theo
- Mỗi vật có một trọng lượng và một giá trị.
- Có một cái túi giới hạn trọng lượng bạn có thể cầm theo.
- Mỗi vật chỉ được chọn tối đa 1 lần.
- Bạn muốn cầm các vật mang theo với max giá trị.



10.4.4.1. Ví dụ về bài toán Knapsack 01

Ví dụ:

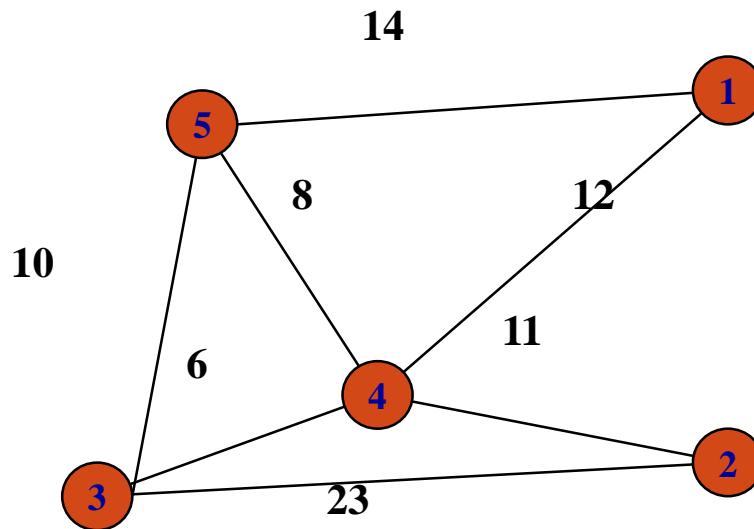
- Đồ vật: 1 2 3 4 5 6 7
- Giá trị: 5 8 3 2 7 9 4
- T.lượng: 7 8 4 10 4 6 4
- Khối lượng tối đa có thể mang là 22 đơn vị.
- Xếp đồ vật để có giá trị lớn nhất???



10.4.4.1. Bài toán TSP (người bán hàng)

Bài toán:

- Một người bán hàng cần ghé qua tất cả các thành phố, mỗi thành phố một lần và trở lại thành phố ban đầu. Có chi phí di chuyển giữa tất cả các thành phố. Tìm hành trình có tổng chi phí nhỏ nhất.



10.4.4.2. Mã hóa (Encoding)

- **Mã hóa nhị phân (Binary encoding)** là kiểu thông dụng nhất : nghiên cứu đầu tiên về thuật toán Gen sử dụng kiểu mã hóa này và bởi vì nó đơn giản
- Trong mã hóa nhị phân, mọi nhiễm sắc thể là chuỗi **bits - 0** hoặc **1**.
Cá thể (Chromosome) A: **101100101100101011100101**
Cá thể (Chromosome) B: **111111100000110000011111**
- Mã hóa nhị phân đưa ra nhiều khả năng của nhiễm sắc thể với một số lượng nhỏ các gen đẳng vị.
- Các mã hóa này thường không tự nhiên cho nhiều bài toán và đôi khi có sai sau khi thực hiện các phép toán **crossover, mutation**.

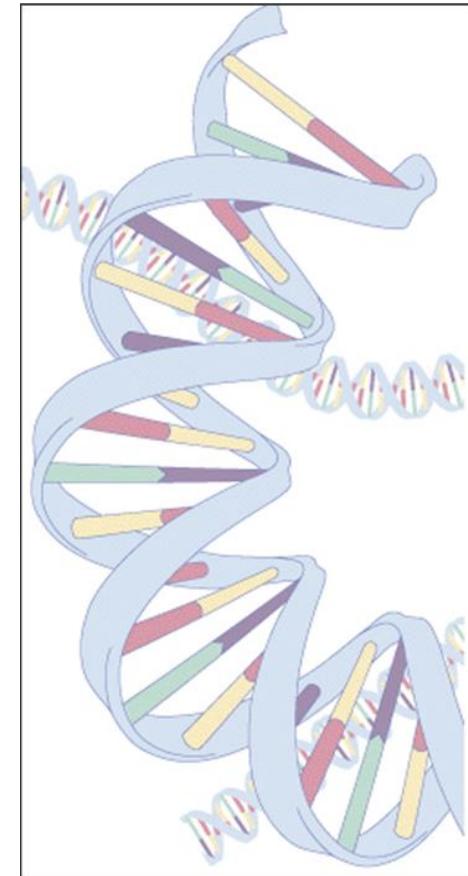
10.4.4.2. Bài toán Knapsack 01

- Encoding: 0 = not exist, 1 = exist in the Knapsack
Chromosome: 1010110

Item.	1	2	3	4	5	6	7
Chro	1	0	1	0	1	1	0
Exist?	y	n	y	n	y	y	n

=> Items taken: 1, 3 , 5, 6.

- Generate random population of n chromosomes:
 - 0101010
 - 1100100
 - 0100011



10.4.4.2. Mã hóa hoán vị (Permutation Encoding)

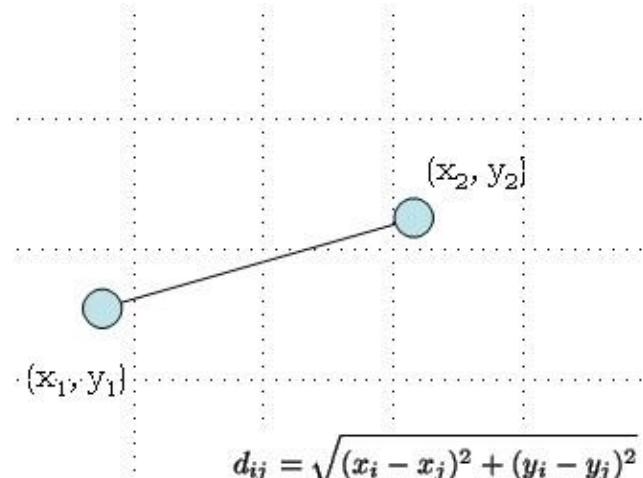
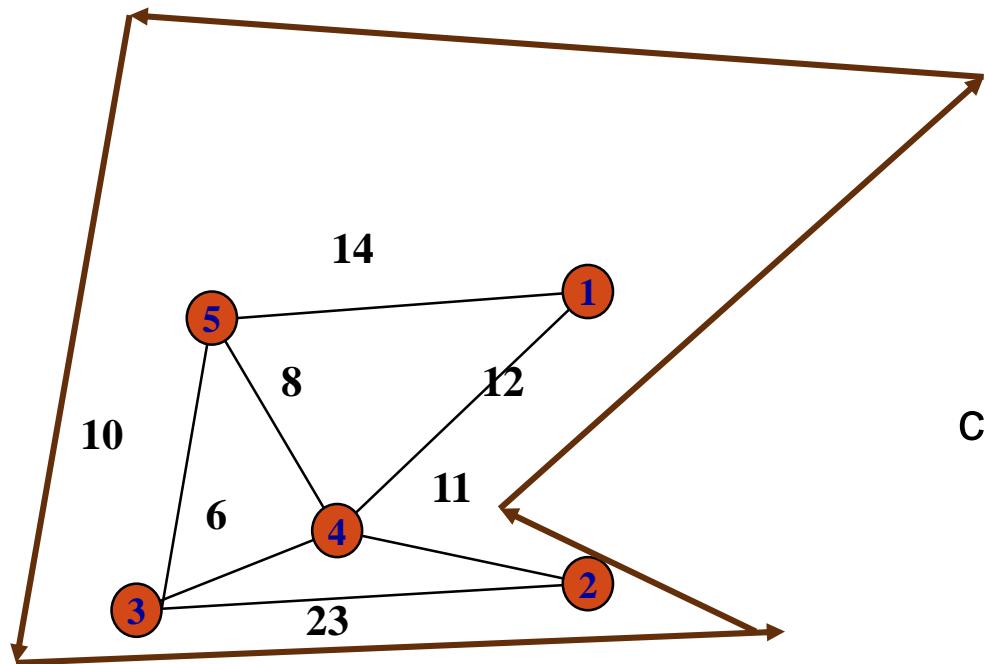
- **Permutation encoding** có thể sử dụng để giải quyết các bài toán có thứ tự như: Người bán hàng.
- Trong permutation encoding, tất cả các NST là chuỗi các số biểu diễn vị trí trong một dãy.

NST A 1 5 3 2 6 4 7 9 8

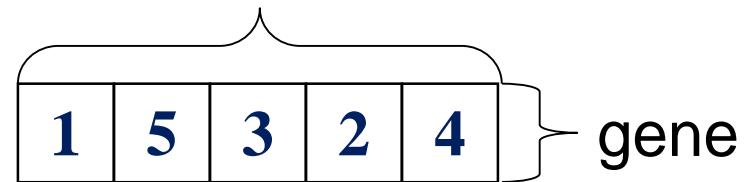
NST B 8 5 6 7 2 3 1 4 9

- Mã hóa hoán vị được sử dụng trong các bài toán có thứ tự.
- **Lưu ý:** Trong một vài trường hợp, việc hiệu chỉnh lai ghép và đột biến phải thực hiện để tạo ra NST phù hợp.

10.4.4.2. Biểu diễn đường đi



chromosome (individual)



10.4.4.2. Mã hóa giá trị (Value Encoding)

- **Value Encoding** được sử dụng trong các bài toán mà việc mã hóa nhị phân là khó thực hiện (Ví dụ như các bài toán mà giá trị là các số thực).
- Trong **VE** mỗi nhiễm sắc thể là một chuỗi các giá trị có thể nhận dạng bất kỳ tùy thuộc vào từng bài toán cụ thể. Ví dụ giá trị có thể là số thực, ký tự, hoặc một đối tượng nào đó.

Chromosome A 1.2324 5.3243 0.4556 2.3293 2.4545

Chromosome B ABDJEIFJDHDIERJFDLDFLATEGT

Chromosome C (back), (back), (right), (forward), (left)

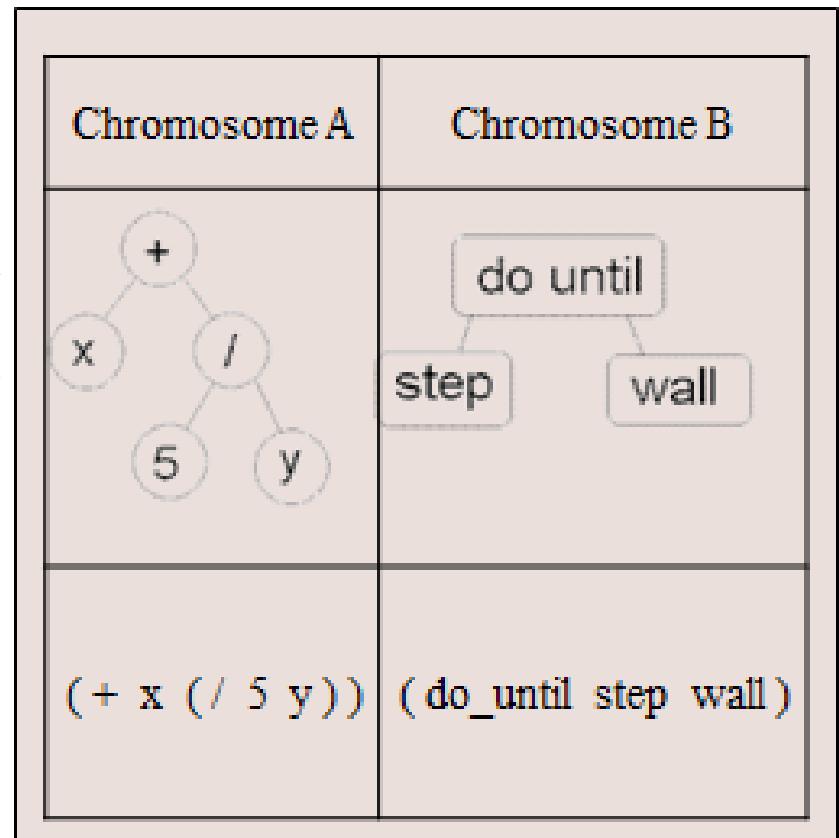
- VE là sự lựa chọn tốt đối với một số bài toán cụ thể.
- Với kiểu mã hóa này thường sẽ phải xây dựng một số phép lai ghép và đột biến cho các bài toán cụ thể.

10.4.4.2. Ví dụ về mã hóa giá trị

- **Bài toán:** Cho mạng neuron A có cấu trúc đã biết. Tìm trọng số giữa các neurons trong mạng để nhận được kết quả ra mong muốn.
- **Mã hóa:** Giá trị thực trong các chromosome biểu diễn các trọng số trong mạng.

10.4.4.2. Mã hóa dạng cây (Tree Encoding)

- TE thường được sử dụng trong các bài toán hoặc các biểu thức suy luận ví dụ như **genetic programming**.
- Trong TE, mỗi chromosome là một cây gồm các đối tượng như là các hàm hoặc các câu lệnh của ngôn ngữ lập trình.
- TE thường được dùng đối với các bài toán suy luận hoặc các cấu trúc có thể biểu diễn bằng cây.
- Các phép lai ghép và đột biến đối với kiểu mã hóa này cũng tương đối dễ thực hiện.



10.4.4.3. Khởi tạo quần thể - Bài toán Knapsack 01

• Khởi tạo:

- Encoding: 0 = not exist, 1 = exist in the Knapsack

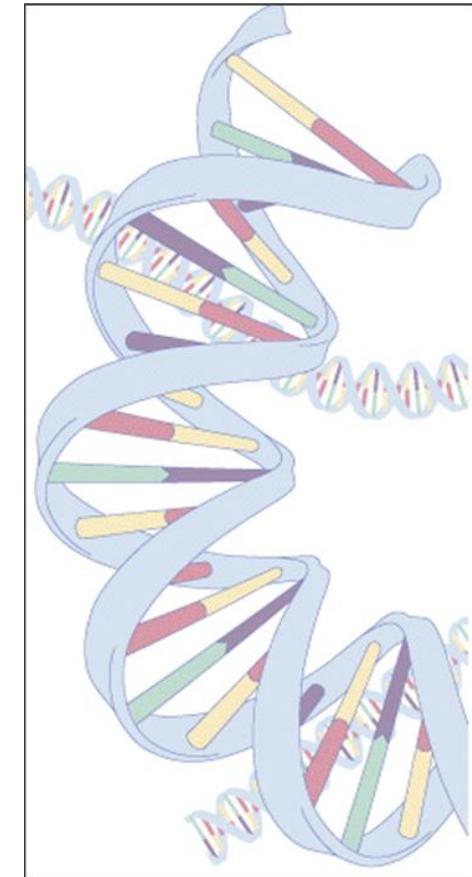
Chromosome: 1010110

Item.	1	2	3	4	5	6	7
Chro	1	0	1	0	1	1	0
Exist?	y	n	y	n	y	y	n

=> Items taken: 1, 3 , 5, 6.

- Generate random population of n chromosomes:

- 0101010
- 1100100
- 0100011



10.4.4.3. Khởi tạo quần thể - Bài toán TSP

1	5	3	2	4
3	2	4	5	1
2	3	4	1	5
4	5	3	2	1

Population

Population size α

individual length

10.4.4.4. Hàm thích nghi - Bài toán Knapsack 01

• Fitness

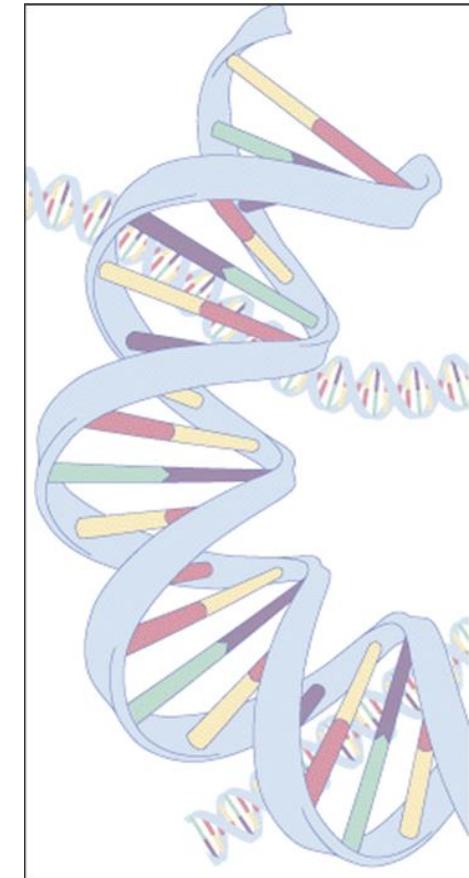
a) 0101010: Benefit= 19, Weight= 24

Item	1	2	3	4	5	6	7
Chro	0	1	0	1	0	1	0
Benefit	5	8	3	2	7	9	4
Weight	7	8	4	10	4	6	4

b) 1100100: Benefit= 20, Weight= 19. ✓

c) 0100011: Benefit= 21, Weight= 18. ✓

=> We select Chromosomes b & c.



10.4.4.4. Hàm thích nghi - Bài toán TSP

<table border="1"><tr><td>1</td><td>5</td><td>3</td><td>2</td><td>4</td></tr></table>	1	5	3	2	4	58
1	5	3	2	4		
<table border="1"><tr><td>3</td><td>2</td><td>5</td><td>1</td><td>4</td></tr></table>	3	2	5	1	4	56
3	2	5	1	4		
<table border="1"><tr><td>2</td><td>3</td><td>4</td><td>1</td><td>5</td></tr></table>	2	3	4	1	5	55
2	3	4	1	5		
<table border="1"><tr><td>4</td><td>5</td><td>3</td><td>2</td><td>1</td></tr></table>	4	5	3	2	1	57
4	5	3	2	1		

$$f(indiv) = \sum_{1 \leq i < n} d_{i(i+1)} + d_{n1}$$

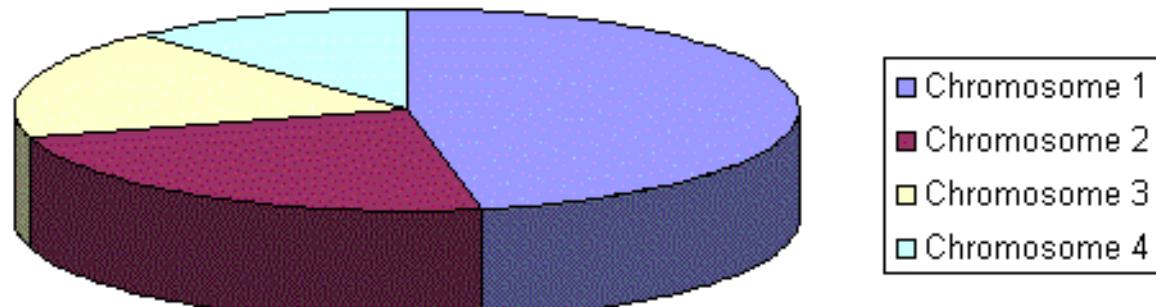
$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

10.4.4.5. Phương pháp lựa chọn

- Các nhiễm sắc thể được chọn từ quần thể là các cha cho lai ghép (crossover). Vấn đề là lựa chọn các nhiễm sắc thể như thế nào?
- Theo thuyết tiến hóa Darwin's những cá thể tốt nhất sẽ được chọn để tạo ra các con.
- Có nhiều phương pháp để chọn nhiễm sắc thể tốt nhất.
- Ví dụ một số phương pháp chọn:
 - roulette wheel selection
 - Boltzman selection
 - tournament selection
 - rank selection
 - steady state selection
 - and many other sometimes weird methods

10.4.4.5.1. Roulette Wheel Selection

- Nhiễm sắc thể được chọn thông qua độ thích nghi của chúng.
- Nhiễm sắc thѣ tốt hơn (mang hàm ý xác suất) được chọn.
- Minh họa về phương pháp chọn:
 - Đặt các NST lên **roulette wheel**.
 - Kích thước của đoạn trong roulette wheel tương ứng với giá trị của hàm thích nghi.
 - Một hòn bi được lăn trong roulette wheel và NST nơi nó dừng lại được lựa chọn.
 - NST với giá trị thích nghi lớn sẽ được chọn nhiều hơn.



10.4.4.5.1. Roulette Wheel Selection

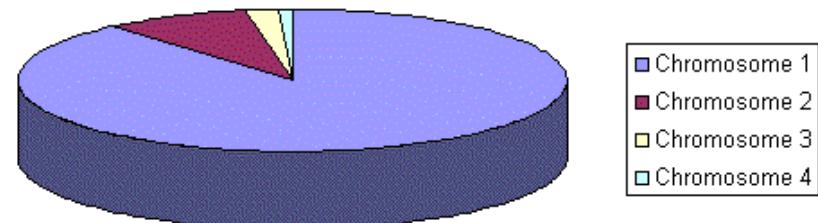
- Tiến trình trên có thể được mô tả bởi thuật toán sau:
 1. **[Tính tổng]** Tính tổng S của giá trị thích nghi của tất cả cá thể trong quần thể.
 2. **[Lựa chọn]** Chọn ngẫu nhiên một giá trị r trong đoạn $(0, S)$.
 3. **[Vòng lặp]** Với giá trị ngẫu nhiên r đã chọn, tương ứng với các thể nào đó khi hình thành S thì cá thể đó được chọn cho thế hệ kế tiếp.

10.4.4.5.2. Rank Selection

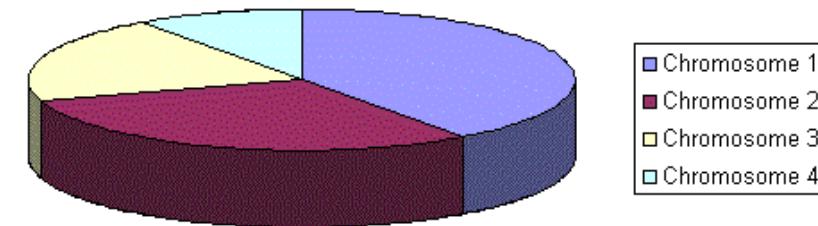
- Roulette Wheel Selection sẽ có vấn đề khi có sự khác nhau lớn giữa các độ thích nghi.
- Ví dụ: nếu NST tốt nhất có độ thích nghi là 90% thì các NST khác rất ít cơ hội được lựa chọn.
- Rank selection tính hạng quần thể đầu tiên và sau đó mọi NST nhận lại giá trị thích nghi được định nghĩa bởi hạng của chúng.
- NST tồi nhất sẽ có độ thích nghi là **1**, NST tồi thứ hai là **2** etc. và NST tốt nhất sẽ có độ thích nghi là **N** (Số nhiễm sắc thể trong quần thể).

10.4.4.5.2. Rank Selection

- Trạng thái sẽ thay đổi sau khi độ thích nghi sẽ được thay đổi bởi hạng của cá thể.
- Nay giờ tất cả các NST sẽ có một số ngẫu nhiên để lựa chọn.



Situation before ranking (graph of fitnesses)



Situation after ranking (graph of order numbers)

10.4.4.5.3. Steady-State Selection

- Trong tất các thế hệ sẽ có một vài NST tốt (Với độ thích nghi cao) được chọn để tạo NST con mới.
- Một vài NST tồi (với độ thích nghi thấp) sẽ bị xóa bỏ và các NST con mới sẽ thay thế chỗ của chúng.
- Phần còn lại của quần thể tạo ra thế hệ mới.

10.4.4.5.4. Chọn cá thể ưu tú - Elitism

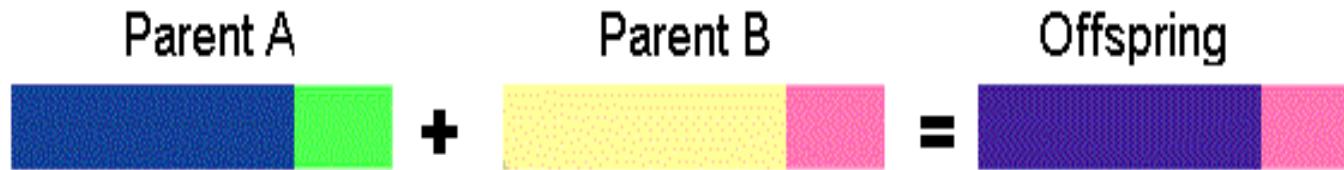
- Khi tạo quần thể mới bằng crossover hoặc mutation, có thể làm mất cá thể tốt nhất.
- Phương pháp **Elitism** cho phép copy những cá thể tốt (hoặc một vài cá thể tốt) sang quần thể mới.
- Thành phần còn lại của quần thể được tạo từ cách khác.
- Phương pháp **Elitism** có thể tăng tốc cho GA, vì nó ngăn không làm mất cá thể tốt.

10.4.4.6. Crossover và Mutation

- **Crossover** và **mutation** là hai phép toán cơ bản của thuật toán gen.
- Sự thi hành thuật toán Gen phụ thuộc rất nhiều vào crossover và mutation.
- Kiểu và sự thể hiện của các toán tử này phụ thuộc vào kiểu mã hóa và cũng phụ thuộc vào bài toán.
- Có nhiều cách để thực hiện toán tử crossover và mutation.

10.4.4.6.1. Crossover for Binary Encoding

- **Single point crossover** – một điểm crossover được lựa chọn, chuỗi nhị phân từ bắt đầu của NST tới điểm crossover được sao chép từ cha 1, phần cuối được sao chép từ cha còn lại
- **11001011+11011111 = 11001111**



10.4.4.6.1. Two point crossover

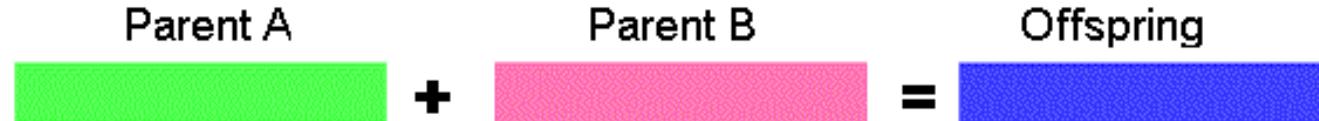
- Hai điểm crossover được chọn.
- Một chuỗi nhị phân từ bắt đầu của NST tới điểm crossover đầu tiên được sao chép từ cha thứ nhất, phần từ điểm đầu tiên tới điểm thứ hai crossover được sao chép từ cha thứ hai và phần cuối được sao chép từ cha đầu tiên
- **11001011 + 11011111 = 11011111**



10.4.4.6.1. Crossover for Binary Encoding

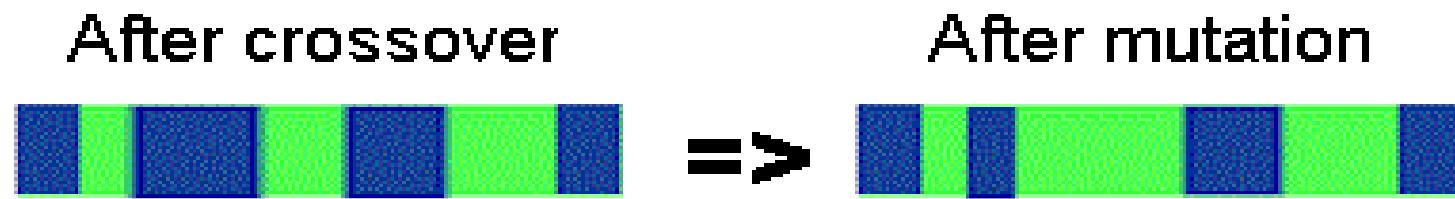
- **Uniform crossover** - Các bit được sao chép ngẫu nhiên từ cha thứ nhất hoặc cha thứ hai.
- **Arithmetic crossover** – Một phép toán được thực hiện để tạo ra con mới (new offspring)

$$11001011 + 11011111 = 11001001 \text{ (LOGICAL AND)}$$



10.4.4.6.1. Mutation for Binary Encoding

- Đảo bit – chọn các bits và đảo ngược giá trị của chúng
- 11001001 => 10001001



10.4.4.6.2. Crossover for Permutation Encoding

- **Single point crossover** – lựa chọn một điểm để lai ghép, permutation được sao chép từ điểm đầu tiên đến điểm lai ghép, sau đó cha còn lại được duyệt qua từng số và nếu số đó chưa có trong con, nó được thêm theo thứ tự của NST thứ hai
- $(1 \ 2 \ 3 \ 4 \ 5 \mid 6 \ 7 \ 8 \ 9) + (4 \ 5 \ 3 \ 6 \ 2 \ 9 \ 1 \ 7 \ 8) = (1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 9 \ 7 \ 8)$
- *Chú ý: có nhiều cách để tạo ra phần sau điểm lai ghép.*

10.4.4.6.2. Crossover for Permutation Encoding

Two point crossover

- Đầu tiên, chọn hai điểm cắt, biểu thị bởi dấu “|”.
- Điểm cắt này được chen vào một cách ngẫu nhiên vào cùng một vị trí của mỗi mẫu cha mẹ

10.4.4.6.2. Ví dụ

- Có hai mẫu cho mẹ p1 và p2, với các điểm cắt sau thành phố thứ ba và thứ bảy

$p1 = (1\ 9\ 2 | 4\ 6\ 5\ 7 | 8\ 3)$

$p2 = (4\ 5\ 9 | 1\ 8\ 7\ 6 | 2\ 3)$

- Hai mẫu con c1 và c2 sẽ được sinh ra theo cách sau. Đầu tiên, các đoạn giữa hai điểm cắt sẽ được chép vào các mẫu con:

$c1 = (x\ x\ x | 4\ 6\ 5\ 7 | x\ x)$

$c2 = (x\ x\ x | 1\ 8\ 7\ 6 | x\ x)$

10.4.4.6.2. Ví dụ (tiếp)

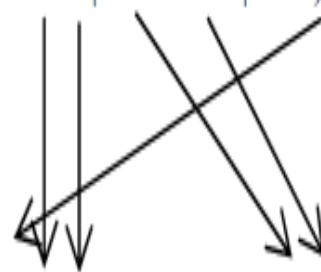
- Bước kế tiếp là bắt đầu từ điểm cắt thứ hai của một trong hai mẫu cha mẹ, nếu ta đang muốn hoàn tất mẫu c1, thì ta sẽ bắt đầu từ điểm cắt thứ hai của mẫu p2, ta chép các thành phố từ điểm cắt này theo thứ tự vào các chỗ còn trống của c1, bỏ qua những thành phố mà c1 đã có. Khi đến cuối mẫu p2, thì quay lại đầu mẫu p2 tiếp tục chép sang c1 cho đến khi c1 đủ.

$$p2 = (4 \textcolor{brown}{5} 9 | 1 8 7 6 | 2 3)$$



$$c1 = (2 3 9 | \textcolor{brown}{4} 6 5 7 | 1 8)$$

$$p1 = (1 9 2 | 4 6 5 7 | 8 3)$$



$$c2 = (3 9 2 | 1 8 7 6 | \textcolor{brown}{4} 5)$$

10.4.4.6.2. Mutation of Permutation Encoding

- Order changing – hai điểm được lựa chọn và thay đổi
- $(1 \text{ } \mathbf{2} \text{ } 3 \text{ } 4 \text{ } 5 \text{ } 6 \text{ } \mathbf{8} \text{ } 9 \text{ } 7) \Rightarrow (1 \text{ } \mathbf{8} \text{ } 3 \text{ } 4 \text{ } 5 \text{ } 6 \text{ } \mathbf{2} \text{ } 9 \text{ } 7)$

10.4.4.6.3. Crossover for Value Encoding

- Có thể sử dụng các lai ghép như kiểu mã hóa nhị phân
- Ví dụ:

(back), (back), (right), (forward), (left)

(right), (back), (left), (back), (forward)

=> (back), (back), (left), (back), (forward)

10.4.4.6.3. Mutation for Value Encoding

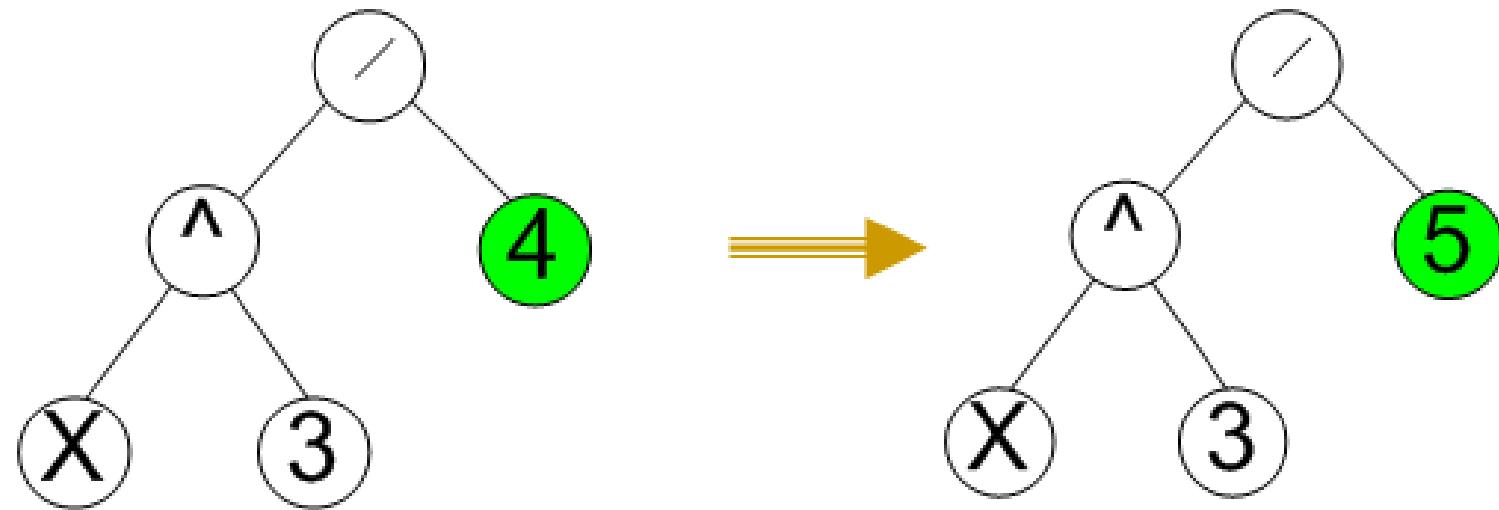
- Cộng vào hoặc trừ đi một số nhỏ từ các giá trị đã được chọn
- Ví dụ:

(1.29 5.68 **2.86** **4.11** 5.55) =>
(1.29 5.68 **2.73** **4.22** 5.55)

10.4.4.6.4. Crossover for Tree Encoding

- **Tree crossover** – Một điểm đột biến được chọn trong cả hai cha, các cha được phân chia bởi điểm đột biến và những phần sau điểm đột biến được thay đổi để tạo ra các con

10.4.4.6.4. Mutation for Tree Encoding



10.4.4.6.5. Crossover and Mutation Probability

- Có hai tham số cơ bản của các thuật toán gen - xác suất crossover và xác suất mutation.
- **Crossover probability:** mức độ thường xuyên crossover được thực hiện .
- Nếu không có crossover, thế hệ con được sao chép chính xác từ cha mẹ.
- Nếu có crossover, cá thể con được tạo ra từ một phần của NST cha.
- Nếu xác suất là **100%**, thì tất cả cá thể con được tạo ra bởi crossover.
- Nếu là **0%**, thế hệ mới tạo ra bằng cách sao chép nguyên các NST của thế hệ cũ.
- Crossover hy vọng tạo ra các NST mới sẽ chứa phần tốt của các NST cũ và như vậy NST mới sẽ tốt hơn.

10.4.4.6.6. Mutation Probability

- **Mutation probability:** Mức độ thường xuyên của các phần sẽ được đột biến.
- Nếu không có mutation, cá thể con sinh ra sẽ giống sau khi crossover (or directly copied).
- Nếu mutation được thực hiện, một hay nhiều phần của NST sẽ được thay đổi.
- Nếu xác suất là **100%**, thì NST sẽ được thay đổi, nếu là **0%**, không có thay đổi gì.
- Mutation sinh ra để tránh thuật toán gen rơi vào cực trị địa phương.
- Mutation không nên xảy ra thường xuyên, vì nếu không GA sẽ thành phương pháp tính toán ngẫu nhiên (**random search**)

10.4.4.7. Các thành phần cơ bản của GAs

- Mã hóa (encoding)
- Khởi tạo quần thể (initial population generation)
- Hàm thích nghi (fitness Function)
- Lựa chọn cho sự kết hợp lại (Selection for recombination)
- Lai ghép (Crossover)
- Đột biến (Mutation)
- **Chiến lược thay thế (Replacement Strategy)**
- Tiêu chuẩn kết thúc (Termination Criteria)

10.4.4.8. Tiêu chuẩn kết thúc

1. Thuật toán dừng khi quần thể hội tụ, i.e. cá thể tốt nhất trong quần thể giống với kết quả mong muốn.
2. Kết thúc khi số thế hệ sinh ra đạt đến số vòng lặp xác định trước.
3. Kết thúc khi các cá thể trở lên giống nhau.
4. Kết thúc khi cá thể tốt nhất trong quần thể không thay đổi theo thời gian.

10.4.4.9. Một số gợi ý cho GENs

- Thử nghiệm thuật toán gen cho bài toán cụ thể, bởi vì không có lý thuyết chung có thể làm hợp các tham số của thuật toán gen cho bất cứ bài toán nào.
- Quá trình có được là kết quả việc học kinh nghiệm của thuật toán gen, thông thường, phương pháp mã hóa nhị phân được sử dụng.

10.4.4.9. Một số gợi ý cho GENs

- **Crossover rate** Tốc độ lai ghép thường là cao, khoảng 80%-95%. (Mặc dù vậy một vài kết quả cho một vài bài toán, tốc độ lai ghép khoảng 60% là tốt nhất.)
- **Mutation rate** Xác suất đột biến thường là rất thấp. Tỷ lệ này tốt nhất khoảng 0.5%-1%.
- **Population size**
 - Kích thước quần thể rất lớn thường không cải tiến tốc độ của thuật toán gen.
 - Kích thước tốt khoảng 20-30, mặc dù một vài trường hợp khoảng 50-100 thì tốt hơn.
 - Nhiều nghiên cứu cho thấy rằng kích thước của quần thể phụ thuộc vào kích thước của chuỗi mã hóa (chromosomes).
 - Nếu có NST 32 bits, thì kích thước quần thể nên cao hơn 16.

10.4.4.9. Một số gợi ý cho GENs

- **Lựa chọn (Selection):**
 - roulette wheel selection có thể được sử dụng, nhưng thỉnh thoảng rank selection có thể tốt hơn.
 - Có một số phương pháp phức tạp khác cho phép thay đổi tham số khi chọn trong khi thực hiện GA.
 - **Elitism** được chọn nếu không sử dụng cách nào khác để lưu thông tin của cá thể tốt nhất.
- **Encoding:** Phụ thuộc vào bài toán.
- Kiểu **Crossover và mutation:** Phụ thuộc vào mã hóa và bài toán

10.4.4.10. Ưu điểm

- Ưu điểm chính là khả năng song song của thuật toán .
- Gas duyệt qua không gian tìm kiếm sử dụng nhiều cá thể (and with genotype rather than phenotype) và ít mắc phải cực trị địa phương như các thuật toán khác.
- Dễ thể hiện.
- Khi đã có thuật toán gen cơ bản, chỉ cần viết một NST mới (just one object) để xử lý bài toán khác.
- Với cùng cách mã hóa, có thể thay đổi hàm thích nghi.
- Mặc dù vậy, trong một số trường hợp chọn và thể hiện mã hóa sẽ gặp khó khăn.

10.4.4.11. Nhược điểm

- Nhược điểm chính của Gas là thời gian tính toán.
- GAs có thể chậm hơn các thuật toán khác.
- Có thể kết thúc tính toán bất cứ lúc nào.

10.4.4.12. Tại Sao GAs Tốt?

- Building blocks hypothesis and schema theorem (Holland, 1975).

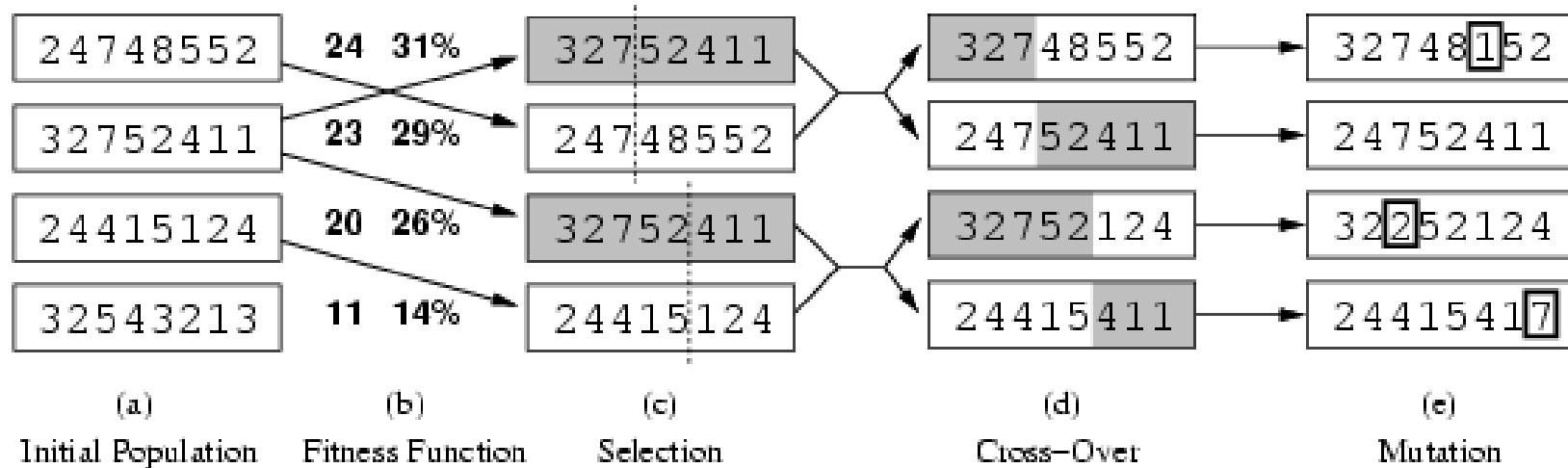
*0010**0*001110*****0

00010110100111010010

- “GAs operator set a bias towards short, low order and highly fit building blocks”.

10.4.4.12. Tại Sao GAs Tốt?

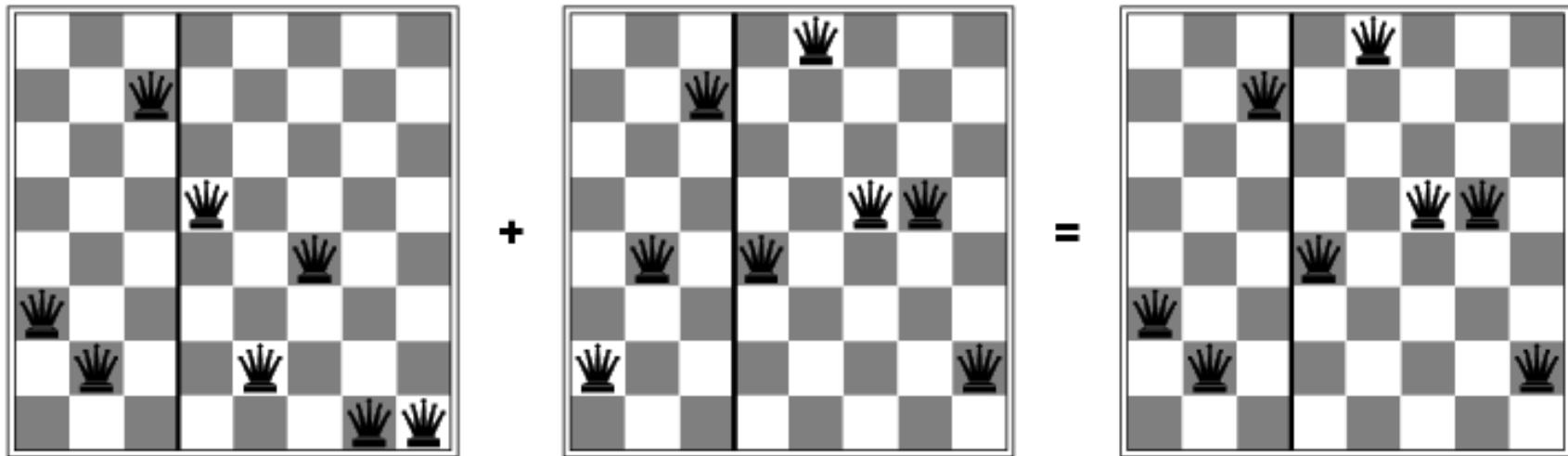
Ví dụ cho biểu diễn phi nhị phân



- Fitness function: số lượng con hậu không ăn nhau ($\min = 0, \max = 8 \times 7/2 = 28$)
- $24/(24+23+20+11) = 31\%$
- $23/(24+23+20+11) = 29\% \text{ etc}$

10.4.4.12. Tại Sao GAs Tốt?

Minh họa



NHẬP MÔN TRÍ TUỆ NHÂN TẠO

Chương 5 CÁC CHIẾN LƯỢC TÌM KIẾM CÓ ĐỐI THỦ

Biên soạn: TS Ngô Hữu Phúc

Bộ môn Khoa học máy tính

ĐT: 098 56 96 580

eMail: ngohuuphuc76@gmail.com

Thông tin chung

- Thông tin về nhóm môn học:

TT	Họ tên giáo viên	Học hàm	Học vị	Đơn vị công tác (Bộ môn)
1	Ngô Hữu Phúc	GVC	TS	BM Khoa học máy tính
2	Trần Nguyên Ngọc	GVC	TS	BM Khoa học máy tính
3	Hà Chí Trung	GVC	TS	BM Khoa học máy tính
4	Trần Cao Trường	GV	ThS	BM Khoa học máy tính

- Thời gian, địa điểm làm việc: Bộ môn Khoa học máy tính Tầng 2, nhà A1.
- Địa chỉ liên hệ: Bộ môn Khoa học máy tính, khoa Công nghệ thông tin.
- Điện thoại, email: 069-515-329, ngohuuphuc76.mta@gmail.com.

Cấu trúc môn học

- Chương 1: Giới thiệu chung.
- Chương 2: Logic hình thức.
- Chương 3: Các phương pháp tìm kiếm mù.
- Chương 4: Các phương pháp tìm kiếm có sử dụng thông tin.
- Chương 5: Các chiến lược tìm kiếm có đối thủ.
- Chương 6: Các bài toán thỏa ràng buộc.
- Chương 7: Nhập môn học máy.

Bài 5: Tìm kiếm có đối thủ

Chương 5, mục: 5.1 – 5.3

Tiết: 1-3; Tuần thứ: 6 (thực hành chương 3-4),7.

Mục đích, yêu cầu:

1. Nắm được ý tưởng phương pháp xây dựng cây trò chơi.
2. Nắm được phương pháp sử dụng chiến lược Minimax.
3. Nắm được phương pháp cắt tỉa Alpha – Beta.
4. Qua đó, xây dựng chương trình cho chương 5.

Hình thức tổ chức dạy học: Lý thuyết.

Thời gian: 3 tiết.

Địa điểm: Giảng đường do Phòng Đào tạo phân công

Nội dung chính: (Slides)

Nội dung:

1. Cây trò chơi và tìm kiếm trên cây trò chơi.
2. Chiến lược Minimax.
3. Phương pháp cắt tỉa Alpha – Beta.

5.1. Cây trò chơi và tìm kiếm trên cây trò chơi

- Trong bài, nghiên cứu các trò chơi có hai người tham gia; như:
 - cờ vua,
 - cờ ca rô,
 - cờ tướng...
- Người chơi là quân Trắng, đối thủ là quân Đen.
- **Mục tiêu: nghiên cứu giải thuật cho quân Trắng đi.**

5.1. Cây trò chơi và tìm kiếm trên cây trò chơi (t)

Một số đặc điểm:

- 2 người thay phiên đưa ra các nước đi tuân theo một luật nào đó.
- Các luật trên là như nhau cho cả 2 người.
- Cả 2 người chơi đều biết được thông tin đầy đủ về các tình thế trong trò chơi.
- Trong vấn đề trò chơi, thực chất là tìm kiếm nước đi, một nước tốt sao cho, sau một số nước đi dẫn đến trạng thái kết thúc.

5.1. Cây trò chơi và tìm kiếm trên cây trò chơi (t)

Khó khăn:

- Vấn đề tìm kiếm ở đây khó khăn hơn với việc tìm kiếm trong các bài trước, vì:
 - Ở trong vấn đề này, có đối thủ, nên không biết đối thủ sẽ đi như thế nào.
 - Nếu có thể tổng quát, cũng sẽ rất khó vì không gian tìm kiếm quá rộng.
 - Nói chung, không thể tìm được lời giải tối ưu, chỉ tìm được lời giải xấp xỉ.

5.1. Cây trò chơi và tìm kiếm trên cây trò chơi (t)

Giải pháp: trong trò chơi, có thể coi như tìm kiếm trong không gian trạng thái, mỗi trạng thái là một tình thế của trò chơi. Có thể tóm tắt giải pháp:

- Trạng thái ban đầu là sự sắp xếp các quân cờ trong lúc đầu của cuộc chơi.
- Các nước đi hợp lệ là các toán tử.
- Các trạng thái kết thúc là các tình thế mà cuộc chơi dừng, thường đã xác định, có thể thông qua hàm kết quả.
- Có thể biểu diễn không gian trạng thái trên cây trò chơi.

5.1. Cây trò chơi và tìm kiếm trên cây trò chơi (t)

Cách xây dựng cây trò chơi:

- Gốc của cây ứng với trạng thái u.
- Có thể gọi đỉnh ứng với trạng thái Trắng (Đen) đưa ra nước đi là đỉnh Trắng (Đen).
- Nếu một đỉnh là Trắng (Đen) ứng với trạng thái u, thì đỉnh con của nó là tất cả các đỉnh biểu diễn trạng thái v, v nhận được từ u do Trắng (Đen) thực hiện nước đi hợp lệ nào đó.

Nhận xét:

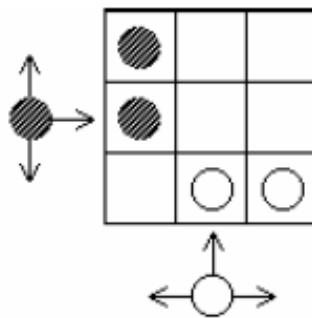
- Độ cao của cây là tổng số nước đi của cả 2 người.
- Trên cùng một mức của cây, các đỉnh đều là Trắng hoặc Đen.
- Các lá của cây ứng với các trạng thái kết thúc.

5.1. Cây trò chơi và tìm kiếm trên cây trò chơi (t)

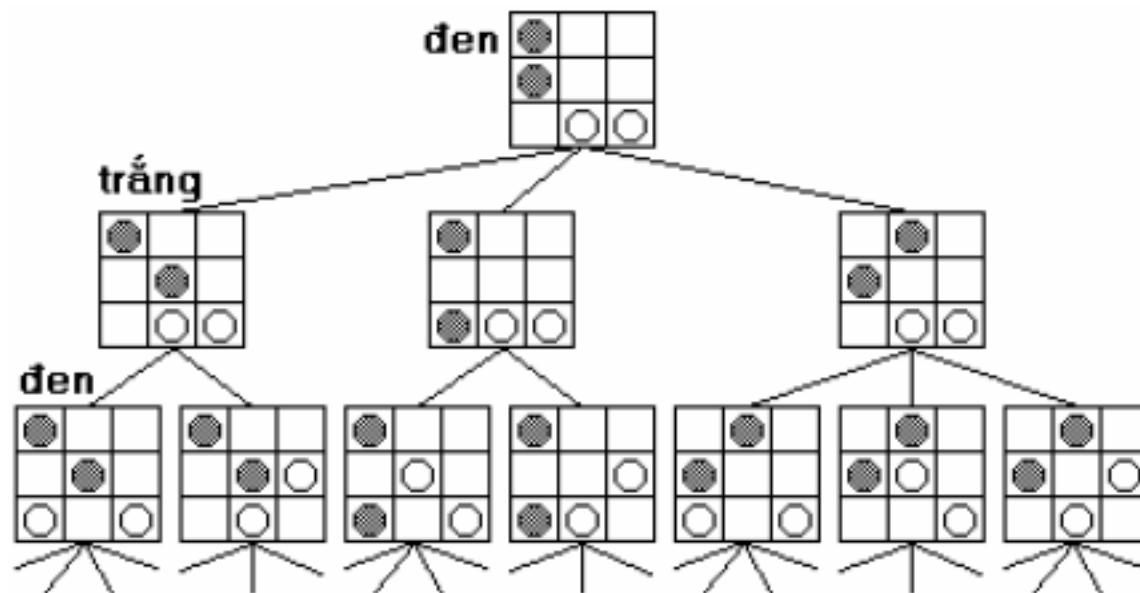
Ví dụ: Xét trò chơi Dodgen (được đưa ra bởi Colin Vout):

- Trên bàn cờ có 2 loại quân Trắng và Đen, được sắp trên bàn cờ 3x3. (như hình vẽ)
- Quân đen có thể đi tới ô trống bên phải, ở trên hoặc bên dưới.
- Quân đen nếu ở cột ngoài cùng có thể đi ra ngoài bàn cờ.
- Quân trắng có thể đi tới ô trống ở bên trái, bên phải, ở trên.
- Quân trắng nếu ở hàng trên cùng có thể đi ra ngoài bàn cờ.
- Trạng thái kết thúc: ai đưa được quân 2 quân của mình ra khỏi bàn cờ; hoặc bắt đối phương không đi được nữa.

5.1. Ví dụ: Trò chơi Dodgen.

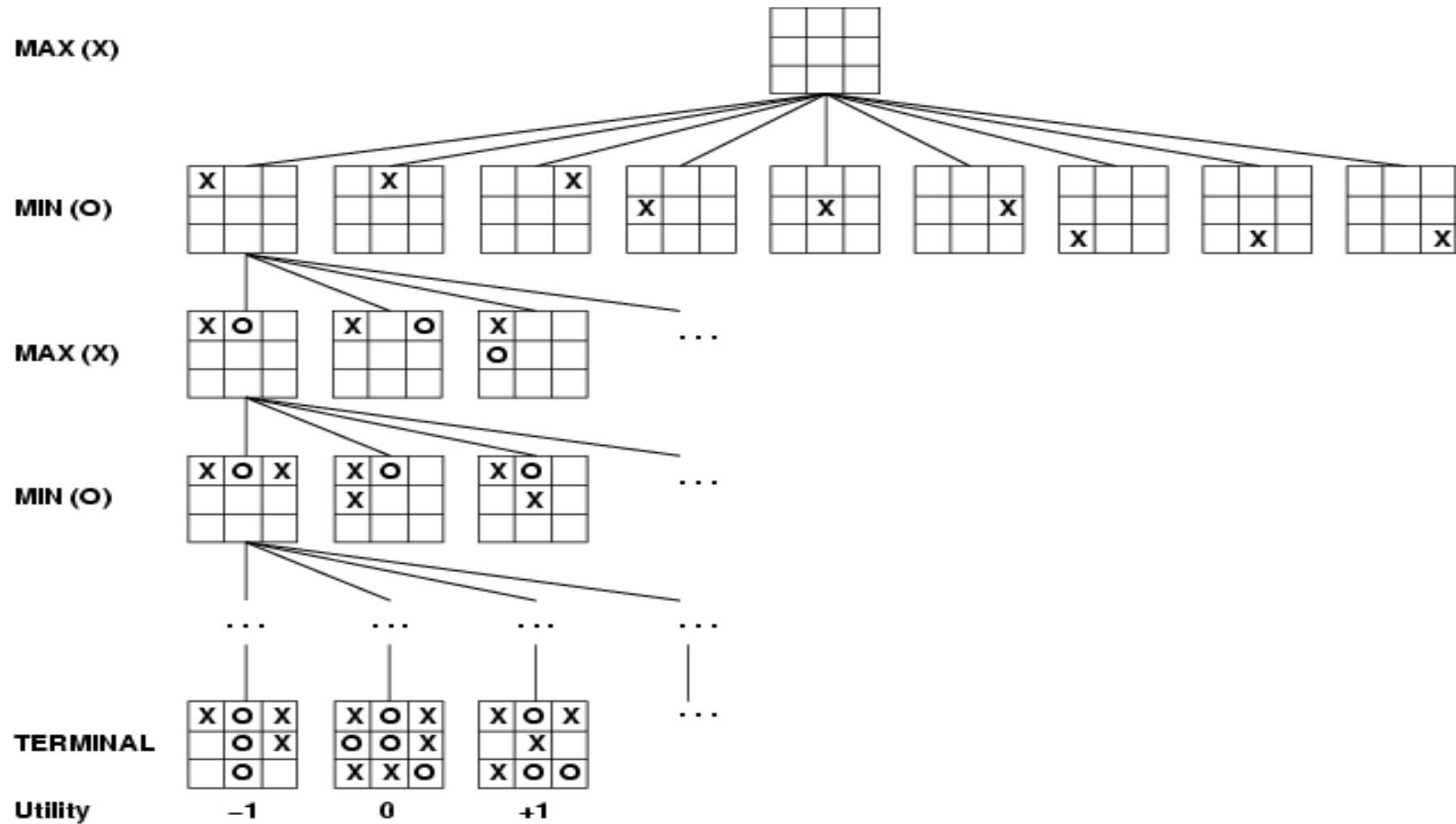


Trạng thái đầu của trò chơi Dodgen



Cây trò chơi Dodgen với quân Đen đi trước

5.1. Ví dụ: Trò chơi dạng caro:



5.2. Chiến lược Minimax

Một số nhận định về chiến lược Minimax:

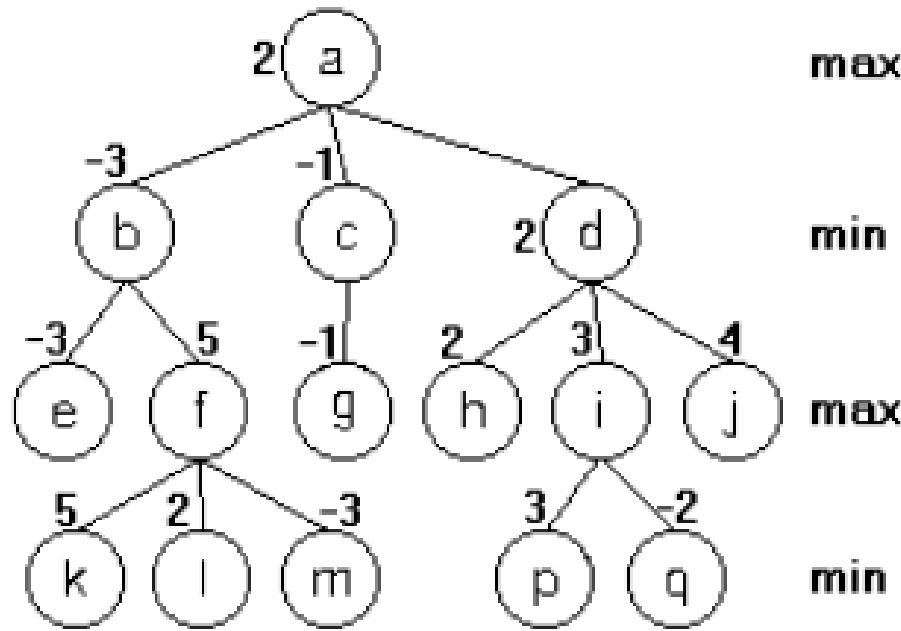
- Giả sử đến một thời điểm đường đi đã dẫn tới đỉnh **u**.
- Nếu **u** là đỉnh Trắng thì Trắng cần chọn đi tới một trong các Đen **v** là con của **u**.
- Nước đi tối ưu cho Trắng là nước đi dẫn tới đỉnh con **v** là đỉnh **tốt nhất** cho Trắng trong số các đỉnh con. Tương tự cho việc lựa chọn nước đi cho quân Đen.
- Để chọn nước đi tốt nhất cho Trắng tại đỉnh **u**, ta cần xác định giá trị các đỉnh của cây trò chơi có gốc là **u**.
- Giá trị của các lá được xác định thông qua hàm kết quả.
- Đỉnh có giá trị càng lớn càng tốt cho Trắng, đỉnh có giá trị càng nhỏ càng tốt cho Đen.

5.2. Chiến lược Minimax (tiếp)

Cách tính điểm cho các đỉnh trên cây trò chơi:

- Để xác định giá trị các đỉnh có gốc là **u**, ta đi từ mức thấp nhất đến **u**.
- Giả sử xét đỉnh **v** trên cây, các giá trị các đỉnh con của nó đã xác định.
- Nếu **v** là đỉnh Trắng, giá của nó được xác định là giá trị lớn nhất trong các giá trị của các đỉnh con.
- Nếu **v** là đỉnh Đen, giá của nó là giá trị nhỏ nhất trong các giá trị của các đỉnh con.

5.2. Chiến lược Minimax (tiếp)



Gán giá trị cho các đỉnh của cây trò chơi.

Ví dụ:

đỉnh f là đỉnh Trắng, giá của đỉnh f = $\max(5, 2, -3) = 5$

đỉnh d là đỉnh Đen, giá của đỉnh d = $\min(2, 3, 4) = 2$

5.2. Chiến lược Minimax (tiếp)

Các hàm trong chiến lược Minimax

Hàm gán giá trị max:

Function MaxValue(u);

Begin

If u là lá **then** MaxValue(u) $\leftarrow f(u)$

Else MaxValue(u) $\leftarrow \max \{MinValue(v) \mid v \text{ là các đỉnh con của } u\}$

End;

Hàm gán giá trị min:

Function MinValue(u);

Begin

If u là lá **then** MinValue(u) $\leftarrow f(u)$

Else MinValue(u) $\leftarrow \min \{MaxValue(v) \mid v \text{ là các đỉnh con của } u\}$

End;

5.2. Chiến lược Minimax (tiếp)

Các hàm trong chiến lược Minimax

Thủ tục minimax:

Procedure Minimax(u,v);

Begin

Value $\leftarrow -\infty$;

For mỗi w là đỉnh con của u **do**

If Value $\leq \text{MinValue}(w)$ **then**

Begin Value $\leftarrow \text{MinValue}(w)$;

v $\leftarrow w$; **end**;

End;

Trong đoạn chương trình trên, chọn nước đi cho Trắng tại u, v là biến lưu lại trạng thái mà Trắng đã chọn đi từ u.

5.2. Chiến lược Minimax (tiếp)

Đánh giá về chiến lược Minimax:

1. Tính đủ

- Có (nếu cây tìm kiếm là hữu hạn).

2. Tính tối ưu

- có (đối với đối thủ luôn ra nước tối ưu)

3. Độ phức tạp thời gian?

- $O(b^m)$ (với m là độ cao của cây, tại mỗi đỉnh có b nước đi)

4. Độ phức tạp không gian?

- $O(bm)$ (DFS)
- Ví dụ cờ vua, thông thường $b \approx 35$, $m \approx 100 \rightarrow$ Tìm lời giải chính xác và tối ưu là không thể được

5.2. Chiến lược Minimax (tiếp)

Hàm đánh giá trong chiến lược Minimax:

- Trong chiến lược Minimax, nếu có hàm $f()$, hàm kết quả, thì chương trình có giá trị tối ưu, tuy nhiên, cần xét cả không gian trạng thái của cây trò chơi.
- Để tìm ra kết quả nhanh, nước đi tốt, ta có thể sử dụng hàm đánh giá, hàm này chỉ xét một bộ phận của cây trò chơi.
- Chất lượng của chương trình phụ thuộc vào hàm đánh giá, nếu hàm đánh giá không chính xác về trạng thái sẽ cho nước đi kém.
- Hàm đánh giá phụ thuộc vào nhiều nhân tố của trò chơi. Ở đây có sự mâu thuẫn giữa độ chính xác và thời gian tính toán.
- Trong trò chơi Dodgen, hàm đánh giá eval() xác định lợi thế của trạng thái u.
 - Nếu eval() càng dương, thuận lợi cho Trắng;
 - Nếu eval() càng âm, thuận lợi cho Đen;
 - Nếu eval() ≈ 0 thì không thuận lợi cho ai cả.

5.2. Ví dụ: Một số hàm đánh giá

Ví dụ 1: Xây dựng hàm đánh giá cho bàn cờ vua.

- Mỗi quân trên bàn cờ được gán một giá trị, phù hợp với “sức mạnh” của con cờ. Giả sử:
 - Quân tốt Trắng (Đen) được gán giá trị 1(-1)
 - Quân mã Trắng (Đen) được gán giá trị 3(-3)
 - Quân xe Trắng (Đen) được gán giá trị 5(-5)
 - Quân hậu Trắng (Đen) được gán giá trị 9(-9)
- Hàm đánh giá tại mỗi trạng thái:

$$\text{Eval}() = s_1w_1 + s_2w_2 + \dots + s_nw_n.$$

- **Nhận xét:** Hàm trên không quan tâm tới vị trí quân cờ.

5.2. Ví dụ: Một số hàm đánh giá

Ví dụ 2: Xây dựng hàm đánh giá cho trò chơi Dodgen.

- Mỗi vị trí của quân Trắng (Đen) được cho giá trị như hình vẽ.

30	35	40
15	20	25
0	5	10

Giá trị quân Trắng.

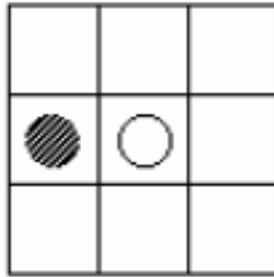
-10	-25	-40
-5	-20	-35
0	-15	-30

Giá trị quân Đen.

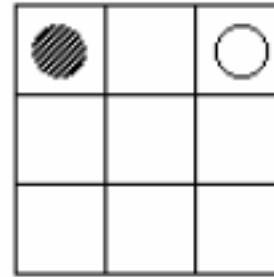
- Nếu quân Trắng cản trực tiếp quân Đen thì thêm 40 điểm, nếu cản gián tiếp được thêm 30 điểm.
- Ngược lại, nếu quân Đen cản trực tiếp quân Trắng thì thêm -40 điểm, nếu cản gián tiếp được thêm -30 điểm.

5.2. Ví dụ: Một số hàm đánh giá

Ví dụ 2 (Tiếp): Xây dựng hàm đánh giá cho trò chơi Dodgen.



Trắng cản trực tiếp Đen
được thêm 40 điểm.



Trắng cản gián tiếp Đen
được thêm 30 điểm.

Đánh giá tương quan giữa quân Trắng và Đen.

Nhận xét: Với cách xây dựng như trên, hàm đánh giá có xét tới vị trí các quân trên bàn cờ và mối tương quan giữa các quân cờ.

5.2. Ví dụ: Một số hàm đánh giá

Ví dụ 2 (Tiếp): Xây dựng hàm đánh giá cho trò chơi Dodgen.

●		
●	○	○

75

	●	
	○	
●		○

-5

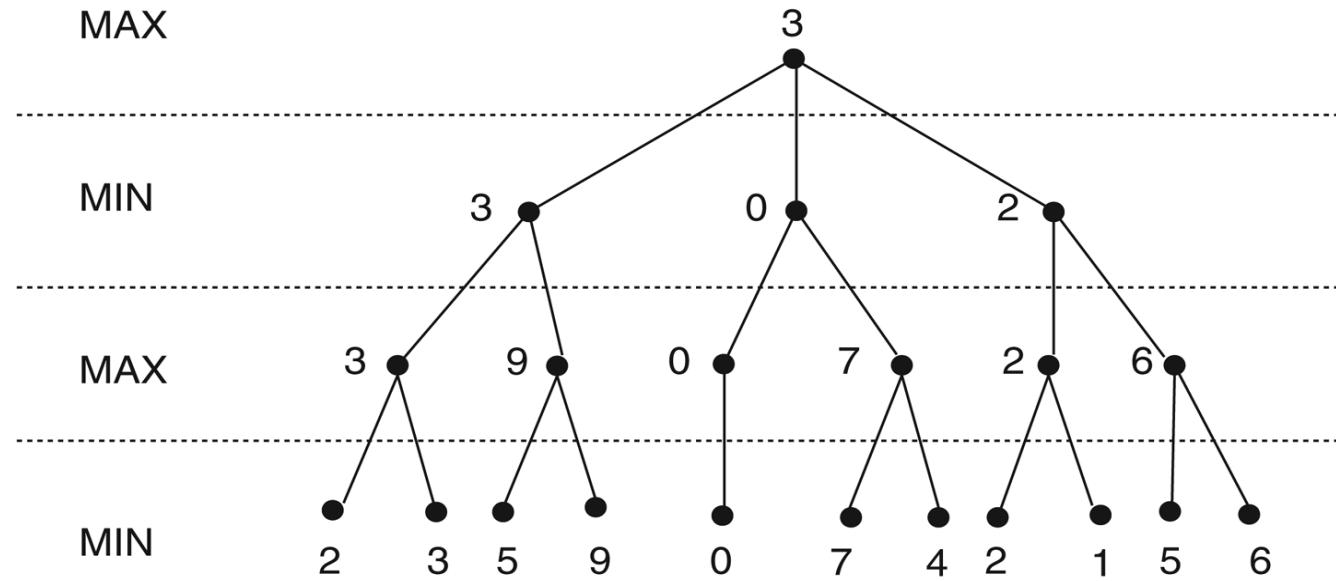
Giá trị của một số trạng thái trong Dodgen

5.3. Chiến lược minimax với độ sâu cố định

- Trong các trò chơi, hiếm khi có khả năng mở rộng đến nút lá.
- Khi đó, có thể áp dụng chiến lược tính trước **n** bước đi.
- Giá trị trong các nút con không phản ánh giá trị thắng thua, chỉ phản ánh giá trị heuristic nào đó.
- Giá trị được truyền ngược cũng không đánh giá việc thắng thua, cũng chỉ là giá trị heuristic của trạng thái tốt nhất có thể tiếp cận.

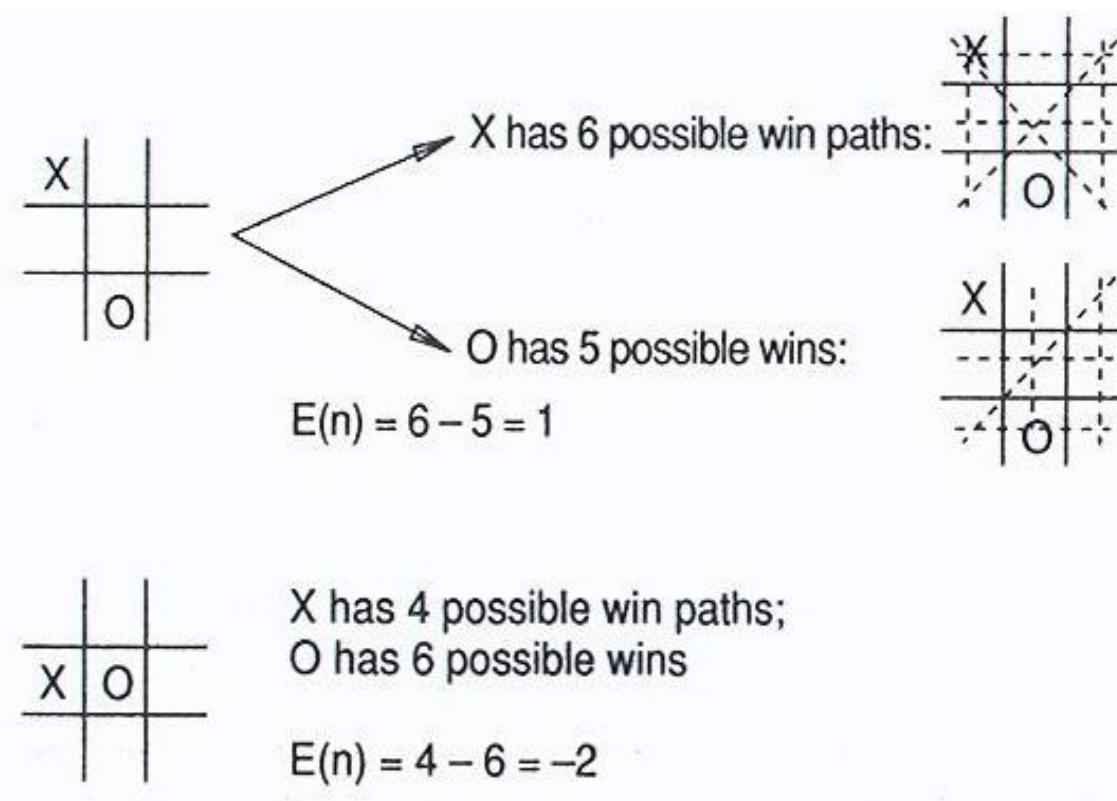
5.3. Chiến lược minimax với độ sâu cố định

- Minimax đối với một KGTT giả định.



- Các nút lá được gán các giá trị ***heuristic***
- Còn giá trị tại các nút trong là các giá trị nhận được dựa trên giải thuật Minimax

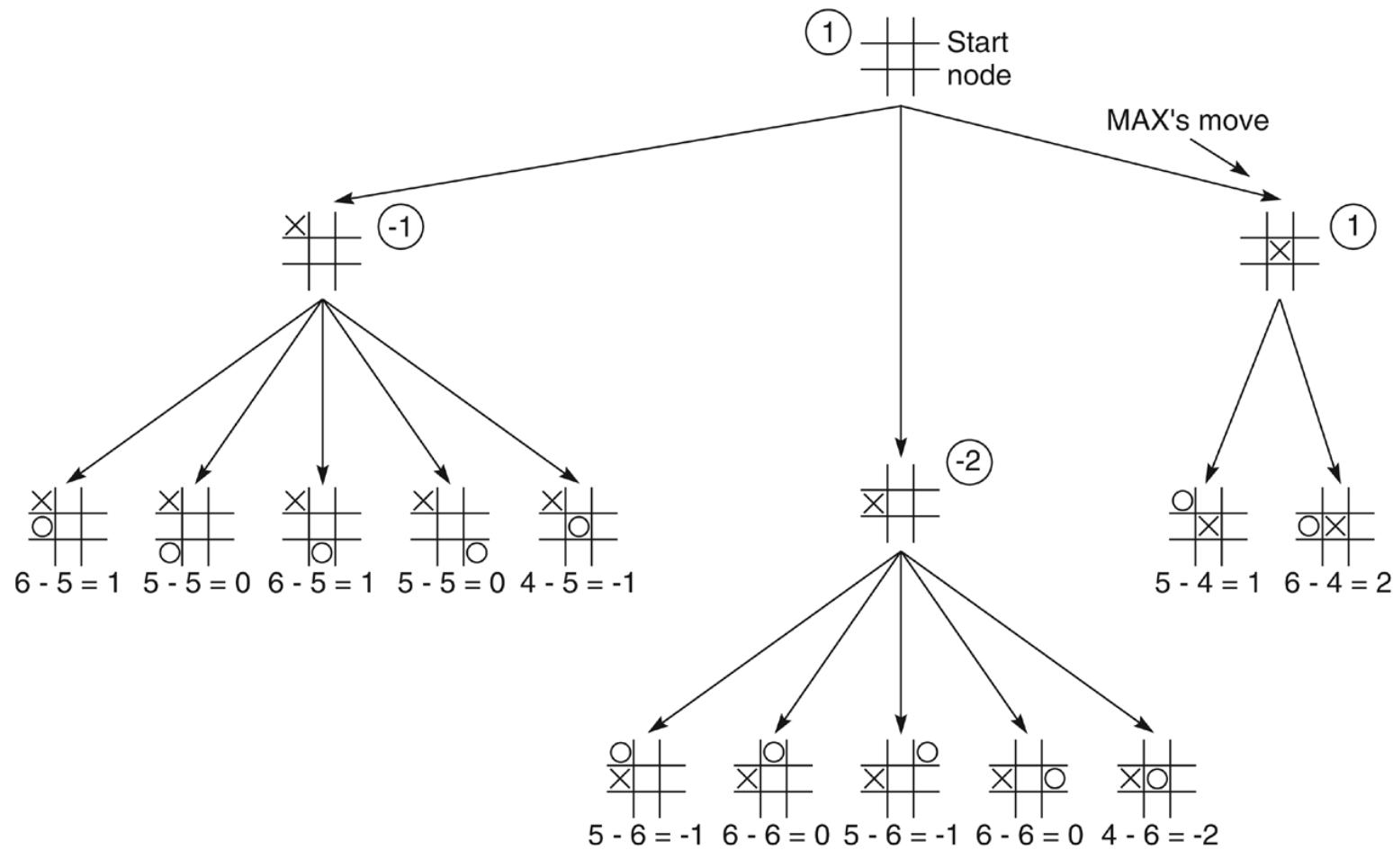
5.3. Heuristic trong trò chơi tic-tac-toe



- **Hàm Heuristic:** $E(n) = M(n) - O(n)$

Trong đó: $M(n)$ là tổng số đường thắng có thể của tôi
 $O(n)$ là tổng số đường thắng có thể của đối thủ
 $E(n)$ là trị số đánh giá tổng cộng cho trạng thái n

5.3. Heuristic trong trò chơi tic-tac-toe (cont)



Trích từ Nilsson (1971).

5.4. Phương pháp cắt tỉa alpha-beta

Nhận xét về các giải thuật trước:

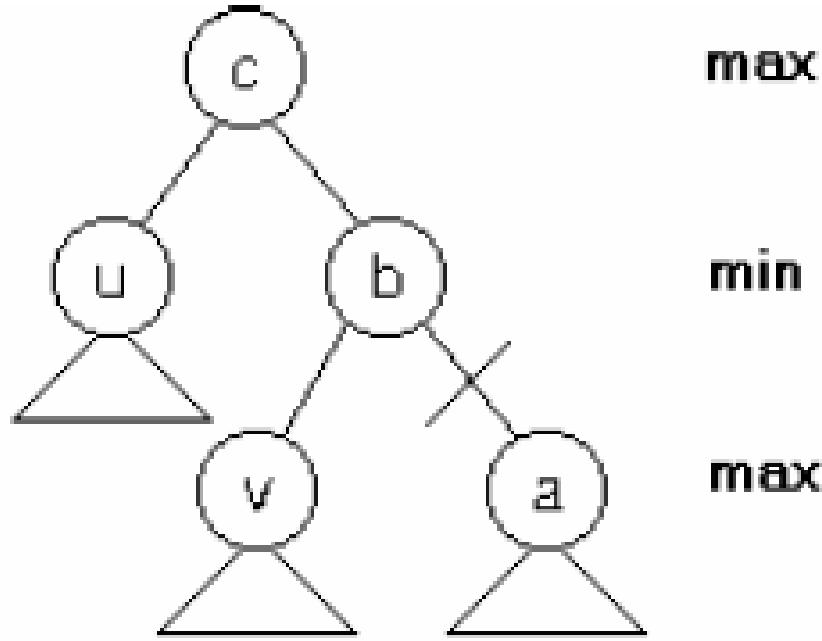
- Trong chiến lược minimax, để tìm nước đi tốt cho quân Trắng tại trạng thái u, cho dù đã hạn chế không gian bằng việc giảm độ cao, thì cũng rất lớn nếu $h \geq 3$.
- Ví dụ:
 - Với trò chơi cờ vua, nếu máy tính có thể tính 10^6 nước/s
 $b^m = 10^6$, $b=35 \rightarrow m=4$
 - Tuy nhiên,
 - 4-ply ≈ người mới học chơi
 - 8-ply ≈ PC, chuyên gia cờ
 - 12-ply ≈ Deep Blue, Kasparov.

5.4. Phương pháp cắt tỉa alpha-beta (cont)

Tư tưởng của phương pháp alpha-beta:

- Giả sử quá trình tìm kiếm đi xuống đỉnh Trắng **a**, đỉnh **a** có đỉnh cùng cấp **v** đã xét.
- Giả sử đỉnh **a** có cha là **b**, **b** có đỉnh cùng cấp là **u** đã xét; cha của **b** là **c**.
- Khi đó, giá của **c** ít nhất là **u**, giá của **b** nhiều nhất là **v**.
- Nếu **eval(u) > eval(v)**, ta không cần đi xuống đỉnh **a** nữa mà không ảnh hưởng tới giá của **c**.
- Lập luận tương tự cho đỉnh **a** là Đen, với đánh giá **eval(u) < eval(v)**.

5.4. Phương pháp cắt tỉa alpha-beta (cont)



Cắt tỉa gốc a nếu $\text{eval}(u) > \text{eval}(v)$

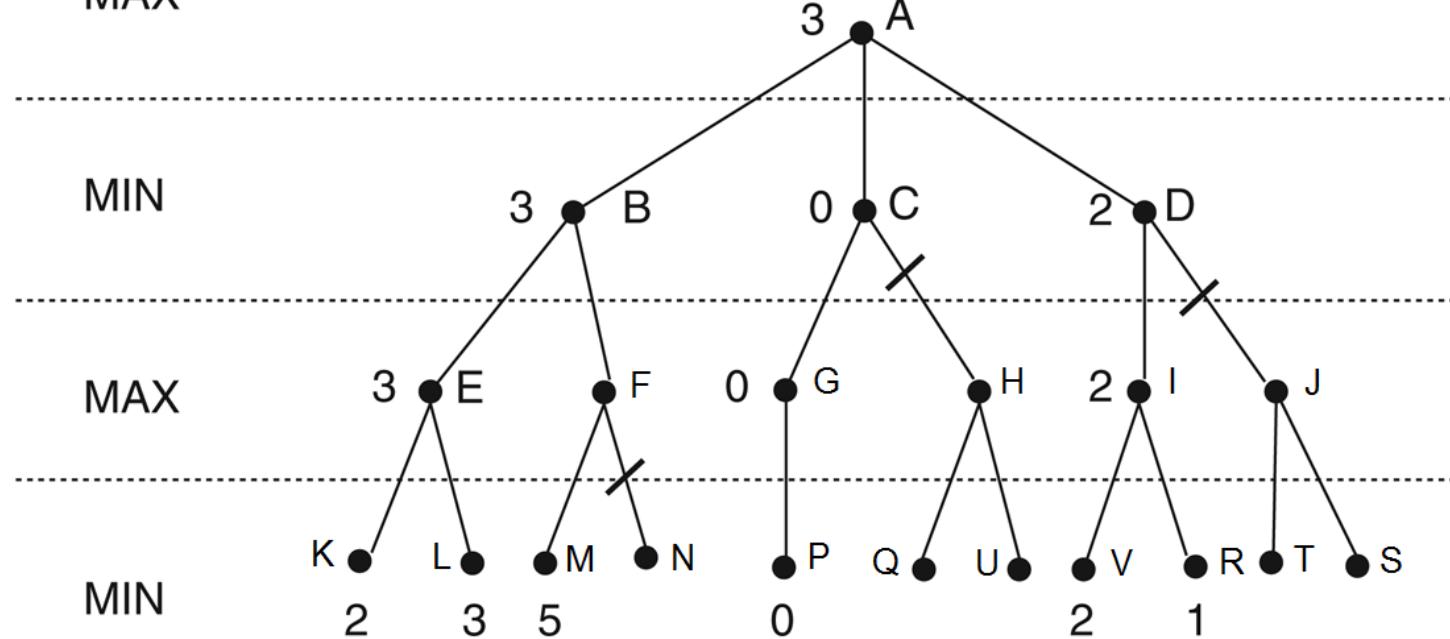
5.4. Phương pháp cắt tỉa alpha-beta (cont)

MAX

MIN

MAX

MIN



- Xét K(2) và L(3), khi đó E có giá trị $3 = \max(2,3)$.
- Vì $M=5$, nên ít nhất $F=5$, do đó, không cần xét nhánh N, có thể kết luận $B=3$ (cắt bớt beta).
- Tương tự, xét P(0), suy ra $G=0$. Do chọn min, suy ra C nhiều nhất =0. Vì A chọn max, nên không cần xét H.
- Tương tự, D nhiều nhất bằng 2, mà chọn A theo max, $B>I$, nên không cần xét J.

5.4. Phương pháp alpha-beta (cont)

Các hàm trong chiến lược Alpha-beta

- Hàm sử dụng α để ghi giá trị lớn nhất trong các giá trị của đỉnh con đã đánh giá của một đỉnh trắng, β ghi giá trị nhỏ nhất trong các đỉnh con của một đỉnh đen.
- Hàm MaxValue(u, α, β) tính giá của đỉnh Trắng u .
- Hàm MinValue(u, α, β) tính giá của đỉnh Đen u .

Hàm gán giá trị max:

Function MaxValue(u, α, β);

Begin

If u là lá của cây hạn chế hoặc là đỉnh kết thúc

then MaxValue \leftarrow eval(u)

Else

for mỗi đỉnh v là con của u **do**

begin

$\alpha \leftarrow \max \{\alpha, \text{MinValue}(v, \alpha, \beta)\};$

if $\alpha > \beta$ **then** exit;

end;

MaxValue $\leftarrow \alpha$;

End;

5.4. Phương pháp alpha-beta (cont)

Hàm gán giá trị min:

Function MinValue(u, α, β);

Begin

If u là lá của cây hạn chế hoặc là đỉnh kết thúc

then MinValue \leftarrow eval(u)

Else

for mỗi đỉnh v là con của u **do**

begin

$\beta \leftarrow \min \{\beta, \text{.MaxValue}(v, \alpha, \beta)\};$

if $\alpha > \beta$ **then** exit;

end;

MinValue $\leftarrow \beta;$

End;

5.4. Phương pháp alpha-beta (cont)

Thủ tục Alpha-Beta: (tìm nước đi cho quân Trắng, v là đỉnh cần tối)

Procedure Alpha_Beta(u, v);

Begin

$\alpha \leftarrow -\infty;$

$\beta \leftarrow \infty;$

for mỗi đỉnh w là đỉnh con của u **do**

If $\alpha \leq \text{MinValue}(w, \alpha, \beta)$ **then**

begin

$\alpha = \text{MinValue}(w, \alpha, \beta);$

$v = w;$

end;

End;

NHẬP MÔN TRÍ TUỆ NHÂN TẠO

Chương 6:

Các bài toán thỏa ràng buộc

Biên soạn: TS Ngô Hữu Phúc

Bộ môn: Khoa học máy tính

Mobile: 098 56 96 580

Email: ngohuuphuc76@gmail.com

Thông tin chung

- Thông tin về nhóm môn học:

TT	Họ tên giáo viên	Học hàm	Học vị	Đơn vị công tác (Bộ môn)
1	Ngô Hữu Phúc	GVC	TS	BM Khoa học máy tính
2	Trần Nguyên Ngọc	GVC	TS	BM Khoa học máy tính
3	Hà Chí Trung	GVC	TS	BM Khoa học máy tính
4	Trần Cao Trường	GV	ThS	BM Khoa học máy tính

- Thời gian, địa điểm làm việc: Bộ môn Khoa học máy tính Tầng 2, nhà A1.
- Địa chỉ liên hệ: Bộ môn Khoa học máy tính, khoa Công nghệ thông tin.
- Điện thoại, email: 069-515-329, ngohuuphuc76.mta@gmail.com.

Cấu trúc môn học

- Chương 1: Giới thiệu chung.
- Chương 2: Logic hình thức.
- Chương 3: Các phương pháp tìm kiếm mù.
- Chương 4: Các phương pháp tìm kiếm có sử dụng thông tin.
- Chương 5: Các chiến lược tìm kiếm có đối thủ.
- Chương 6: Các bài toán thỏa ràng buộc.
- Chương 7: Nhập môn học máy.

Bài 6: Các bài toán thỏa ràng buộc

Chương 6, mục: 6.1 – 6.3

Tiết: 1-3; Tuần thứ: 8,9 (làm thực hành chương 5),10.

Mục đích, yêu cầu:

1. Nắm được khái niệm về thỏa ràng buộc.
2. Nắm được phương pháp tìm kiếm thỏa ràng buộc.
3. Xây dựng chương trình minh họa.

Hình thức tổ chức dạy học: Lý thuyết.

Thời gian: 3 tiết.

Địa điểm: Giảng đường do Phòng Đào tạo phân công

Nội dung chính: (Slides)

Nội Dung

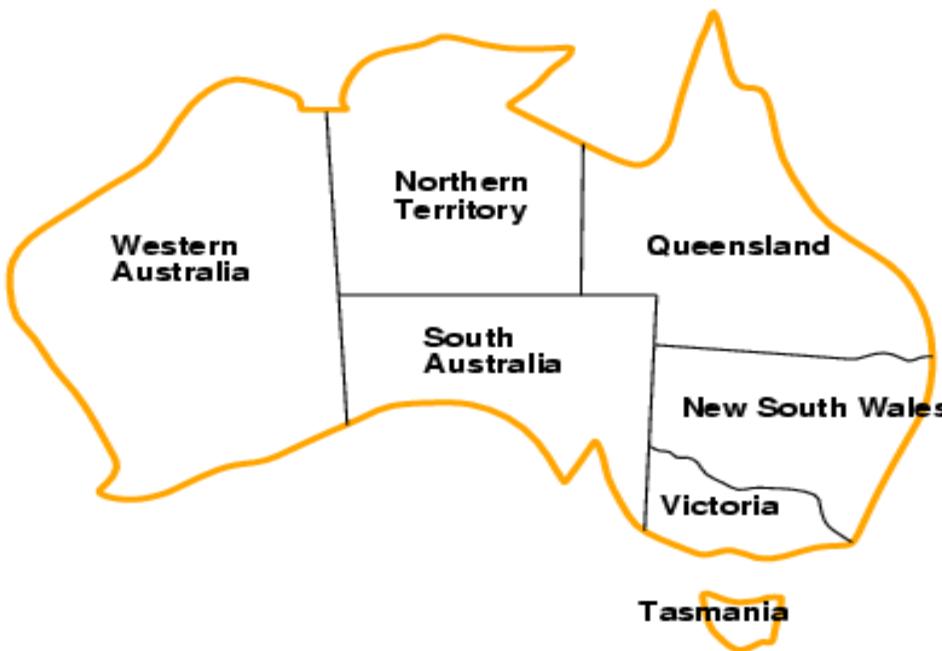
Tìm kiếm thỏa ràng buộc:

- Bài toán thỏa ràng buộc (CSP)
- Tìm kiếm backtracking cho CSP
- Tìm kiếm địa phương cho CSP

Bài toán thỏa ràng buộc (CSPs)

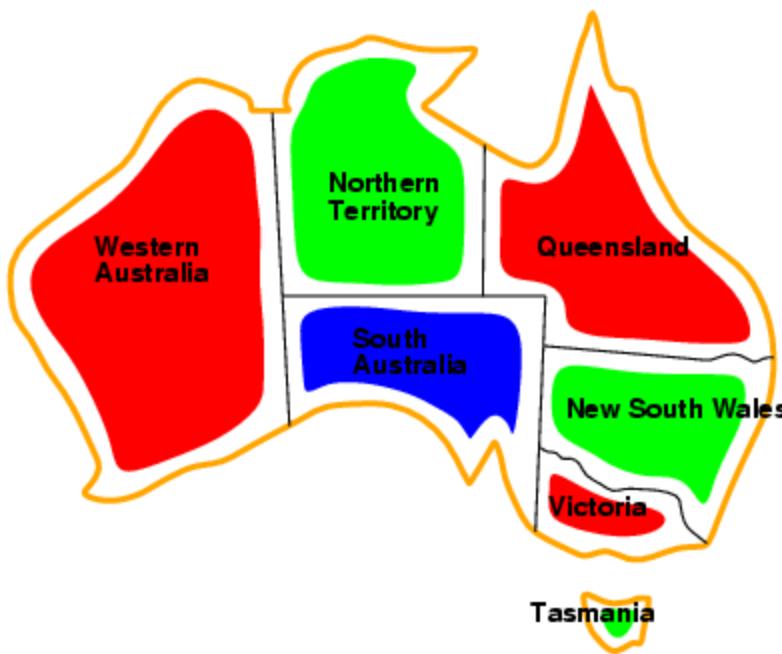
- Bài toán tìm kiếm ở tiết trước:
 - Trạng thái: dạng “hộp đen” – Tất cả các cấu trúc mà trên đó có thể định nghĩa successor function, heuristic function, and goal test
- CSP:
 - Trạng thái được định nghĩa là các biến X_i với giá trị lấy từ các miền xác định D_i
 - goal test là tập các ràng buộc quy định trên tổ hợp các giá trị của tập con của các biến
- Có thể đưa ra những thuật toán chung có công hiệu lớn hơn những tìm kiếm Heuristics ở tiết trước.

Ví dụ Tô Màu Đồ Thị



- Biển WA, NT, Q, NSW, V, SA, T
- Miền giá trị $D_i = \{\text{red}, \text{green}, \text{blue}\}$
- Ràng buộc: Các miền khác nhau được tô màu khác nhau
 - e.g., $WA \neq NT$, or $(WA, NT) \in \{(\text{red}, \text{green}), (\text{red}, \text{blue}), (\text{green}, \text{red}), (\text{green}, \text{blue}), (\text{blue}, \text{red}), (\text{blue}, \text{green})\}$
- Ví dụ bài toán: SAT, 3-SAT.

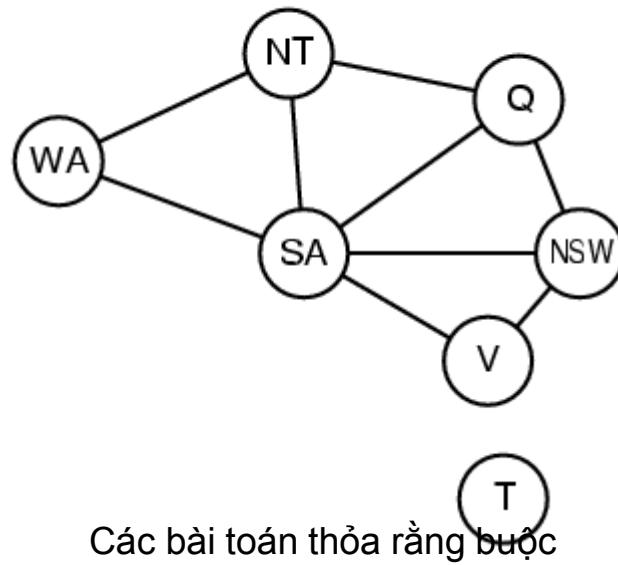
Ví Dụ



- Lời giải là các tổ hợp giá trị đủ và nhất quán với ràng buộc VD: WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green

Đồ Thị Ràng Buộc

- Bài toán CSP nhị phân: Mỗi ràng buộc chỉ gồm hai biến
- Đồ thị ràng buộc: đỉnh là các biến, cung biểu diễn các ràng buộc



Các biến thể của CSPs

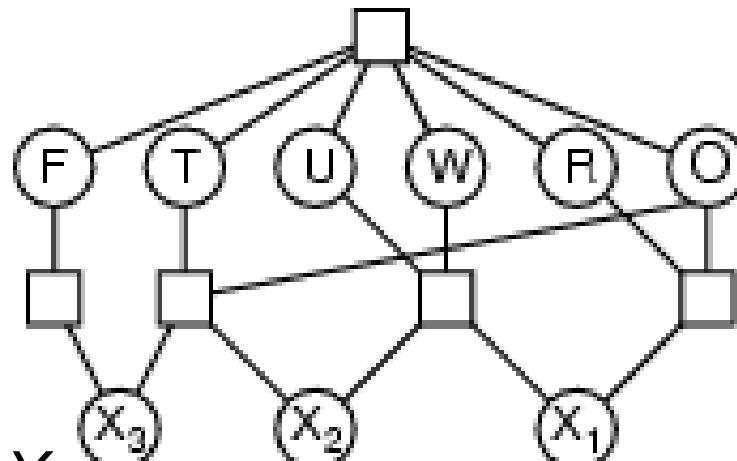
- Biến rời rạc
 - Miền giá trị hữu hạn:
 - n biến, kích thước miền giá trị $d \rightarrow O(d^n)$ tổ hợp giá trị
 - e.g., Boolean CSPs, như Boolean Satisfiability (NP-complete) –SAT (3SAT).
 - Miền giá trị vô hạn:
 - integers, strings, etc.
 - e.g., lập lịch, biến là ngày bắt đầu ngày kết thúc công việc.
 - ngôn ngữ biểu diễn ràng buộc $StartJob_1 + 5 \leq StartJob_3$
- Biến liên tục
 - Các bài toán quy hoạch tuyến tính và phi tuyến trong vận trù học.

Các Loại Ràng Buộc

- Ràng buộc đơn,
 - e.g., SA \neq green
- Ràng buộc nhị phân:
 - e.g., SA \neq WA
- Ràng buộc bậc cao (3 biến trở lên)
- e.g., bài toán cryptarithmetic (SEND+ME=MONEY).

Ví dụ: Cryptarithmetic

$$\begin{array}{r} \text{T} \ \text{W} \ \text{O} \\ + \ \text{T} \ \text{W} \ \text{O} \\ \hline \text{F} \ \text{O} \ \text{U} \ \text{R} \end{array}$$



- Biến: $F \ T \ U \ W \ O \ X_1 \ X_2 \ X_3$
- Miền giá trị: $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- Ràng buộc: *Alldiff* (F, T, U, W, R, O)
 - $O + O = R + 10 \cdot X_1$
 - $X_1 + W + W = U + 10 \cdot X_2$
 - $X_2 + T + T = O + 10 \cdot X_3$
 - $X_3 = F, T \neq 0, F \neq 0$

Các bài toán thỏa ràng buộc

Các bài toán CSPs trong thực tế

- Bài toán gán
 - e.g., Ai dạy lớp nào?
- Bài toán thời khoá biểu
 - Lớp nào học lúc nào ở đâu.
- Bài toán lập lịch xe (giao thông)
- Bài toán lập lịch gia công cho dây truyền sản xuất

Tìm kiếm mù

Không gian trạng thái là tập các biến được gán giá trị.

- Trạng thái đầu: { }
 - Successor function: gán giá trị cho một biến mà không gây mâu thuẫn với ràng buộc trên các giá trị đang được gán.
 - Thất bại nếu không tìm được giá trị nào như vậy
 - Goal test: Tập giá trị gán là đầy đủ.
-
1. Đối với mỗi bài toán CSPs:
 2. Lời giải tồn tại ở độ sâu tìm kiếm n nếu bài toán có n biến
 3. → dùng DFS
 4. Lời giải là trạng thái (không phải là đường đi)
 5. $b = (n - l)d$ tại độ sâu l , vì vậy có $n! \cdot d^n$ lá

Tìm kiếm Backtracking

- Chú ý: các cặp gán giá trị có tính giao hoan, i.e.,
[WA = red sau đó NT = green] sau đó [NT = green sau đó
WA = red]
- Chỉ cần gán giá trị cho 1 biến tại mỗi node
 $\rightarrow b = d$ có d^n lát
- DFS cho CSPs gán trị đơn biến – tìm kiếm backtracking
- Backtracking là dạng tìm kiếm mù đơn giản cho CSPs
- Có thể giải n -queens với $n \approx 25$

Backtracking search

```
function BACKTRACKING-SEARCH( csp) returns a solution, or failure
    return RECURSIVE-BACKTRACKING( {}, csp)
function RECURSIVE-BACKTRACKING( assignment, csp) returns a solution, or
failure
    if assignment is complete then return assignment
    var  $\leftarrow$  SELECT-UNASSIGNED-VARIABLE( Variables[csp], assignment, csp)
    for each value in ORDER-DOMAIN-VALUES( var, assignment, csp) do
        if value is consistent with assignment according to Constraints[csp] then
            add { var = value } to assignment
            result  $\leftarrow$  RECURSIVE-BACKTRACKING( assignment, csp)
            if result  $\neq$  failure then return result
            remove { var = value } from assignment
    return failure
```

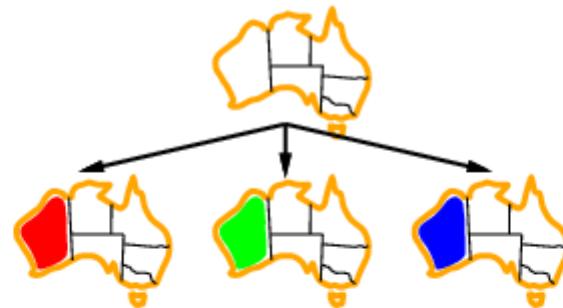
Ví Dụ Backtracking



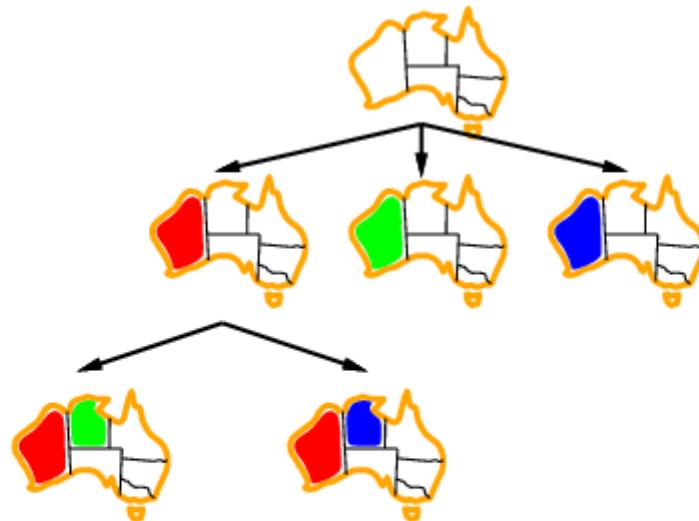
Các bài toán thỏa ràng buộc

17

Ví Dụ Backtracking



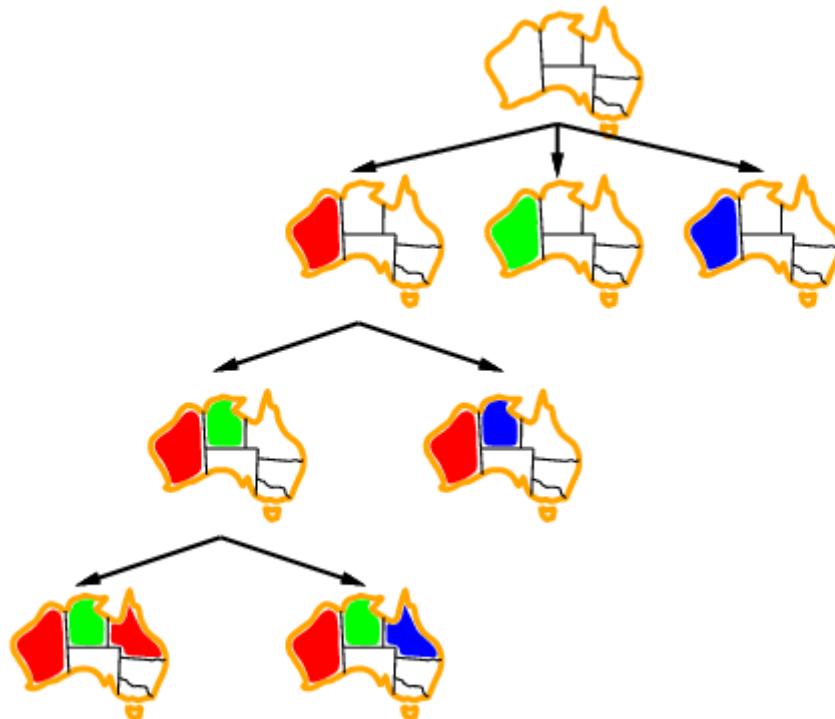
Ví Dụ Backtracking



Các bài toán thỏa ràng buộc

19

Ví Dụ Backtracking



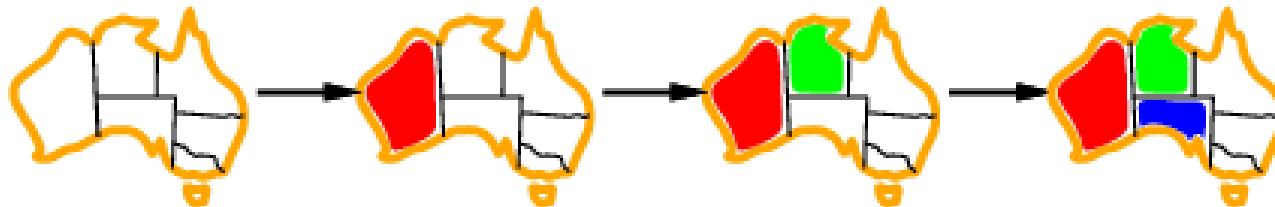
Cải thiện hiệu suất của backtracking

- Heuristic chung cần được đưa ra để tra lời các câu hỏi:
 - Biến nào được gán trị tiếp theo?
 - Các giá trị để thử theo thứ tự nào?
 - Liệu có thể biết trước những hướng vô vọng?

Việc trả lời tốt những câu hỏi trên có thể giúp tăng hiệu suất đáng kể Backtracking cho CSP trong thực tế!

Biến nhiều ràng buộc nhất

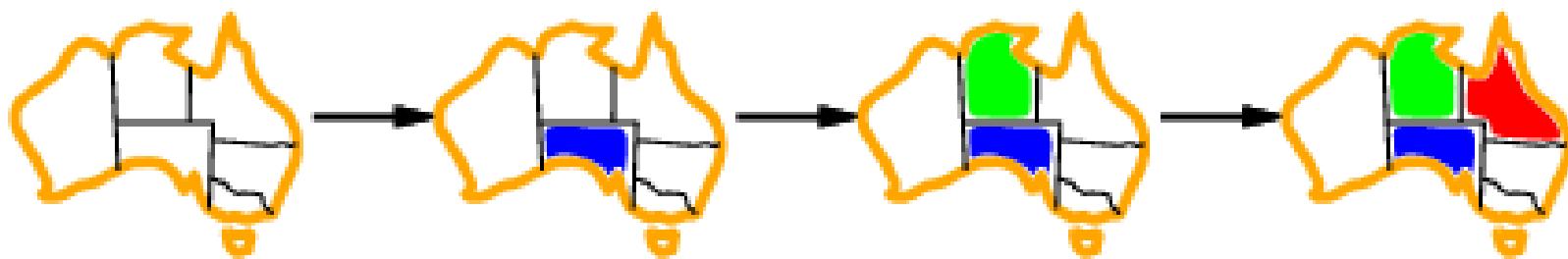
- chọn biến nhiều ràng buộc nhất:
chọn biến với ít giá trị hợp lệ để chọn nhất.



- minimum remaining values (MRV)
heuristic.

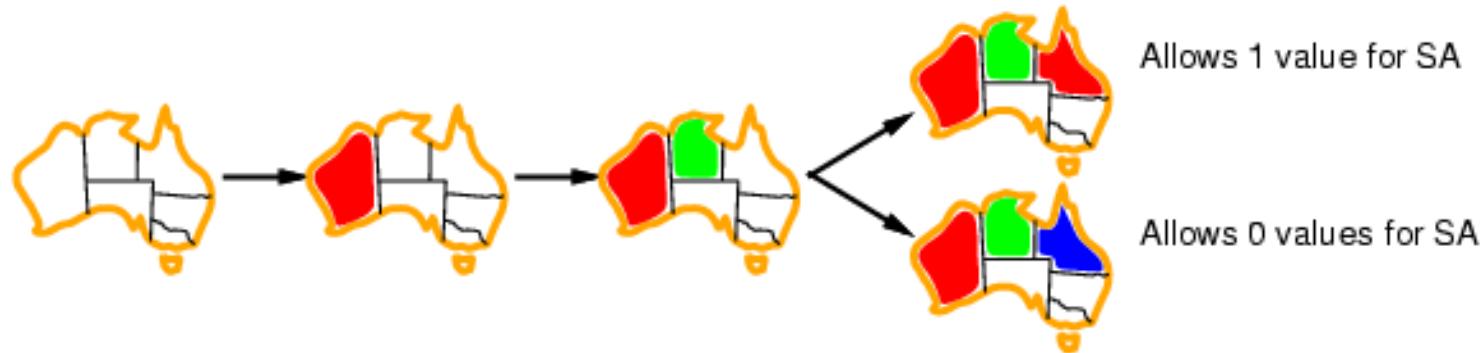
Biến ràng buộc nhiều nhất

- Trong trường hợp có nhiều biến như vậy:
- Chọn biến có nhiều ràng buộc với các biến khác nhất.



Chọn giá trị ràng buộc tối thiểu

- Đối với mỗi biến, chọn giá trị ràng buộc tối thiểu:
 - giá trị làm hạn chế ít nhất các giá trị cho các biến khác



- Kết hợp các heuristics chung nói trên có thể giúp giải bài toán 1000-queens

Kiểm Tra Trước

- Ý tưởng:
 - Lưu trữ và kiểm soát các giá trị có thể gán được cho các biến chưa được gán trị.
 - Kết thúc nếu có bất cứ biến nào không còn giá trị gán hợp lệ.



WA

NT

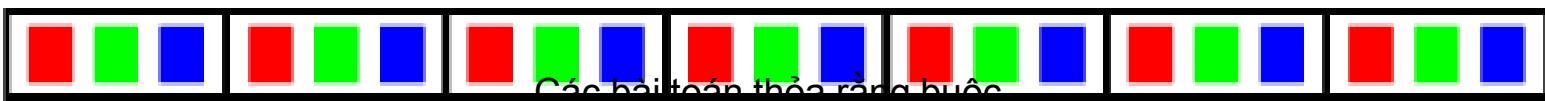
Q

NSW

V

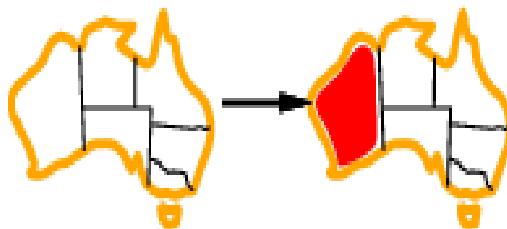
SA

T



Kiểm Tra Trước

- Ý tưởng:
 - Lưu trữ và kiểm soát các giá trị có thể gán được cho các biến chưa được gán trị.
 - Kết thúc nếu có bất cứ biến nào không còn giá trị gán hợp lệ.



WA NT Q NSW V SA T

Red	Green	Blue												
Red	White	Blue	Red	Green	Blue	Red	Green	Blue	Red	White	Blue	Red	Green	Blue

Các bài toán thỏa ràng buộc

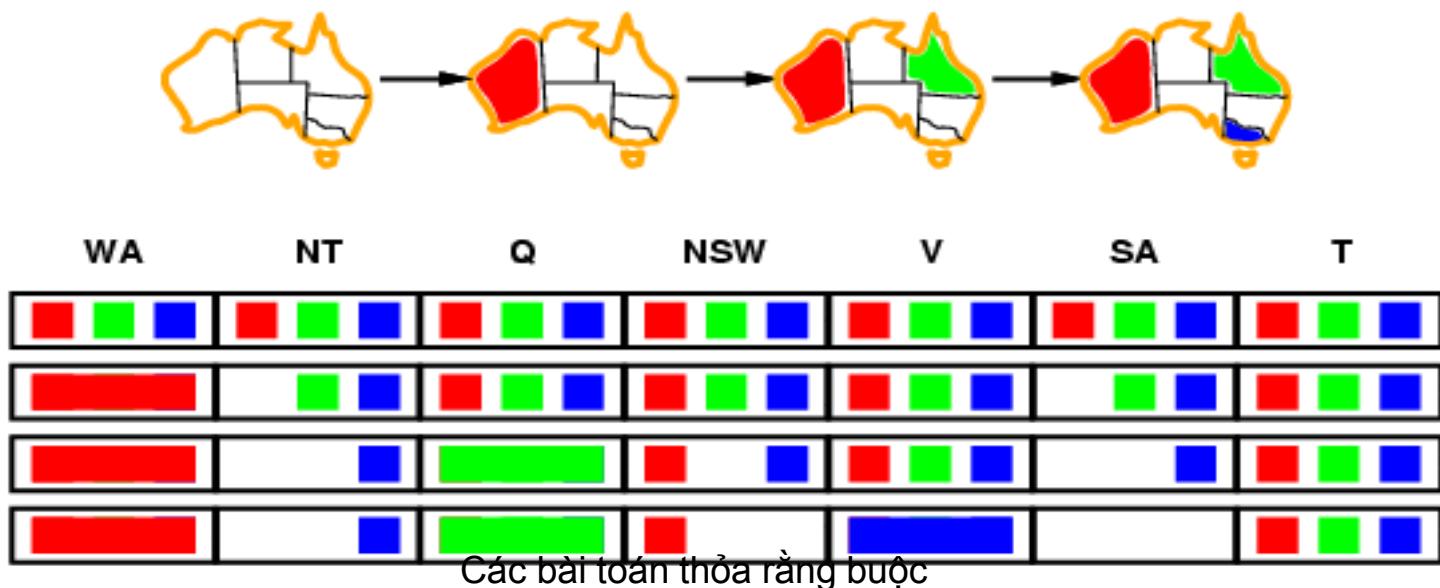
Kiểm Tra Trước

- Ý tưởng:
 - Lưu trữ và kiểm soát các giá trị có thể gán được cho các biến chưa được gán trị.
 - Kết thúc nếu có bất cứ biến nào không còn giá trị gán hợp lệ.



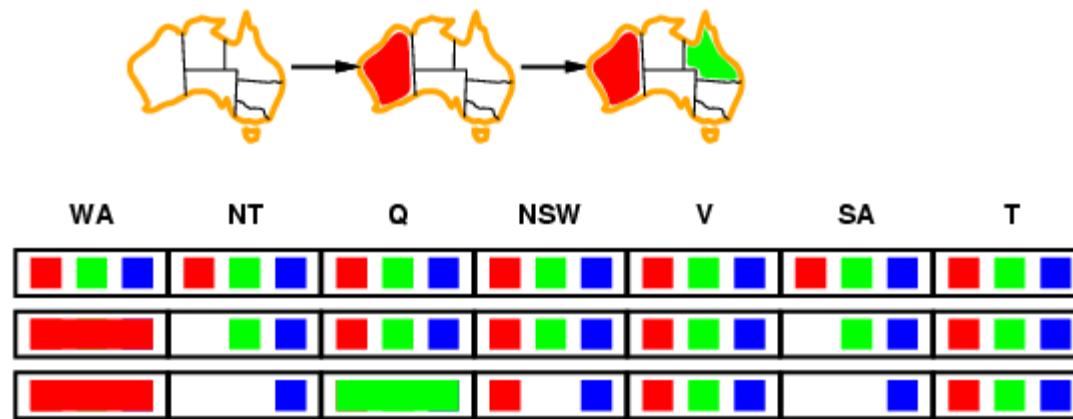
Kiểm Tra Trước

- Ý tưởng:
 - Lưu trữ và kiểm soát các giá trị có thể gán được cho các biến chưa được gán trị.
 - Kết thúc nếu có bất cứ biến nào không còn giá trị gán hợp lệ.



Lan Truyền Ràng Buộc

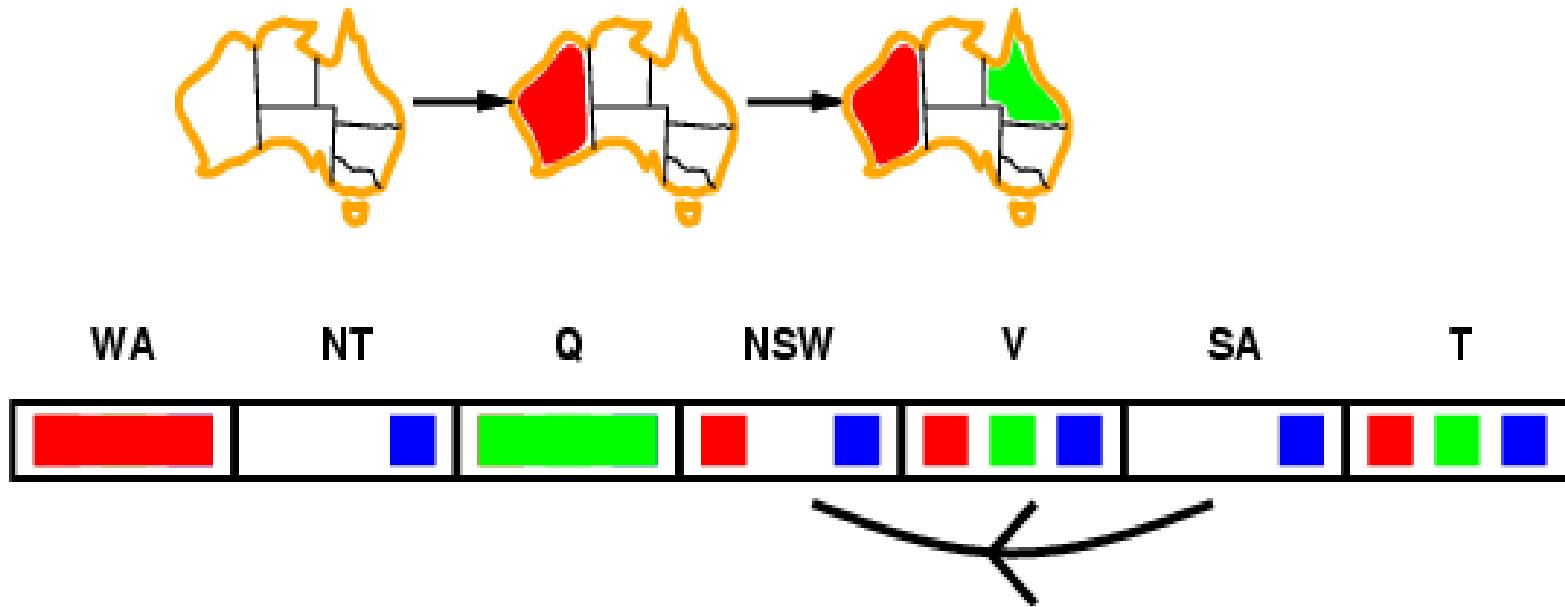
- kiểm tra trước lan truyền thông tin từ biến đã được gán sang biến chưa được gán, nhưng không phát hiện sớm được tất cả các trường hợp thất bại



- NT và SA không thể cùng blue!
- Lan truyền ràng buộc làm mạnh ràng buộc vào phạm vi địa phương

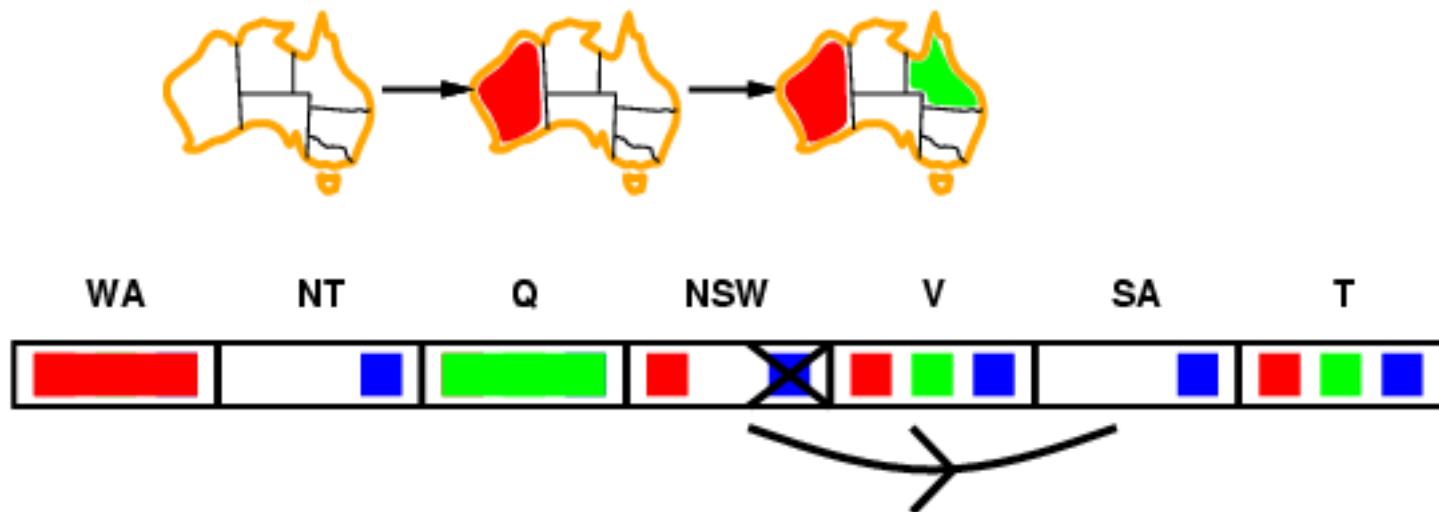
Nhất quán theo cung

- $X \rightarrow Y$ là nhất quán khi và chỉ khi với mọi giá trị x của X đều có giá trị hợp lệ cho y



Nhất quán theo cung

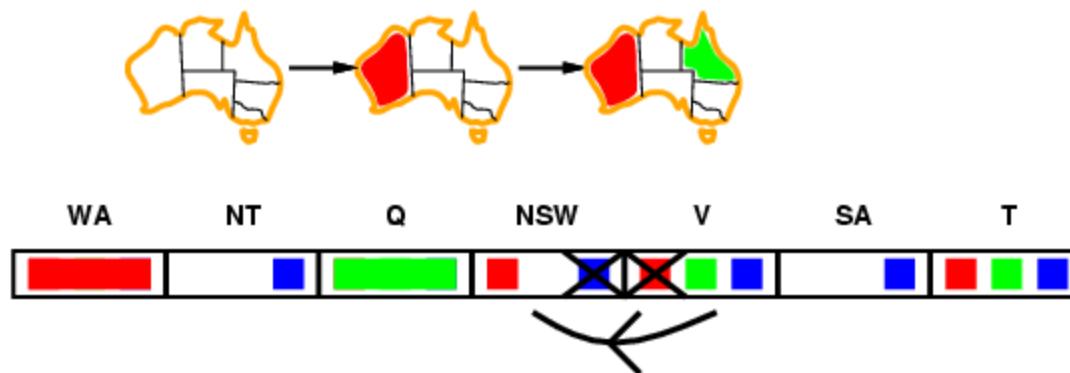
- $X \rightarrow Y$ là nhất quán khi và chỉ khi với mọi giá trị x của X đều có giá trị hợp lệ cho y
-



Các bài toán thỏa ràng buộc

Nhất quán theo cung

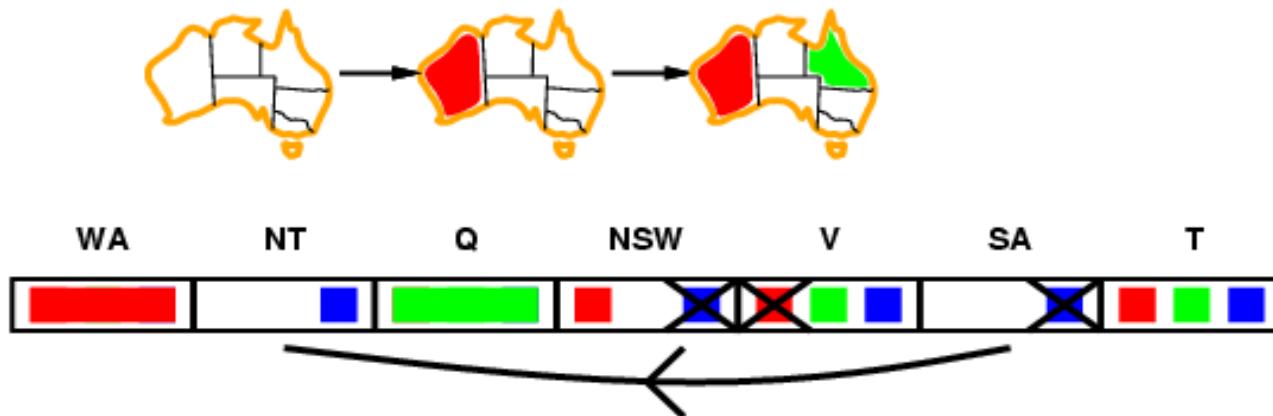
- $X \rightarrow Y$ là nhất quán khi và chỉ khi với mọi giá trị x của X đều có giá trị hợp lệ cho y



- Nếu X mất một giá trị, thì cần kiểm tra lại lân cận của X .

Nhất quán theo cung

- $X \rightarrow Y$ là nhất quán khi và chỉ khi với mọi giá trị x của X đều có giá trị hợp lệ cho y



- Nếu X mất một giá trị, thì cần kiểm tra lại lân cận của X .
- Nhất quán theo cung có khả năng phát hiện thất bại sớm hơn kiểm tra trước.
- Có thể được thực hiện trước mỗi khi gán trị cho biến.

Thuật toán AC-3

function AC-3(*csp*) **returns** the CSP, possibly with reduced domains

inputs: *csp*, a binary CSP with variables $\{X_1, X_2, \dots, X_n\}$

local variables: *queue*, a queue of arcs, initially all the arcs in *csp*

while *queue* is not empty **do**

$(X_i, X_j) \leftarrow \text{REMOVE-FIRST}(\text{queue})$

if RM-INCONSISTENT-VALUES(X_i, X_j) **then**

for each X_k in NEIGHBORS[X_i] **do**

 add (X_k, X_i) to *queue*

function RM-INCONSISTENT-VALUES(X_i, X_j) **returns** true iff remove a value

removed $\leftarrow \text{false}$

for each x in DOMAIN[X_i] **do**

if no value y in DOMAIN[X_j] allows (x, y) to satisfy constraint(X_i, X_j)

then delete x from DOMAIN[X_i]; **removed** $\leftarrow \text{true}$

return **removed**

- Độ phức tạp thời gian $O(n^2d^3)$

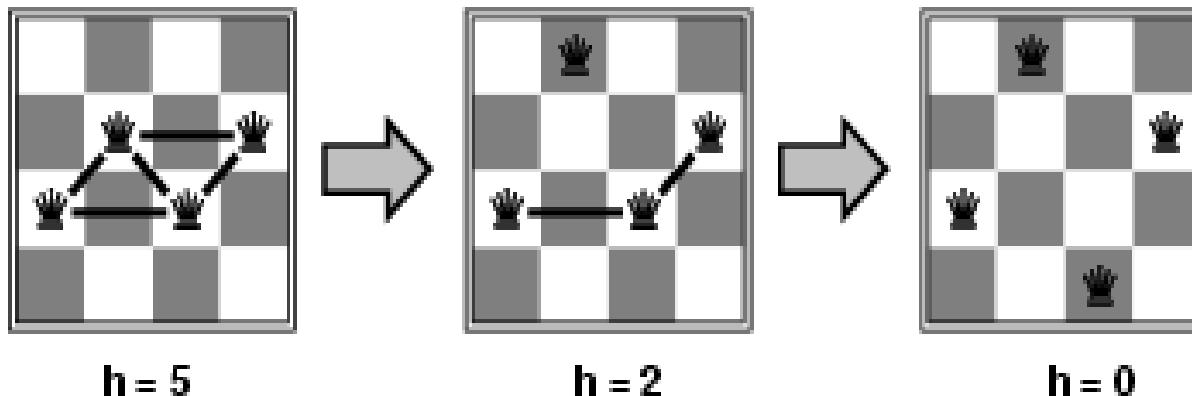
Các bài toán thỏa ràng buộc

Tìm kiếm địa phương cho CSPs

- Các phương pháp tìm kiếm địa phương (GHB, SA, GA, ACO,...) có thể áp dụng cho CSP
- Để áp dụng vào CSPs:
 - Cho phép các trạng thái không thỏa ràng buộc.
 - Toán tử để gán lại các giá trị cho biến
- Lựa chọn biến: lựa chọn ngẫu nhiên một biến xung đột
- Lựa chọn giá trị dựa trên heuristic min-conflict:
 - Chọn giá trị vi phạm ít ràng buộc nhất
 - i.e., hill-climb với $h(n)$ = số lượng ràng buộc bị vi phạm

Ví dụ: 4-Queens

- Trạng thái: 4 queens trong 4 cột ($4^4 = 256$ trạng thái)
- Toán tử: chuyển một con hậu trong cột
- Goal test: không con hậu nào tấn công nhau
- Đánh giá: $h(n) =$ Số lượng cặp hậu tấn công nhau



- Trạng thái đầu ngẫu nhiên, tìm kiếm địa phương có thể giúp giải bài toán n -queens với n rất cao (e.g., $n = 10,000,000$)

Các bài toán thỏa rằng buộc

Câu hỏi ôn tập

1. Phát biểu bài toán thỏa ràng buộc, tìm ví dụ.
2. Cài đặt thuật toán Backtracking, Kiểm tra trước, AC3,..
3. Giải các bài toán n-queen, map coloring, lập lịch,... bằng tìm kiếm địa phương và các thuật toán trên.

Đọc thêm:

- + Giáo trình chương 5.
- + OCW : ch3_csp_games1.
- + Tập hợp các bài toán NP-đầy đủ: *Computers and Intractability: A Guide to the Theory of NP-Completeness*, M.R.Garey and D.S. Johnson.
- + *Programming with constraints: An Introduction*, K. Marriott and P.J. Stuckey

NHẬP MÔN TRÍ TUỆ NHÂN TẠO

CHƯƠNG 7

Nhập môn học máy

Bộ môn Khoa học máy tính
Biên soạn: TS Ngô Hữu Phúc

Thông tin chung

- Thông tin về nhóm môn học:

TT	Họ tên giáo viên	Học hàm	Học vị	Đơn vị công tác (Bộ môn)
1	Ngô Hữu Phúc	GVC	TS	BM Khoa học máy tính
2	Trần Nguyên Ngọc	GVC	TS	BM Khoa học máy tính
3	Hà Chí Trung	GVC	TS	BM Khoa học máy tính
4	Trần Cao Trưởng	GV	ThS	BM Khoa học máy tính

- Thời gian, địa điểm làm việc: Bộ môn Khoa học máy tính Tầng 2, nhà A1.
- Địa chỉ liên hệ: Bộ môn Khoa học máy tính, khoa Công nghệ thông tin.
- Điện thoại, email: 069-515-329, ngohuuphuc76.mta@gmail.com.

Cấu trúc môn học

- Chương 1: Giới thiệu chung.
- Chương 2: Logic hình thức.
- Chương 3: Các phương pháp tìm kiếm mù.
- Chương 4: Các phương pháp tìm kiếm có sử dụng thông tin.
- Chương 5: Các chiến lược tìm kiếm có đối thủ.
- Chương 6: Các bài toán thỏa ràng buộc.
- Chương 7: Nhập môn học máy.

Bài 7: Nhập môn máy học

Chương 7, mục: 7.1 – 7.5

Tiết: 1-3; 4-6; Tuần thứ: 11,12 (xây dựng ứng dụng chương 6),13,14,15 (xây dựng ứng dụng chương 7) .

Mục đích, yêu cầu:

1. Nắm được khái niệm về máy học.
2. Nắm được phương pháp học quy nạp.
3. Nắm được phương pháp xây dựng cây quyết định.
4. Nắm được phương pháp học trong Mạng Neural.

Hình thức tổ chức dạy học: Lý thuyết.

Thời gian: 6 tiết.

Địa điểm: Giảng đường do Phòng Đào tạo phân công

Nội dung chính: (Slides)

Nội dung

7.1. Khái niệm về máy học? (4)

7.2. Học quy nạp.

7.3. Học với cây quyết định.

7.4. Học trong Mạng Neural.

- Mạng Perceptron.
- Mạng Perceptron đa lớp với thuật giải BP.

7.1. Khái niệm về máy học (1/4)

- Nhớ và làm lại (học vẹt).
- Học nhờ quan sát và khái quát hoá (học hiểu).
- Định nghĩa của H. Simon.

“Học là **sự thay đổi thích ứng** trong hệ thống giúp cho hệ thống có thể xử lý vấn đề ngày càng **hiệu quả hơn** khi gặp lại những tình huống tương tự”

7.1. Khái niệm về máy học (2/4)

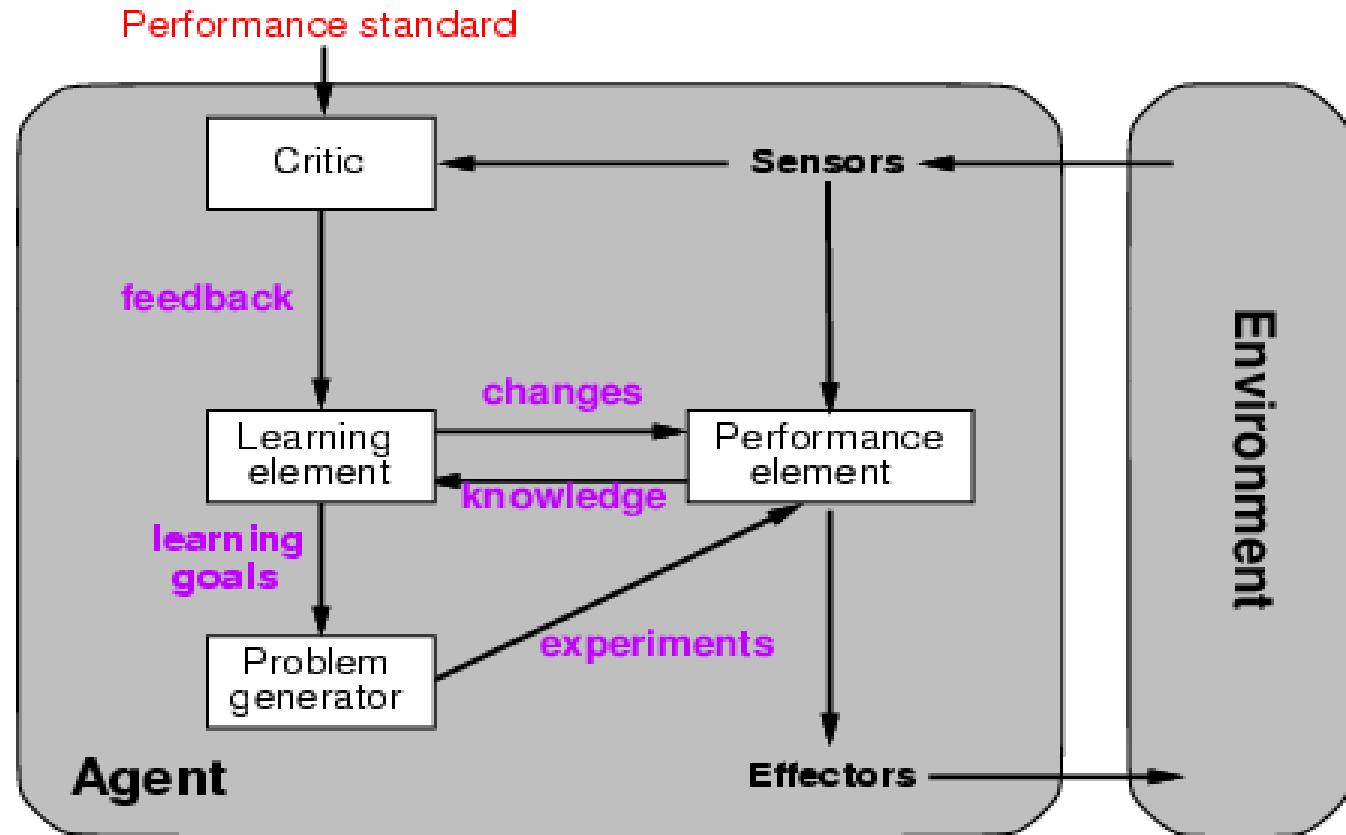
Học làm gì?

- Học là cần thiết trong những môi trường chưa quen thuộc,
- Học là một phương pháp hữu hiệu để xây dựng hệ thống
- Học là cách để các chương trình thông minh có thể hiệu chỉnh hành vi để tăng hiệu quả giải quyết vấn đề.

7.1. Khái niệm về máy học (3/4)

Học của tác tử:

- Environment: Môi trường.
- Sensors: cảm biến.
- Critic: phân tích, đánh giá.
- Effectors:



7.1. Khái niệm về máy học (4/4)

Xây dựng module học cho hệ thống:

- Việc xây dựng module học cho hệ thống phải tính đến yếu tố:
 - Phần nào cần học để tăng hiệu năng giải quyết vấn đề.
 - Thông tin phản hồi đối với phần học của hệ thống.
 - Cách biểu diễn tri thức cần học.
- Dạng thông tin phản hồi:
 - Học có giám sát**: trả lời chính xác cho các câu hỏi.
 - Học không giám sát**: không có câu trả lời chính xác.
 - Học tăng cường**: giành lấy phần thưởng nếu làm đúng.

7.2. Học quy nạp

- Ví dụ: học một hàm từ mẫu ví dụ.

f là hàm mục tiêu

Một mẫu ví dụ là một cặp ($x, f(x)$)

Bài toán: Tìm giả thuyết h

sao cho $h \approx f$

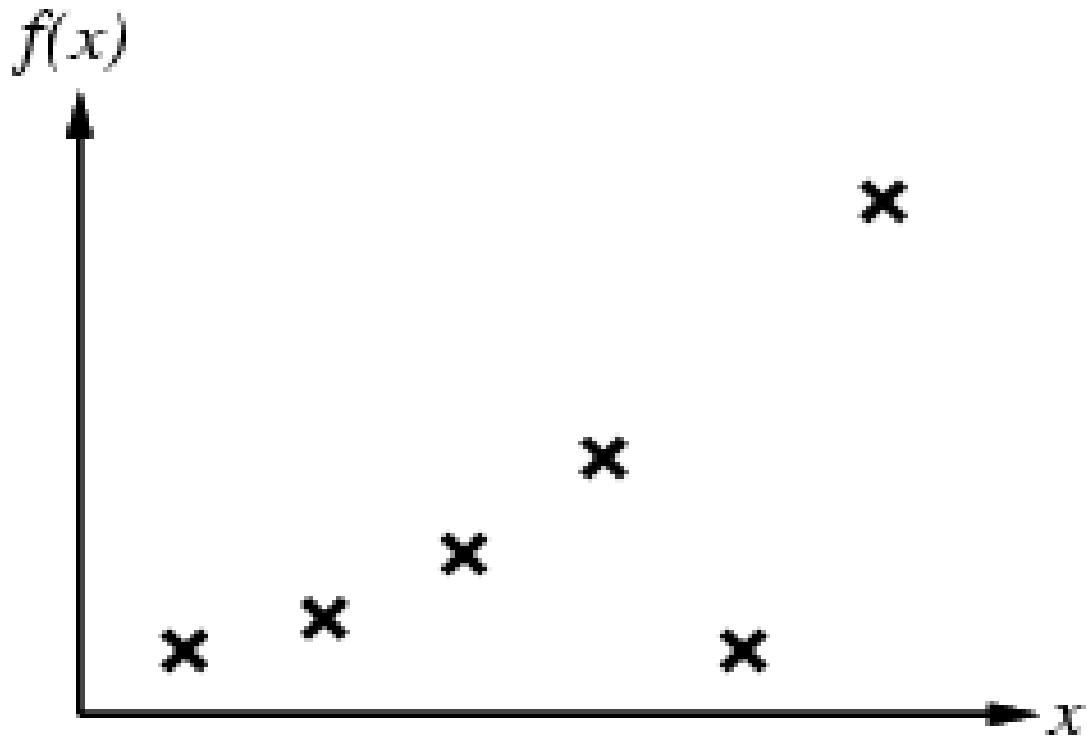
dựa trên tập mẫu cho trước

Mô hình đơn giản hóa việc học:

- Không tính đến tri thức có sẵn
- Giả sử tập mẫu là có đủ.

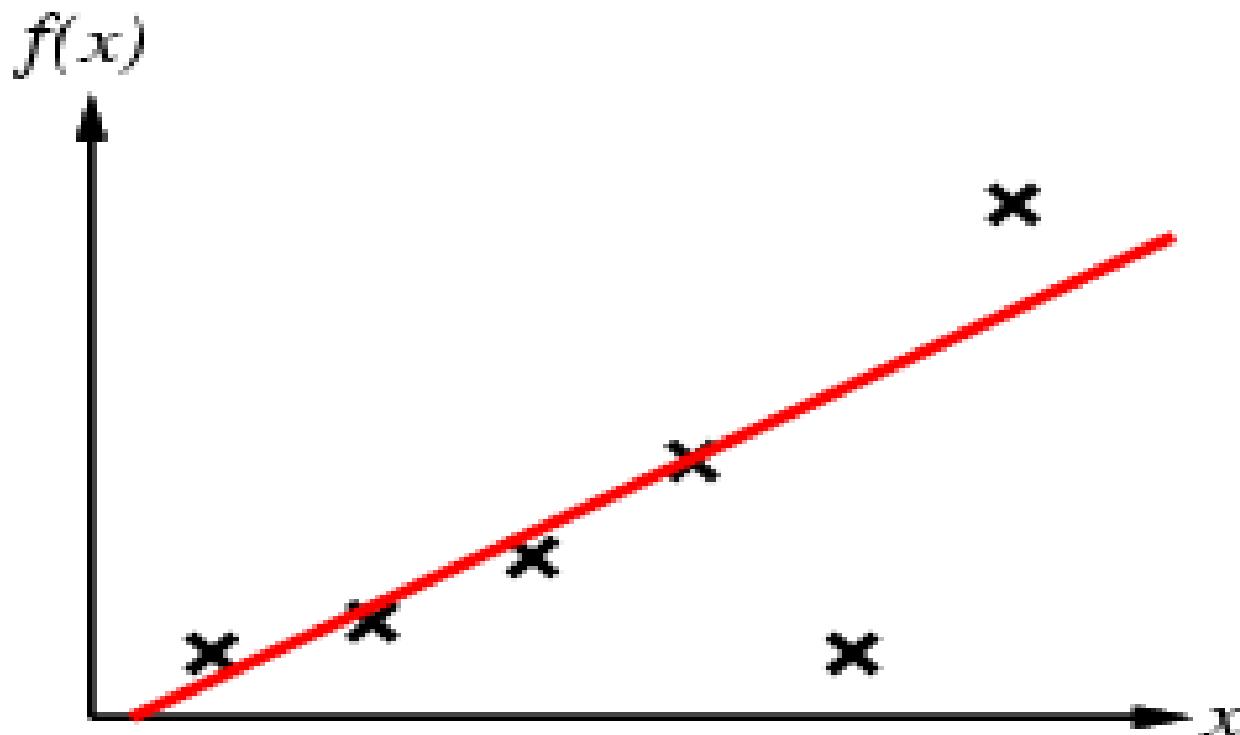
Phương pháp học quy nạp

- Xây dựng h gần với f trên tập huấn luyện
- (h được gọi là nhất quán với f trên tập mẫu)
- E.g., khớp đường cong:



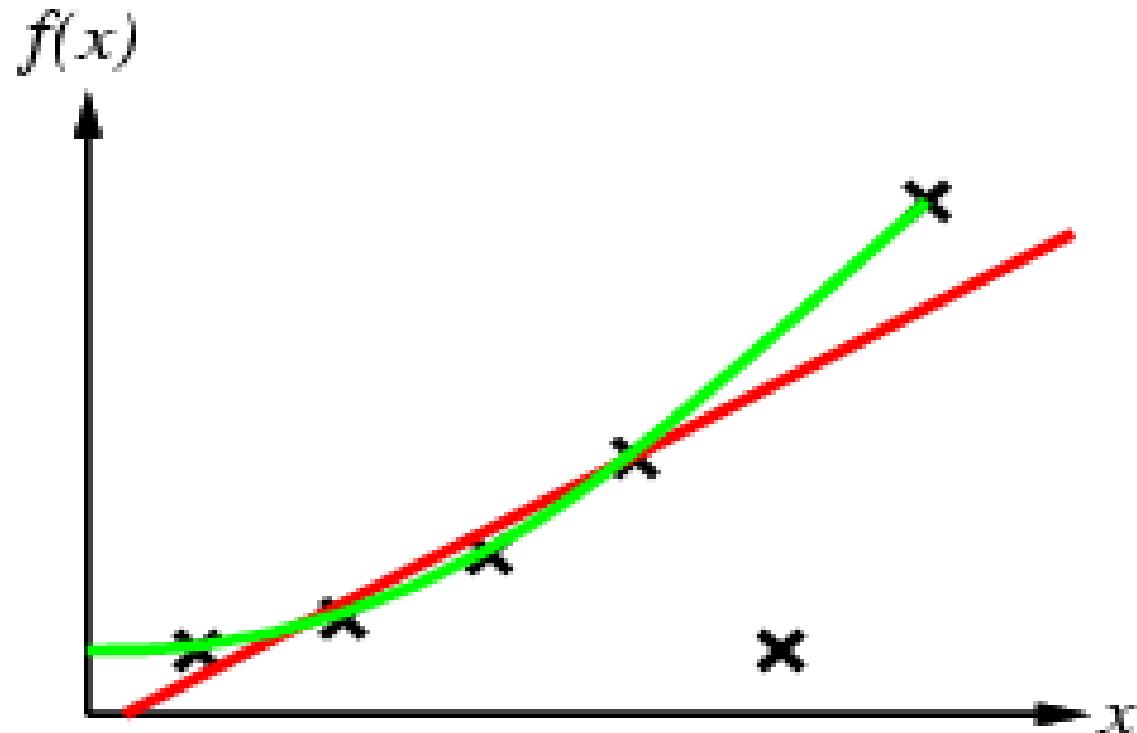
Phương pháp học quy nạp

- Xây dựng h gần với f trên tập huấn luyện
- (h được gọi là nhất quán với f trên tập mẫu)
- E.g., khớp đường cong:



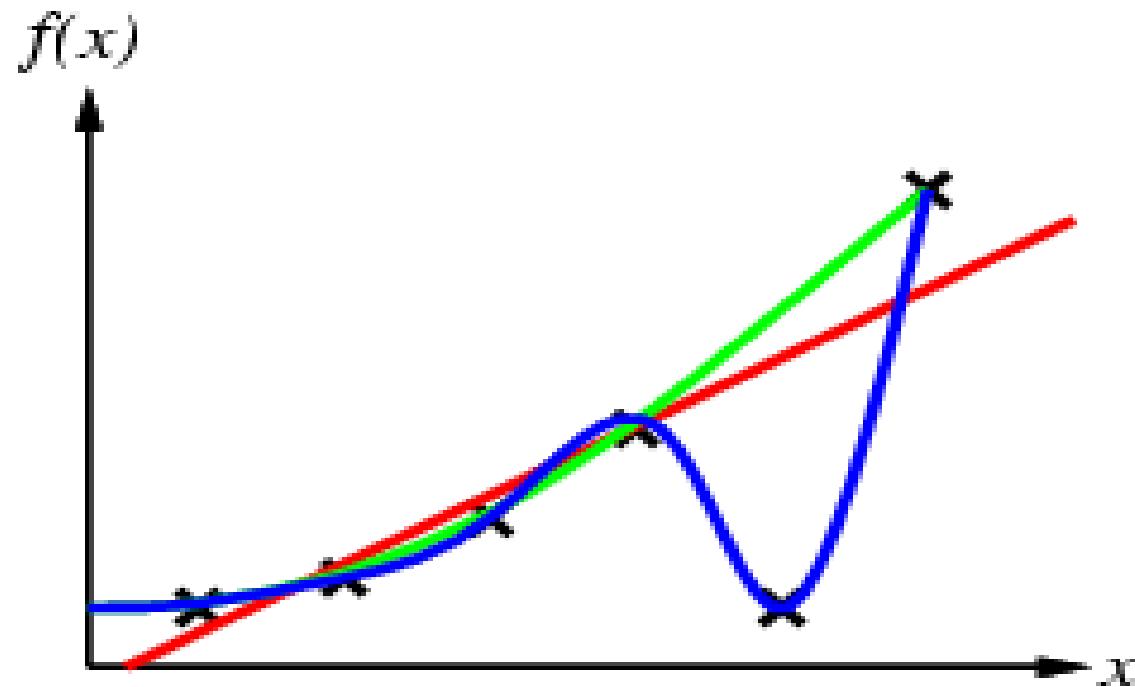
Phương pháp học quy nạp

- Xây dựng h gần với f trên tập huấn luyện
- (h được gọi là nhất quán với f trên tập mẫu)
- E.g., khớp đường cong:



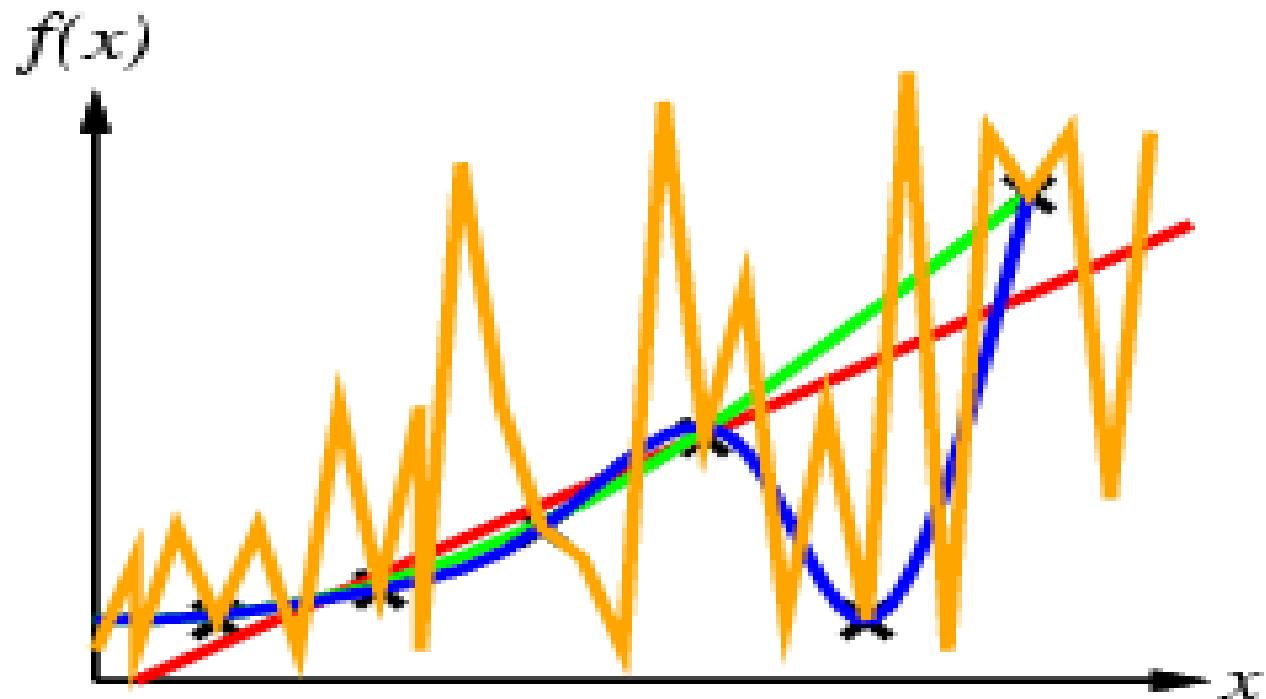
Phương pháp học quy nạp

- Xây dựng h gần với f trên tập huấn luyện
- (h được gọi là nhất quán với f trên tập mẫu)
- E.g., khớp đường cong:



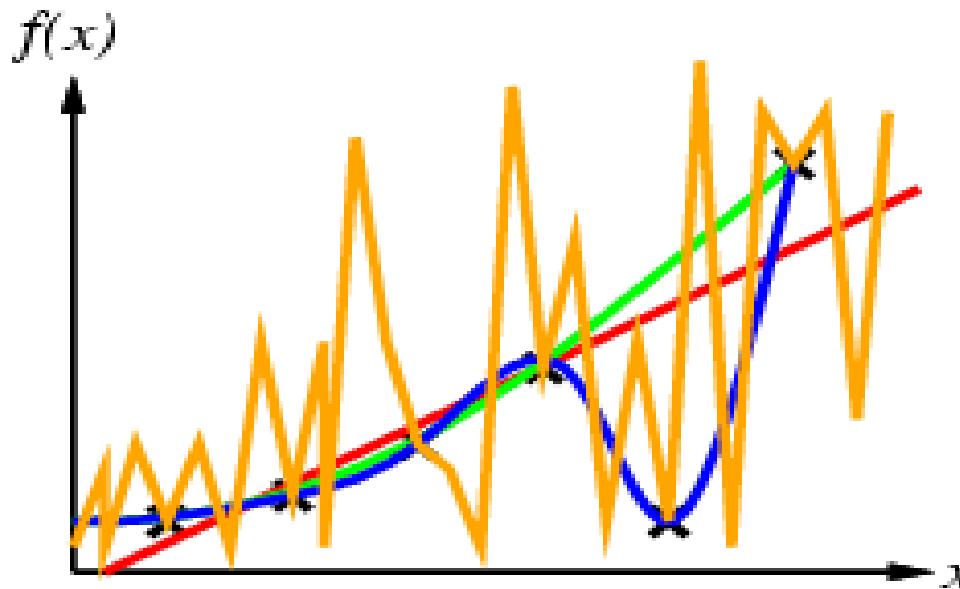
Phương pháp học quy nạp

- Xây dựng h gần với f trên tập huấn luyện
- (h được gọi là nhất quán với f trên tập mẫu)
- E.g., khớp đường cong:



Phương pháp học quy nạp

- Xây dựng h gần với f trên tập huấn luyện
- (h được gọi là nhất quán với f trên tập mẫu)
- E.g., khớp đường cong:



- Ockham's razor: ưu tiên những giả thiết nào xấp xỉ tốt hàm mục tiêu và càng đơn giản càng tốt

7.3. Học các cây quyết định

Bài toán: Học xem khi nào thì nên ngồi bàn đợi tại một restaurant:

1. Alternate: Có restaurant nào cạnh đây không?
2. Bar: Liệu có khu vực quầy bar có thể ngồi không?
3. Fri/Sat: hôm nay là thứ 6 hay thứ 7?
4. Hungry: có đang đói không?
5. Patrons: Số người trong restaurant (None, Some, Full)
6. Price: khoảng giá (\$, \$\$, \$\$\$)
7. Raining: ngoài trời có mưa không?
8. Reservation: đã đặt trước chưa?
9. Type: loại restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: thời gian chờ đợi (0-10, 10-30, 30-60, >60)

Biểu diễn thuộc tính giá trị

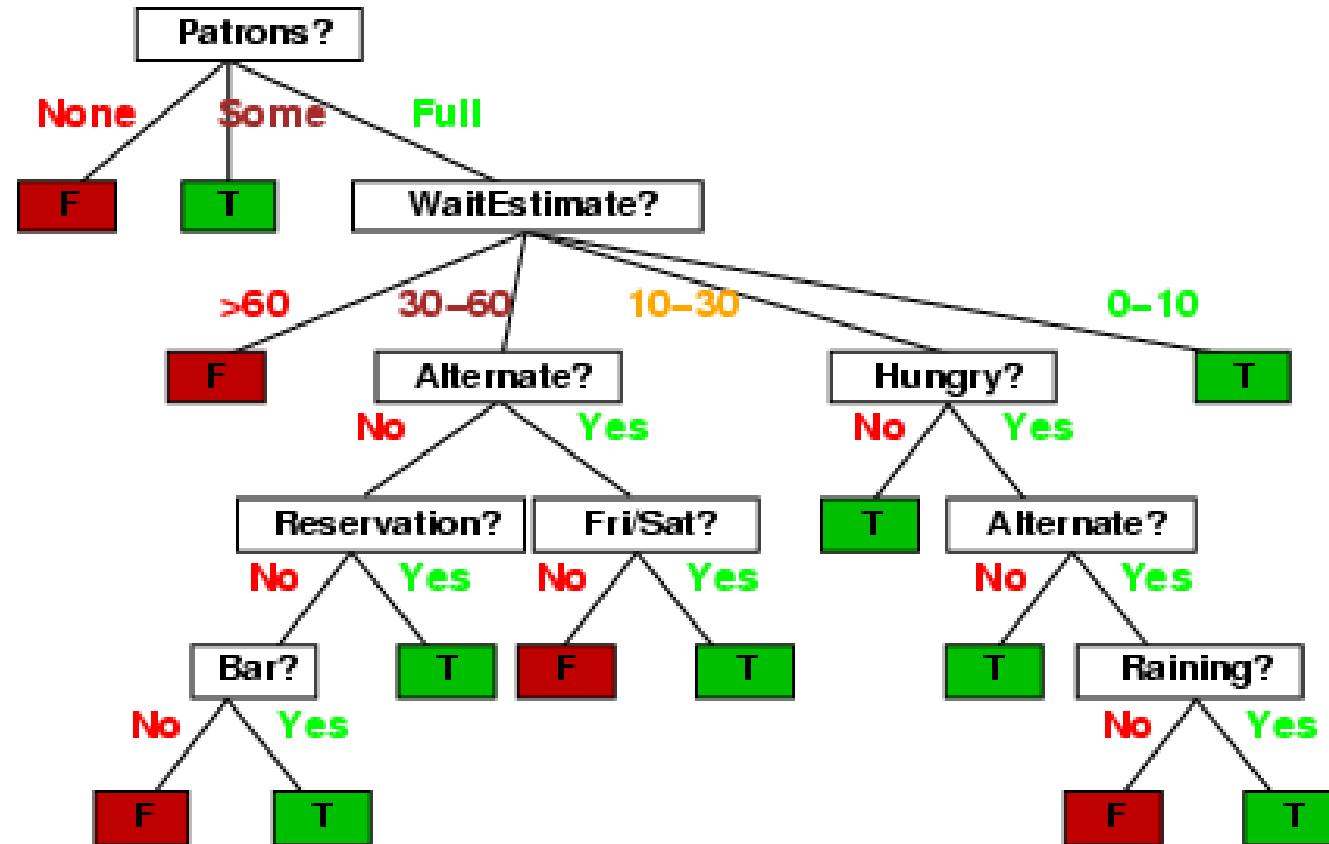
- Các mẫu được biểu diễn bằng các thuộc tính và giá trị (Boolean, discrete, continuous)

Example	Attributes										Target Wait
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0–10	T
X_2	T	F	F	T	Full	\$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	\$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	\$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X_6	F	T	F	T	Some	\$\$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	\$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	\$\$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	\$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	\$\$\$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	\$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	\$	F	F	Burger	30–60	T

- Nhiệm vụ đặt ra là phân loại xem trường hợp nào trong tương lai là **positive** (T) hay **negative** (F)

Cây quyết định

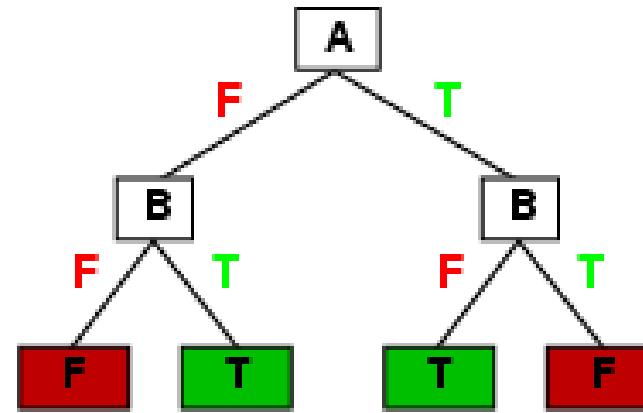
- Biểu diễn giả thiết cần học.
- Ví dụ:



Khả năng biểu diễn

- Cây quyết định có khả năng dùng để biểu diễn bất cứ hàm nào.
- E.g. hàm Boolean:

A	B	$A \text{ xor } B$
F	F	F
F	T	T
T	F	T
T	T	F



- Với một cây quyết định nhất quán với tập mẫu huấn luyện thì mỗi input, output của hàm tương ứng với một đường đi trong cây. Nhưng cũng có thể khả năng khái quát hoá không cao đối với các ví dụ mới chưa biết.
- Ưu tiên tìm cây có độ phức tạp nhỏ.

Không gian giả thuyết

Số lượng cây quyết định cho hàm Boolean =

= Số lượng hàm boolean

= số lượng bảng luận ý với 2^n hàng = 2^{2^n}

- E.g., nếu có 6 thuộc tính Boolean, có 18,446,744,073,709,551,616 cây

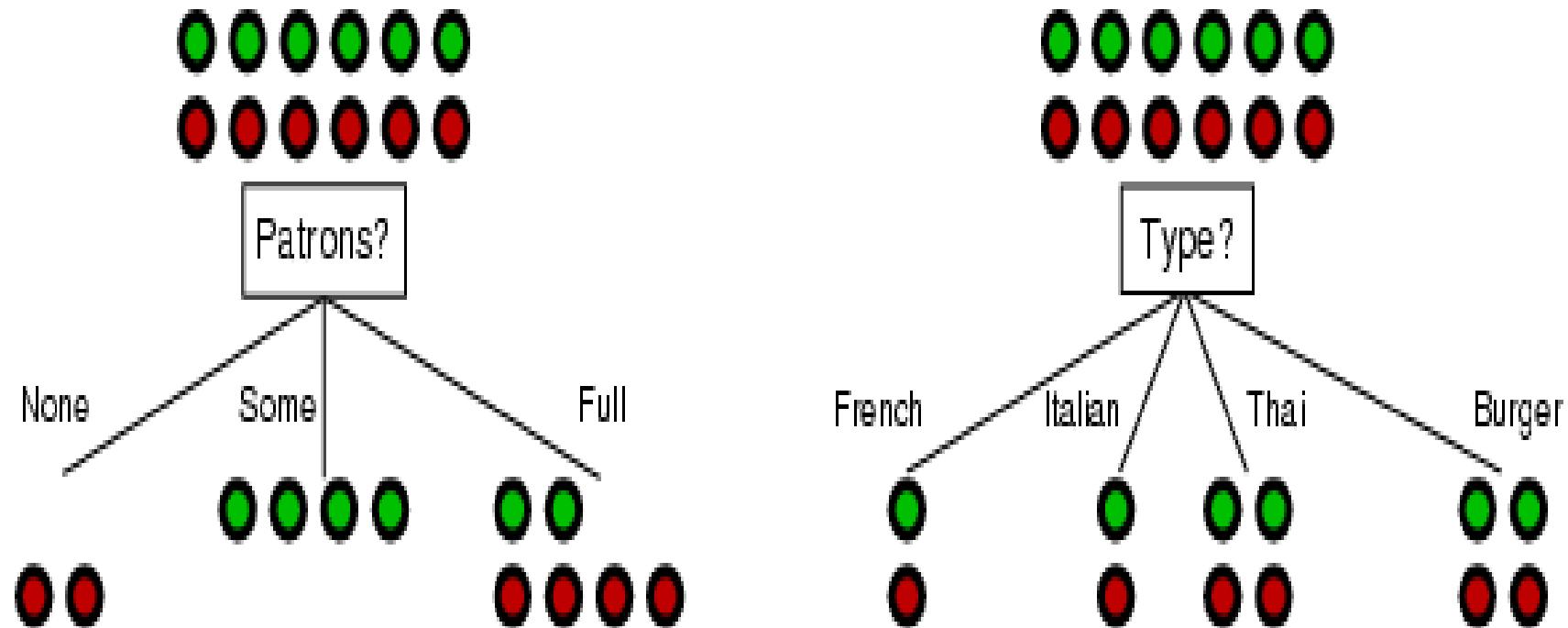
Thuật toán học cây quyết định

- Mục đích: Tìm cây nhỏ nhất quát với tập mẫu huấn luyện.
- Ý tưởng: Tìm kiếm heuristic chọn thuộc tính quan trọng nhất để phân tách (đệ quy)

```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best  $\leftarrow$  CHOOSE-ATTRIBUTE(attributes, examples)
        tree  $\leftarrow$  a new decision tree with root test best
        for each value vi of best do
            examplesi  $\leftarrow$  {elements of examples with best = vi}
            subtree  $\leftarrow$  DTL(examplesi, attributes – best, MODE(examples))
            add a branch to tree with label vi and subtree subtree
        return tree
```

Chọn thuộc tính

- Ý tưởng: chọn thuộc tính (giá trị) sao cho nó giúp phân tách tập mẫu thành hai tập thuần khiết (chỉ có positive hay chỉ có negative).



- Patrons?* là lựa chọn tốt hơn

Sử dụng lý thuyết thông tin

- để cài đặt Choose-Attribute trong thuật toán DTL:
- Lượng thông tin (Entropy):
$$I(P(v_1), \dots, P(v_n)) = \sum_{i=1} -P(v_i) \log_2 P(v_i)$$
- Đối với tập có p mẫu positive và n negative:

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}$$

Lợi thông tin (Information gain)

- chọn thuộc tính A chia tập huấn luyện E thành các tập con E_1, \dots, E_v tính theo giá trị của A , và giả sử A có v giá trị khác nhau.

$$remainder(A) = \sum_{i=1}^v \frac{p_i + n_i}{p + n} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

- Lợi thông tin (IG) là độ giảm trong entropy trong việc test thuộc tính:

$$IG(A) = I\left(\frac{p}{p + n}, \frac{n}{p + n}\right) - remainder(A)$$

- Chọn thuộc tính có IG lớn nhất

Lợi thông tin (Information gain)

Trong tập mẫu của ví dụ, $p = n = 6$, $I(6/12, 6/12) = 1$ bit

Xét thuộc tính *Patrons* và *Type* (và các thuộc tính khác):

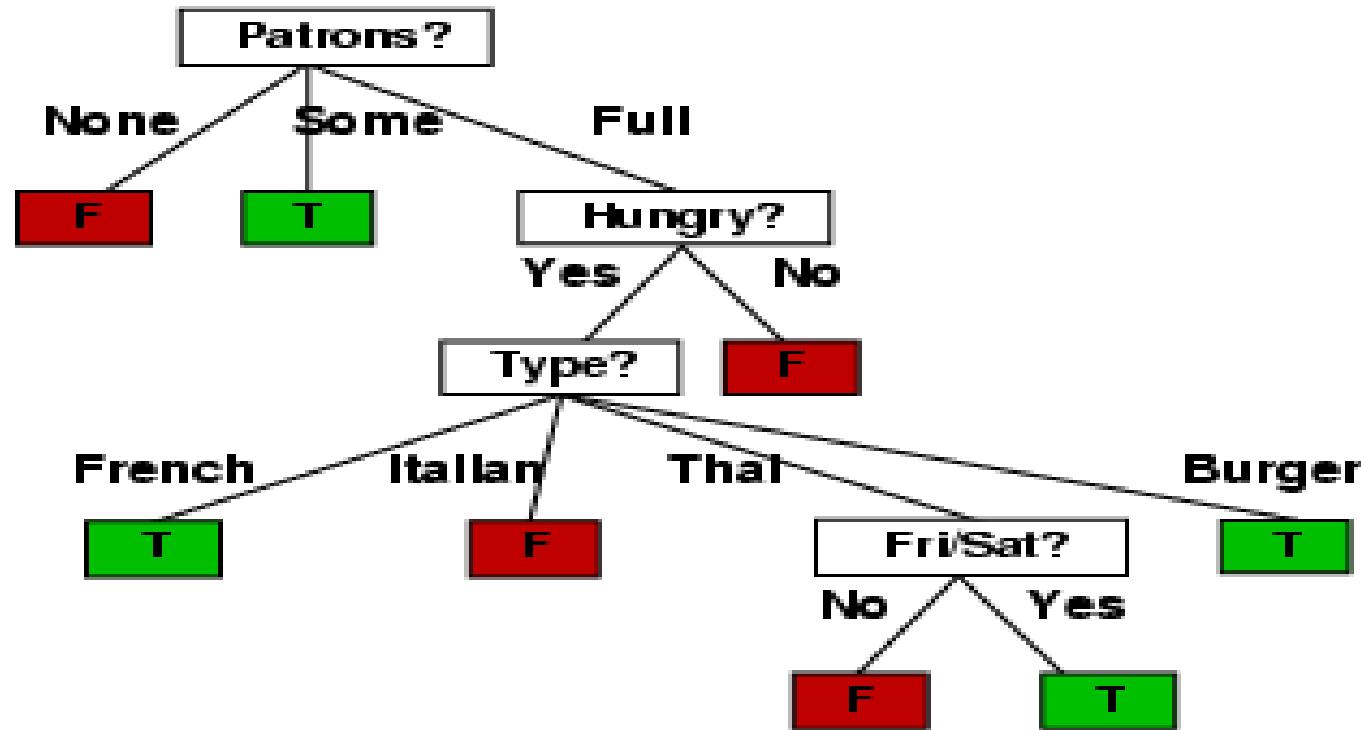
$$IG(Patrons) = 1 - \left[\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I\left(\frac{2}{6}, \frac{4}{6}\right) \right] = .541 \text{ bits}$$

$$IG(Type) = 1 - \left[\frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) \right] = 0 \text{ bits}$$

Patrons có giá trị IG cao nhất nên được DTL chọn làm gốc của cây quyết định.

Lợi thông tin (Information gain)

- Cây quyết định học bởi DTL từ 12 ví dụ:

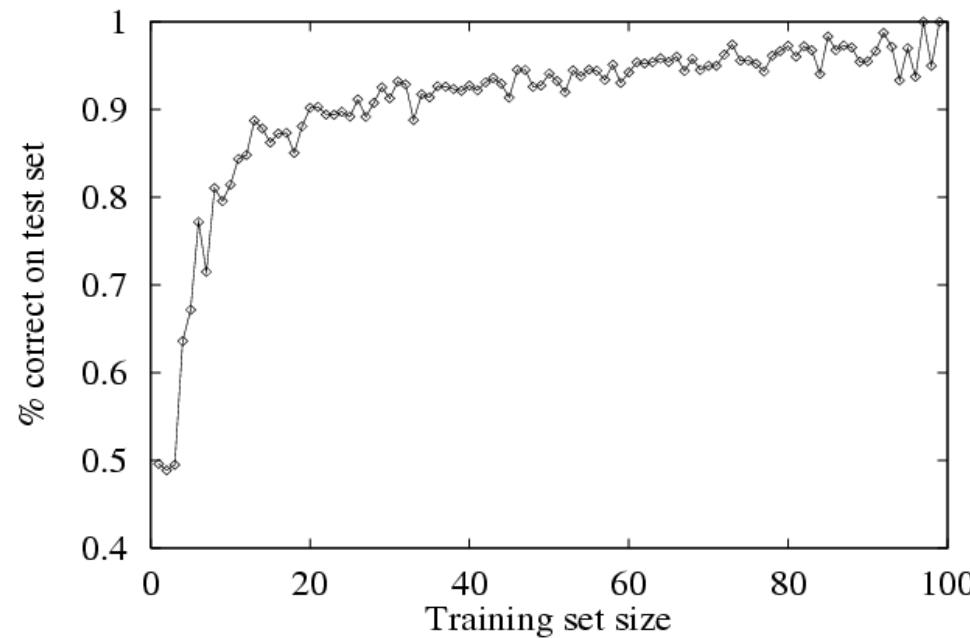


- Nhỏ hơn cây quyết định đưa ra lúc đầu

Khi nào học tốt?

- Làm sao chắc rằng $h \approx f$?
 1. sử dụng các kết quả trong thống kê và học thống kê.
 2. Thủ h trên tập ví dụ mới (test set)

Learning curve = % Số lượng đoán đúng trên tập test khi kích thước tập huấn luyện tăng lên.



Đọc thêm

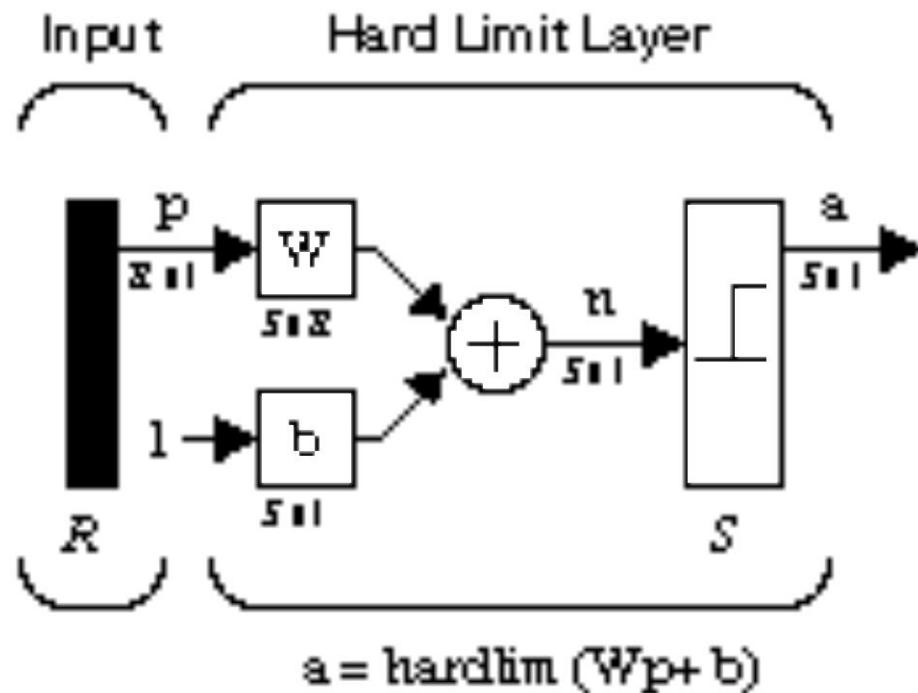
- Giáo trình: chương 18 (phần 1-3).
- MIT Open courseware: ch5, ch6, ch7.
- T. Mitchell, *Machine Learning*, McGraw-Hill.
- J.R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann.

Câu hỏi ôn tập

1. Định nghĩa việc học?
2. Cho biết các loại học khác nhau?
3. Cho biết các dạng học trong học máy?
4. Cấu trúc cây quyết định?
5. Cài đặt thuật toán DTL.
6. Dùng C4.5, hoặc thuật toán DTL để giải các bài toán về Data Mining.

PERCEPTRON ĐƠN LỚP

Cấu trúc Perceptron



$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

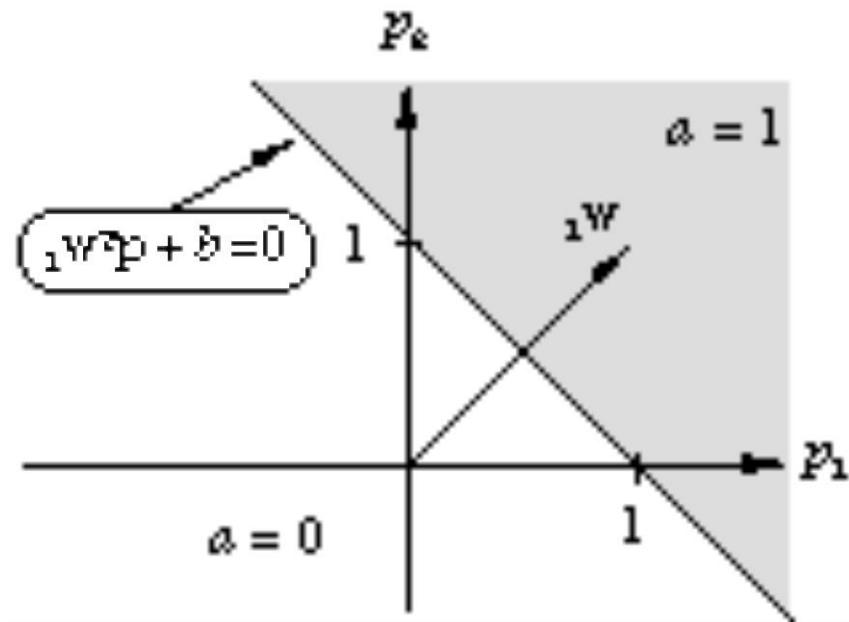
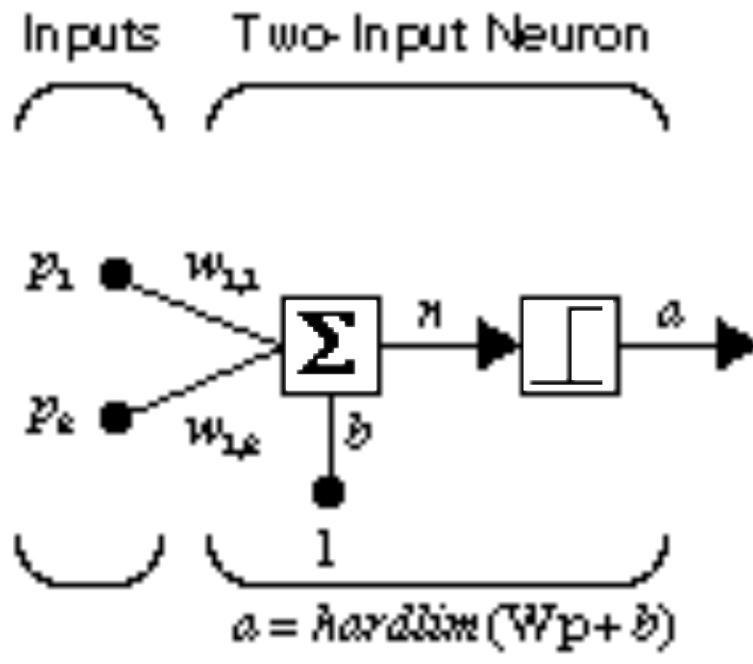
$$iW = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix}$$

$$W = \begin{bmatrix} 1^T W \\ 2^T W \\ \vdots \\ S^T W \end{bmatrix}$$

$$a_i = \text{hardlim}(n_i) = \text{hardlim}(i^T W p + b_i)$$

Perceptron đơn lớp một đầu ra

$$w_{1,1} = 1 \quad w_{1,2} = 1 \quad b = -1$$



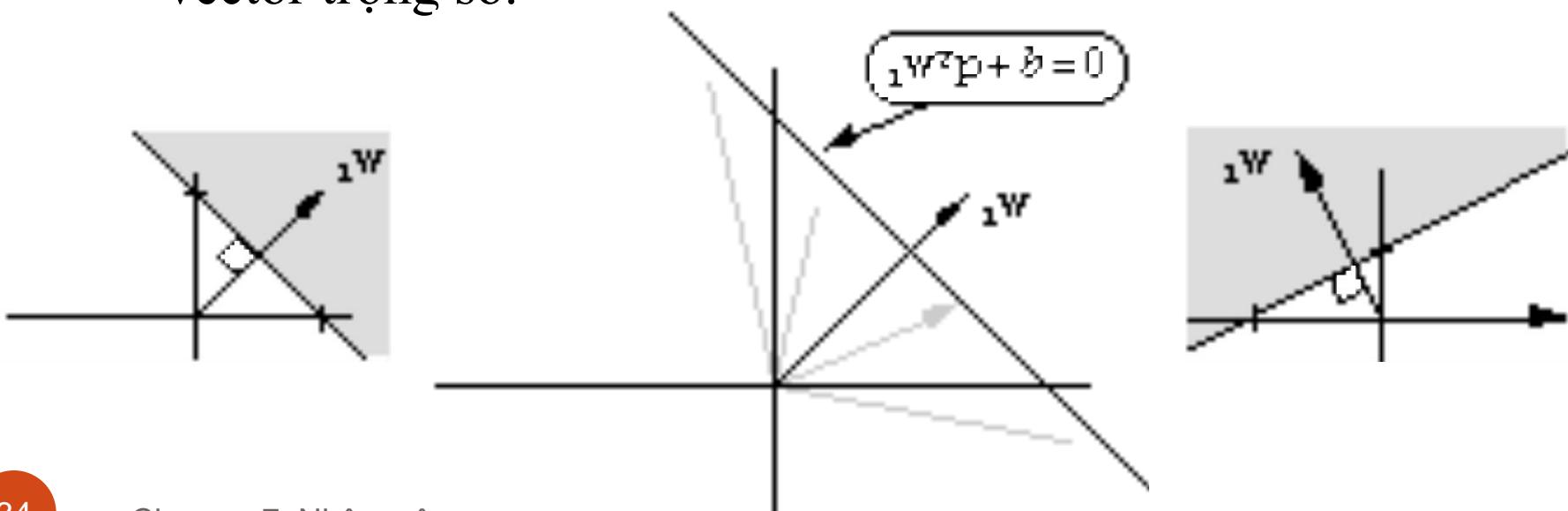
$$a = \text{hardlim}(w^T p + b) = \text{hardlim}(w_{1,1}p_1 + w_{1,2}p_2 + b)$$

Biên quyết định

$$_1\mathbf{w}^T \mathbf{p} + b = 0$$

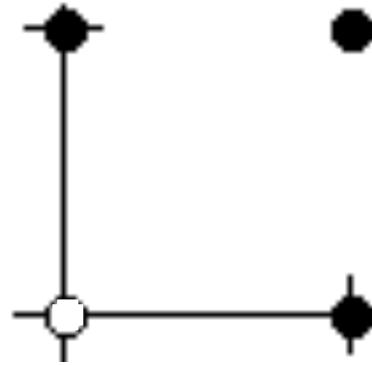
$$_1\mathbf{w}^T \mathbf{p} = -b$$

- Tất cả các điểm trên biên quyết định có cùng tích vô hướng với vector trọng số.
- Tất cả khi chiếu lên vector trọng số đều quy về một điểm. Nói khác đi chúng nằm trên đường thẳng vuông góc với vector trọng số.

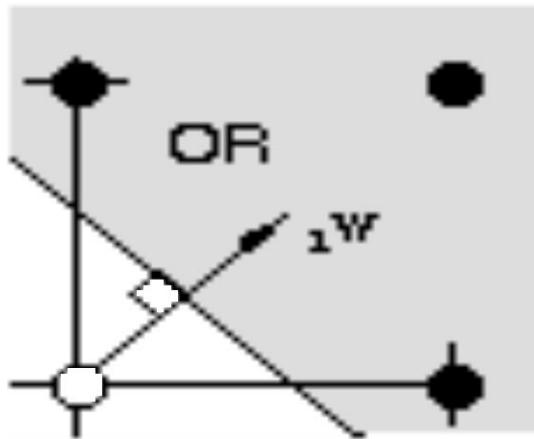


Ví dụ hàm - OR

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, t_1 = 0 \right\} \quad \left\{ \mathbf{p}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, t_2 = 1 \right\} \quad \left\{ \mathbf{p}_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, t_3 = 1 \right\} \quad \left\{ \mathbf{p}_4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_4 = 1 \right\}$$



Lời giải cho bài toán phân lớp OR



Siêu phẳng biên phải vuông góc với vector trọng số.

$$_1\mathbf{w} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

Lấy một điểm bất kỳ trên siêu phẳng biên để tính giá trị của bias.

$$_1\mathbf{w}^T \mathbf{p} + b = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} + b = 0.25 + b = 0 \quad \Rightarrow \quad b = -0.25$$

Perceptron đơn lớp nhiều đầu ra

Mỗi một neuron có một siêu phẳng biên riêng.

$$_i \mathbf{w}^T \mathbf{p} + b_i = 0$$

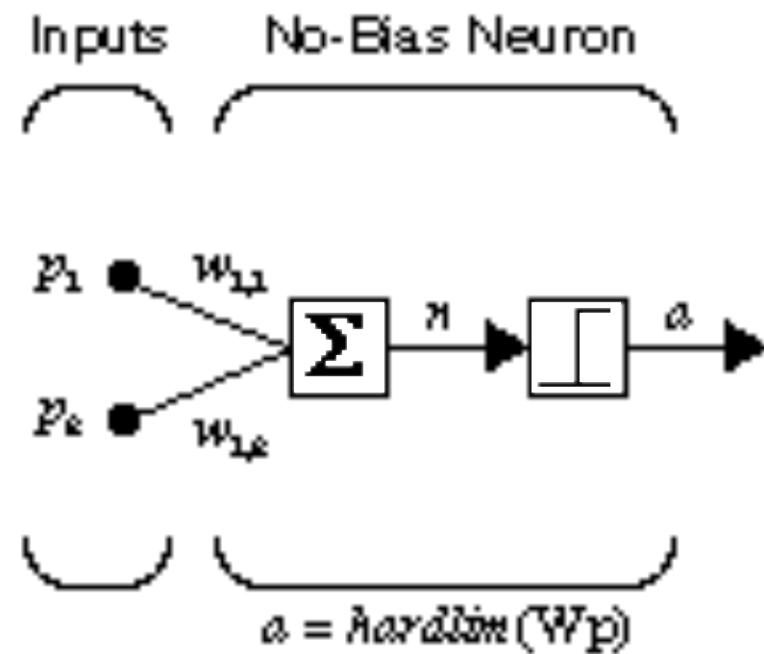
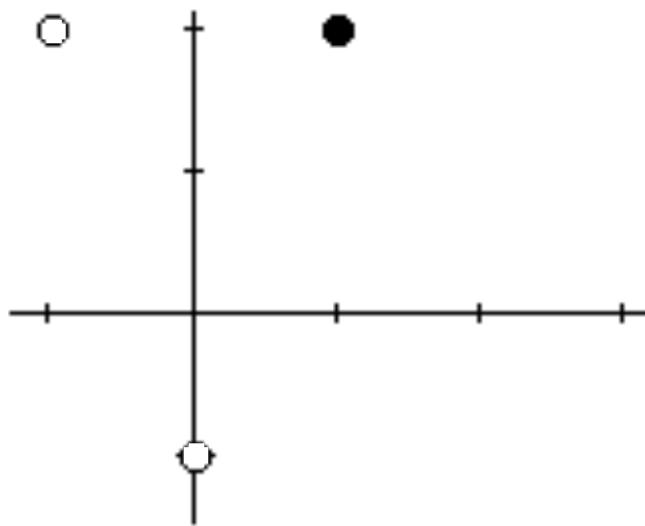
Mỗi neuron có thể dùng để phân tách hai lớp.

Do đó nếu n-neuron đầu ra thì có thể dùng để phân tách 2^n lớp.

Luật học qua ví dụ

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \dots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

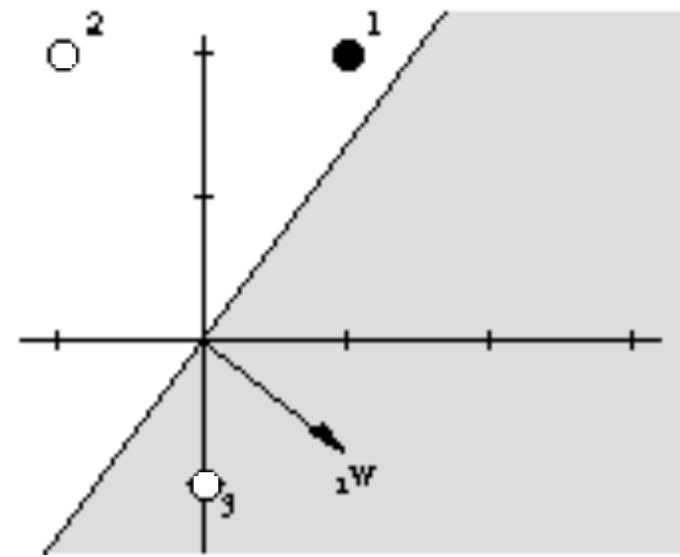
$$\left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, t_1 = 1 \right\} \quad \left\{ \mathbf{p}_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, t_2 = 0 \right\} \quad \left\{ \mathbf{p}_3 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, t_3 = 0 \right\}$$



Khởi tạo ban đầu

Khởi tạo ngẫu nhiên:

$$_1\mathbf{w} = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix}$$



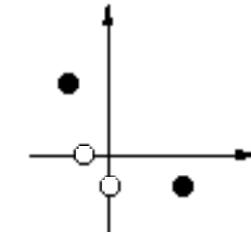
nạp \mathbf{p}_1 vào mạng neural:

$$a = \text{hardlim}(_1\mathbf{w}^T \mathbf{p}_1) = \text{hardlim}\left(\begin{bmatrix} 1.0 & -0.8 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right)$$

$$a = \text{hardlim}(-0.6) = 0$$

LUẬT HỌC

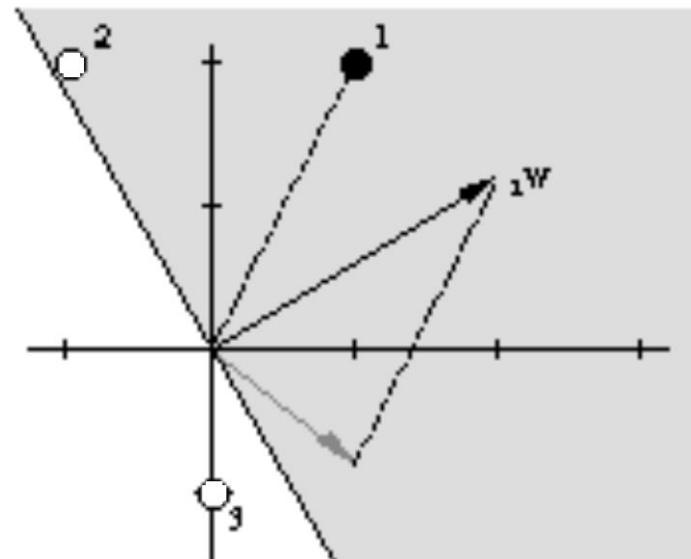
- đặt ${}_1\mathbf{w}$ to \mathbf{p}_1
 - Không ổn định \times



- cộng \mathbf{p}_1 vào ${}_1\mathbf{w}$ \checkmark

Ta có luật: If $t = 1$ and $a = 0$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}_1 = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix}$$



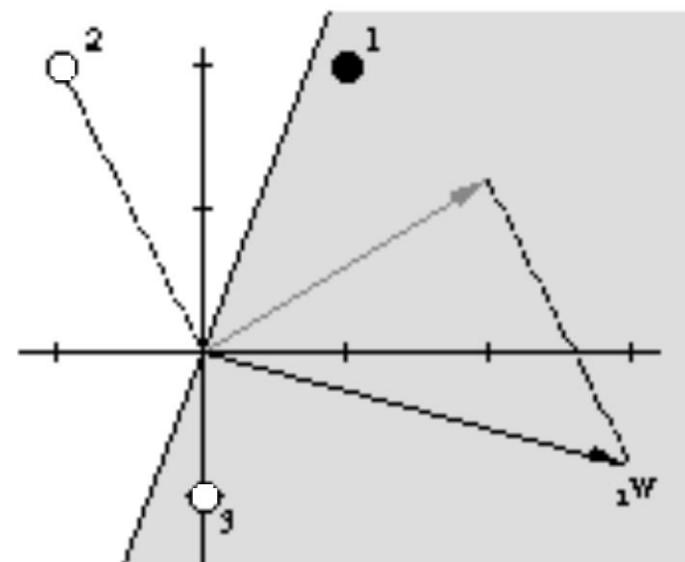
Vector mẫu thứ hai

$$a = \text{hardlim}({}_1\mathbf{w}^T \mathbf{p}_2) = \text{hardlim}\left(\begin{bmatrix} 2.0 & 1.2 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \end{bmatrix}\right)$$

$$a = \text{hardlim}(0.4) = 1 \quad (\text{Phân lớp sai})$$

Ta có luật học: If $t = 0$ and $a = 1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_2 = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} - \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix}$$

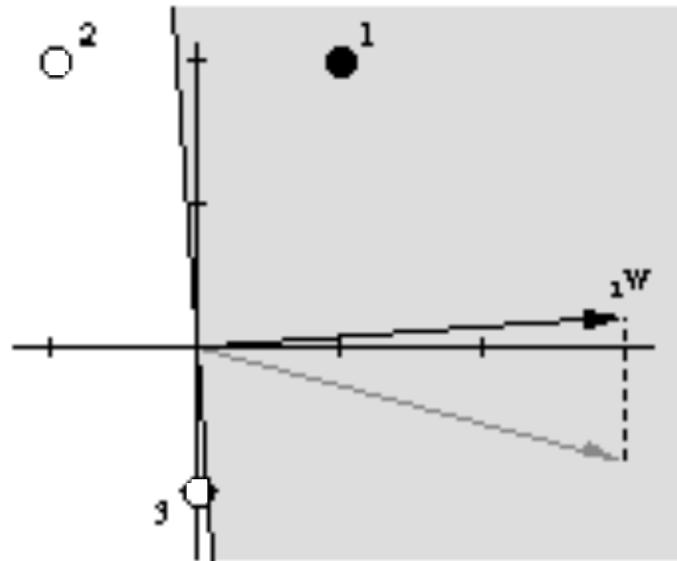


Vector mẫu thử ba

$$a = \text{hardlim}({}_1\mathbf{w}^T \mathbf{p}_3) = \text{hardlim}\left(\begin{bmatrix} 3.0 & -0.8 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \end{bmatrix}\right)$$

$$a = \text{hardlim}(0.8) = 1 \quad (\text{Phân lớp sai})$$

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_3 = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 3.0 \\ 0.2 \end{bmatrix}$$



Siêu phẳng đã phân lớp chính xác.

If $t = a$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$.

Luật học thống nhất

If $t = 1$ and $a = 0$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$

If $t = 0$ and $a = 1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$

If $t = a$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$

$$e = t - a$$

If $e = 1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$

If $e = -1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$

If $e = 0$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + e\mathbf{p} = {}_1\mathbf{w}^{old} + (t-a)\mathbf{p}$$

$$b^{new} = b^{old} + e$$

chú ý: bias
tương đương
với đầu vào có
kết nối
trọng số = 1.

Perceptron đơn lớp nhiều đầu ra

Update dòng thứ i của ma trận trọng số:

$$_i\mathbf{w}^{new} = _i\mathbf{w}^{old} + e_i \mathbf{p}$$

$$b_i^{new} = b_i^{old} + e_i$$

Dạng ma trận:

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{e} \mathbf{p}^T$$

$$\mathbf{b}^{new} = \mathbf{b}^{old} + \mathbf{e}$$

Ví dụ tự động phân loại Táo/Chuối

Tập huấn luyện (ba thuộc tính: Shape, Texture, Weight)

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}, t_1 = \boxed{1} \right\} \quad \left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, t_2 = \boxed{0} \right\}$$

Trọng số khởi tạo

$$\mathbf{W} = \begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} \quad b = 0.5$$

Lần lặp đầu tiên

$$a = \text{hardlim}(\mathbf{W}\mathbf{p}_1 + b) = \text{hardlim}\left(\begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} + 0.5\right)$$

$$a = \text{hardlim}(-0.5) = 0 \quad e = t_1 - a = 1 - 0 = 1$$

$$\mathbf{W}^{new} = \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} + (1) \begin{bmatrix} -1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} -0.5 & 0 & -1.5 \end{bmatrix}$$

Chương 7: Nhập môn ~~bày học~~ $b^{new} = b^{old} + e = 0.5 + (1) = 1.5$

Lần lặp thứ hai

$$a = \text{hardlim}(\mathbf{W}\mathbf{p}_2 + b) = \text{hardlim}(\begin{bmatrix} -0.5 & 0 & -1.5 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + (1.5))$$
$$a = \text{hardlim}(2.5) = 1$$

$$e = t_2 - a = 0 - 1 = -1$$

$$\mathbf{W}^{new} = \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix} -0.5 & 0 & -1.5 \end{bmatrix} + (-1) \begin{bmatrix} 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} -1.5 & -1 & -0.5 \end{bmatrix}$$

$$b^{new} = b^{old} + e = 1.5 + (-1) = 0.5$$

Kiểm tra

$$a = \text{hardlim}(\mathbf{W}\mathbf{p}_1 + b) = \text{hardlim}(\begin{bmatrix} -1.5 & -1 & -0.5 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} + 0.5)$$

$$a = \text{hardlim}(1.5) = 1 = t_1$$

$$a = \text{hardlim}(\mathbf{W}\mathbf{p}_2 + b) = \text{hardlim}(\begin{bmatrix} -1.5 & -1 & -0.5 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + 0.5)$$

$$a = \text{hardlim}(-1.5) = 0 = t_2$$

Định Lý hội tụ cho Perceptron

1. Khả tách tuyến tính: Hai tập điểm A và B trong không gian vector X n-chiều được coi là khả tách tuyến tính nếu tồn tại $n+1$ số thực w_1, \dots, w_n sao cho với mọi $\mathbf{x}=(x_1, x_2, \dots, x_n) \in A$ thoả mãn $\mathbf{wx} \geq w_{n+1}$ và với mọi $\mathbf{y}=(y_1, y_2, \dots, y_n) \in B$ thoả mãn $\mathbf{wy} < 0$.
2. Khả tách tuyến tính mạnh: Hai tập điểm A và B trong không gian vector X n-chiều được coi là khả tách tuyến tính nếu tồn tại $n+1$ số thực w_1, \dots, w_n sao cho với mọi $\mathbf{x}=(x_1, x_2, \dots, x_n) \in A$ thoả mãn $\mathbf{wx} > w_{n+1}$ và với mọi $\mathbf{y}=(y_1, y_2, \dots, y_n) \in B$ thoả mãn $\mathbf{wy} < 0$.

Bổ đề: Khả tách tuyến tính \Rightarrow khả tách tuyến tính tuyệt đối.

Định Lý hội tụ cho Perceptron

Định lý (cho perceptron đơn lớp một đầu ra): Nếu hai tập P và N là hữu hạn và khả tách tuyến tính thì luật toán học Perceptron sẽ hội tụ (có nghĩa là w sẽ chỉ được update một số hữu hạn lần).

Chứng minh:

- i) Đặt $P' = P \cup N'$ trong đó N' gồm những phần tử không thuộc N.
- ii) Các vector thuộc P' có thể chuẩn hoá (chuan bằng 1) vì nếu tồn tại w sao cho $wx > 0$ (với mọi $x \in P'$) thì $\eta x > 0$ với mọi η
- iii) Vector có thể chuẩn hoá. Do giả thiết là siêu phẳng biên tồn tại, nên phải có vector lời giải được chuẩn hoá w^* .

Định Lý hội tụ cho Perceptron

Chứng minh (tiếp): Giả sử thuật toán đã chạy được $t+1$ bước, $\mathbf{w}(t+1)$ đã được update. Có nghĩa là tại bước t tồn tại vector \mathbf{p}_i bị phân lớp sai (có thể lập luận tương tự trong trường hợp vector bị phân lớp sai là n_i).

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \mathbf{p}_i \quad (1)$$

Cosine của góc ρ giữa \mathbf{w} và \mathbf{w}^* là:

$$\cos \rho = (\mathbf{w}^* \cdot \mathbf{w}(t+1)) / (\|\mathbf{w}^*\| \cdot \|\mathbf{w}(t+1)\|) = (\mathbf{w}^* \cdot \mathbf{w}(t+1)) / (\|\mathbf{w}(t+1)\|) \quad (2)$$

Thế (1) vào tử số của (2) ta có:

$$\begin{aligned} \mathbf{w}^* \cdot \mathbf{w}(t+1) &= \mathbf{w}^* \cdot (\mathbf{w}(t) + \mathbf{p}_i) = \mathbf{w}^* \cdot \mathbf{w}(t) + \mathbf{w}^* \cdot \mathbf{p}_i \geq \mathbf{w}^* \cdot \mathbf{w}_t + \delta \text{ trong đó} \\ \delta &= \min\{\mathbf{w}^* \cdot \mathbf{p} / \forall \mathbf{p} \in P'\}. \end{aligned}$$

Do đó bằng quy nạp ta có: $\mathbf{w}^* \cdot \mathbf{w}(t+1) \geq \mathbf{w}^* \cdot \mathbf{w}(0) + (t+1) \delta \quad (3)$

Mặt khác xét mẫu số của (2):

$$\|\mathbf{w}(t+1)\|^2 = (\mathbf{w}(t) + \mathbf{p}_i) \cdot (\mathbf{w}(t) + \mathbf{p}_i) = \|\mathbf{w}(t)\|^2 + 2\mathbf{w}(t) \cdot \mathbf{p}_i + \|\mathbf{p}_i\|^2$$

Do $\mathbf{w}(t) \cdot \mathbf{p}_i$ là số âm hoặc bằng 0 (do đó mới phải update $\mathbf{w}(t)$)

$$\Rightarrow \|\mathbf{w}(t+1)\|^2 \leq \|\mathbf{w}(t)\|^2 + \|\mathbf{p}_i\|^2 \leq \|\mathbf{w}(t)\|^2 + 1 \leq \|\mathbf{w}(0)\|^2 + (t+1) \quad (4)$$

Kết hợp (3) và (4) ta có:

$$\cos \rho \geq (\mathbf{w}^* \cdot \mathbf{w}(0) + (t+1)\delta) / \sqrt{\|\mathbf{w}(0)\|^2 + (t+1)}$$

Về phải tăng tỷ lệ với \sqrt{t} (do $\delta > 0$) do đó t bắt buộc phải hữu hạn (vì $\cos \rho \leq 1$). Kết thúc chứng minh.

Cải tiến thuật toán học Perceptron

- Mặc dù định lý hội tụ đảm bảo tính dừng của thuật toán học Perceptron, tuy nhiên trong thực tế số lần lặp có thể rất lớn (thậm chí là hàm số mũ với đầu vào).
- Corrective Learning (perceptron một đầu ra):

$$\delta = \mathbf{e} \cdot \mathbf{w}(t) \cdot \mathbf{p}_i \quad (\mathbf{e} = \mathbf{y}_i - \mathbf{t}_i)$$

$$\mathbf{w}(t+1) = \mathbf{w}(t) + ((\delta + \varepsilon) \cdot \mathbf{p}_i) / \|\mathbf{p}_i\|$$

Mỗi sample bị phân lớp sai có thể hiệu chỉnh lại trong một bước để phân lớp đúng. Ví dụ nếu $\mathbf{p}_i \in P$ bị phân lớp sai ($\mathbf{w}(t) \cdot \mathbf{p}_i < 0$).

$$\mathbf{w}(t+1) \cdot \mathbf{x} = (\mathbf{w}(t) + ((\delta + \varepsilon) \cdot \mathbf{p}_i) / \|\mathbf{p}_i\|) \cdot \mathbf{x} = \mathbf{w}(t) \cdot \mathbf{x} + \delta + \varepsilon = -\delta + \delta + \varepsilon = \varepsilon > 0.$$

Cải tiến thuật toán học Perceptron

2. Thuật giải học Pocket (Gallant, 1990):

Trong thực tế tập dữ liệu không phải lúc nào cũng khả tách tuyến tính một cách hoàn hảo. Do đó cần lưu giữ vector trọng số (hay ma trận trọng số) đạt mức phân lớp tốt nhất.

Giải thuật:(perceptron một đầu ra)

Bước khởi tạo: Khởi trị vector w ngẫu nhiên. Đặt $w_s = w$ (w_s là vector lưu trữ), $h_s = 0$ (h_s là số vector phân lớp thành công liên tiếp).

Bước lặp: Update w sử dụng luật học Perceptron. Dùng biến h ghi lại số lần phân lớp thành công liên tiếp. Nếu $h > h_s$ thì thay w_s bằng w và h_s thay bằng h . Tiếp tục lặp.

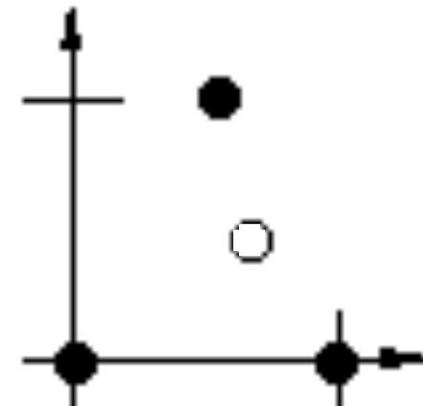
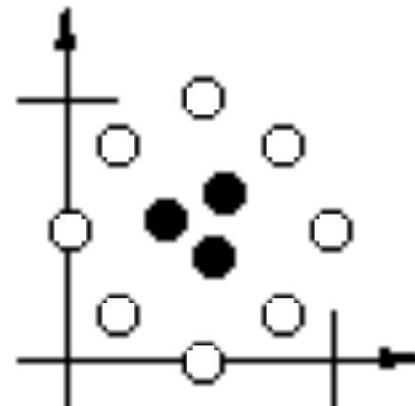
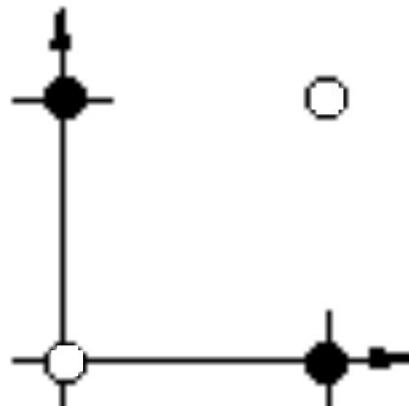
(Gallant, 1990) chứng minh rằng nếu tập training là hữu hạn và các vector đầu vào cũng như trọng số là hữu tỷ thì thuật toán sẽ hội tụ tới lời giải tối ưu với xác suất bằng 1.

Hạn chế của Perceptron đơn lớp

Biên quyết định là tuyến tính

$$_1\mathbf{w}^T \mathbf{p} + b = 0$$

Không dùng được đối với các bài toán không khả tách tuyến tính



Đọc thêm

1. M.T. Hagan, H.B. Demuth, and M. Beale, *Neural Network Design*, PWS Publishing.
2. N.J. Nilson, *Mathematical Foundations of Learning Machines*, Morgan Kaufmann.

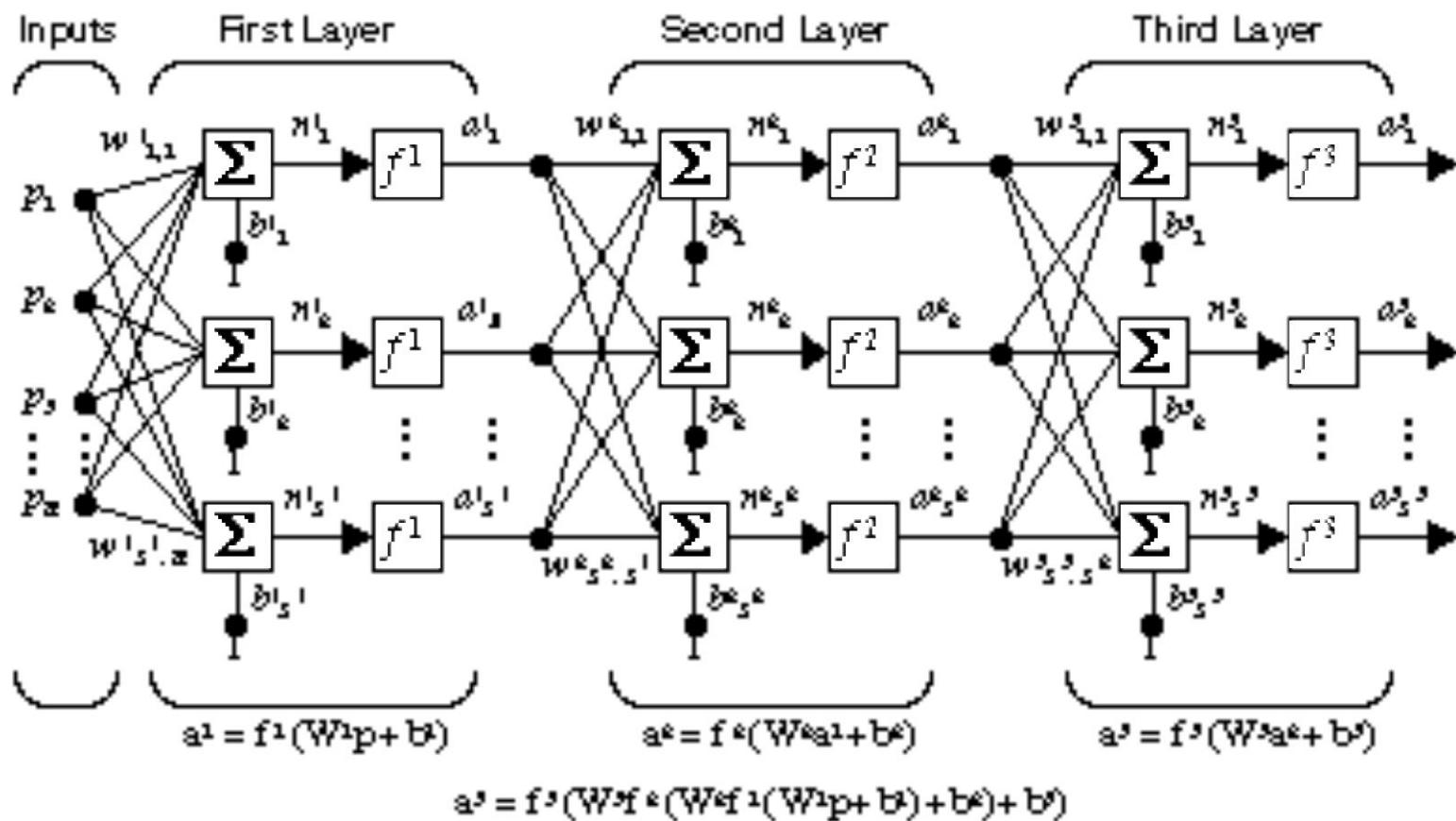
Câu Hỏi Ôn Tập

1. Nêu cấu trúc và luật học của mạng Perceptron?
2. Chứng minh định lý hội tụ của mạng Perceptron?
3. Lấy các ví dụ và mô phỏng Perceptron trong MatLab?

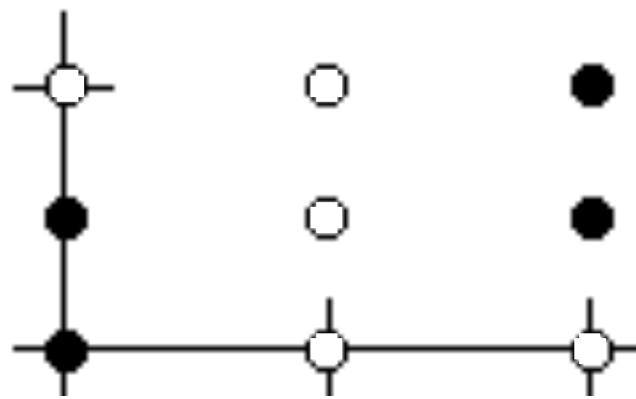
PERCEPTRON ĐA LỚP

- **Nội dung:**
 - Cấu trúc mạng Perceptron đa lớp.
 - Thuật giải lan truyền ngược (BP).
 - Định Lý Kolmogorov và độ phức tạp học.
 - Một số Heuristics cho việc cải tiến BP.
 -

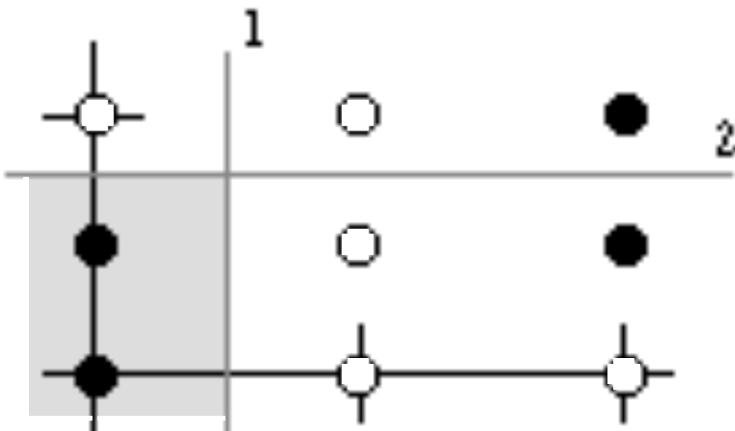
Perceptron Đa Lớp



Ví dụ



Các biên quyết định



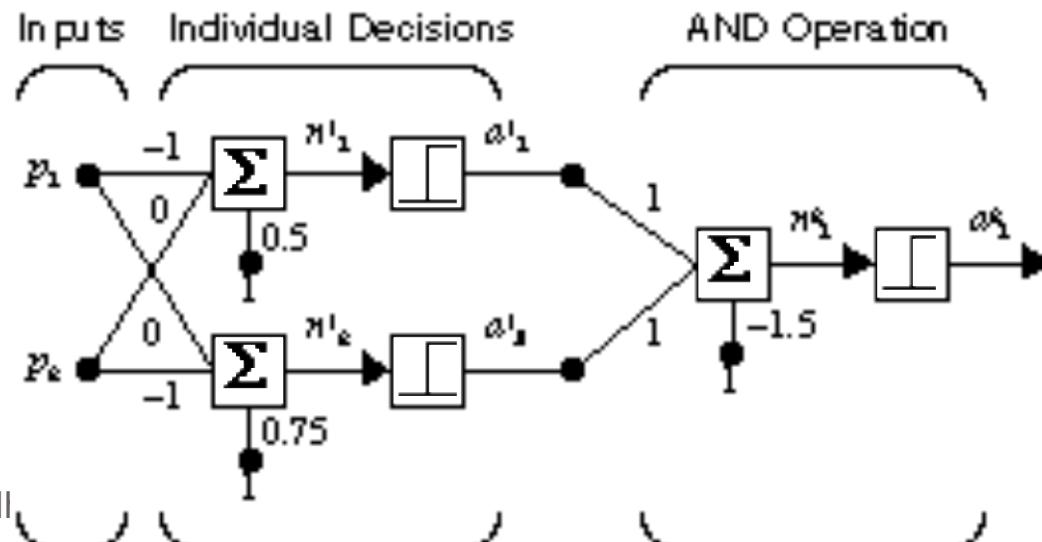
Biên thứ nhất:

$$a_1^1 = \text{hardlim}(\begin{bmatrix} -1 & 0 \end{bmatrix} \mathbf{p} + 0.5)$$

Biên thứ hai:

$$a_2^1 = \text{hardlim}(\begin{bmatrix} 0 & -1 \end{bmatrix} \mathbf{p} + 0.75)$$

Mạng con thứ nhất



Các Biên Quyết Định



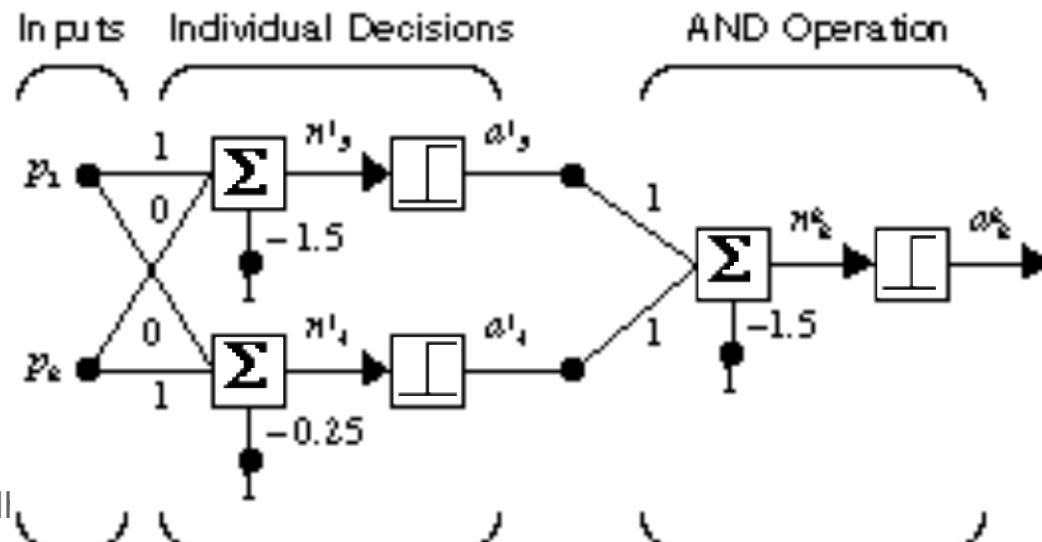
Biên thứ ba:

$$a_3^1 = \text{hardlim}(\begin{bmatrix} 1 & 0 \end{bmatrix} p - 1.5)$$

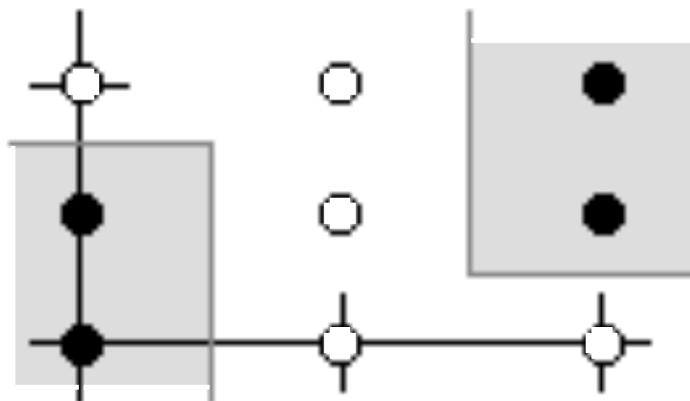
Biên thứ tư:

$$a_4^1 = \text{hardlim}(\begin{bmatrix} 0 & 1 \end{bmatrix} p - 0.25)$$

Mạng con thứ hai



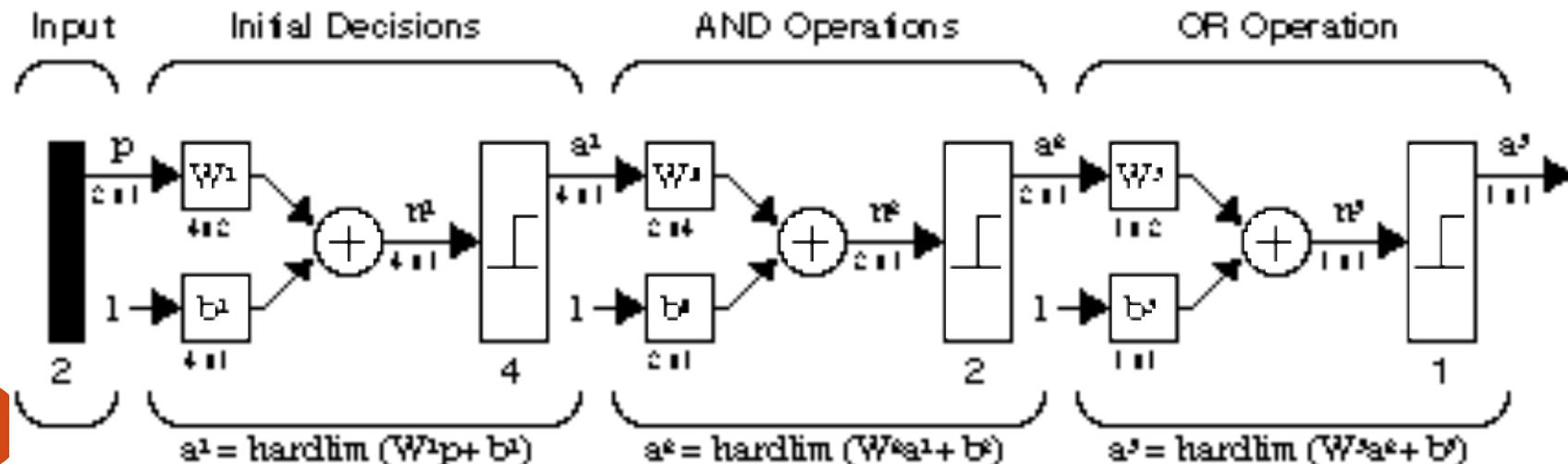
Mạng tổng hợp



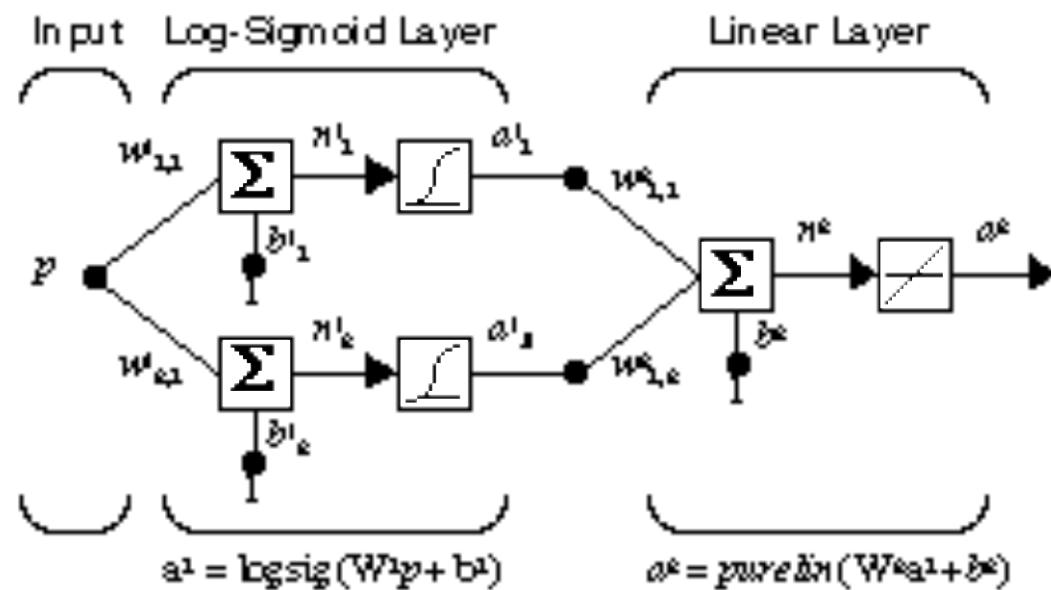
$$\mathbf{W}^1 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \mathbf{b}^1 = \begin{bmatrix} 0.5 \\ 0.75 \\ -1.5 \\ -0.25 \end{bmatrix}$$

$$\mathbf{W}^2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad \mathbf{b}^2 = \begin{bmatrix} -1.5 \\ -1.5 \end{bmatrix}$$

$$\mathbf{W}^3 = \begin{bmatrix} 1 & 1 \end{bmatrix} \quad \mathbf{b}^3 = \begin{bmatrix} -0.5 \end{bmatrix}$$



Xấp Xỉ Hàm



$$f^1(n) = \frac{1}{1 + e^{-n}}$$

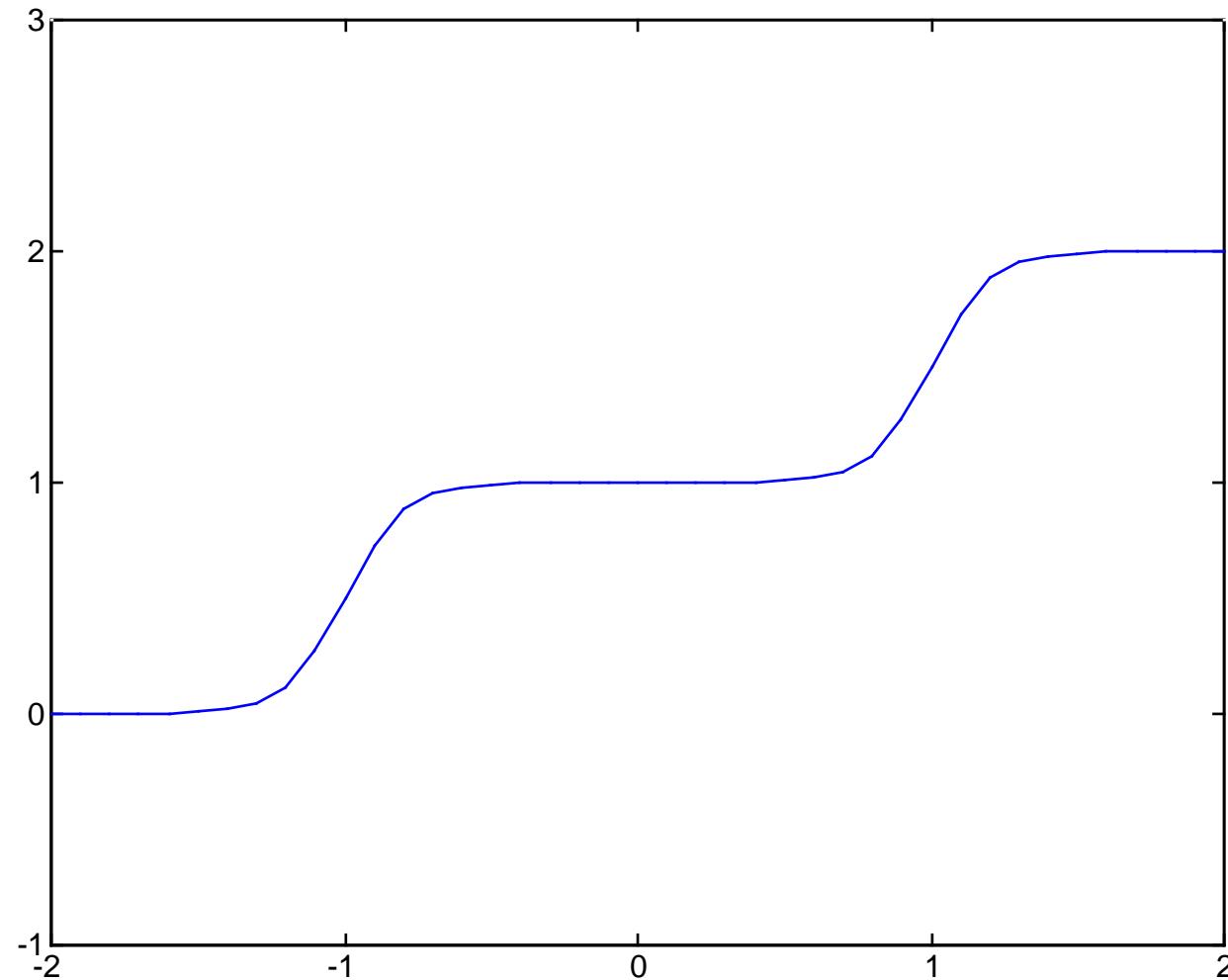
$$f^2(n) = n$$

Giá trị các tham số

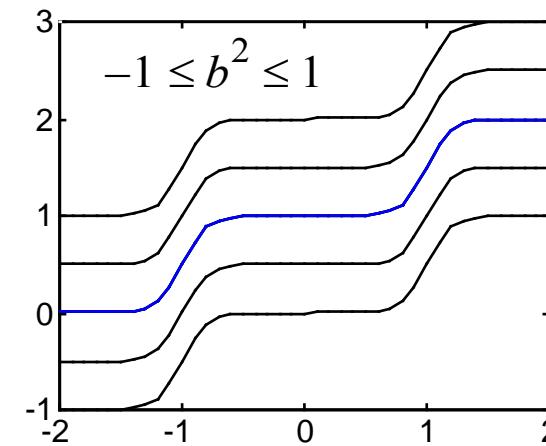
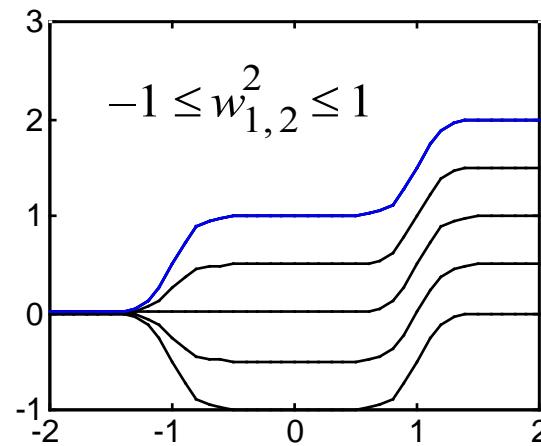
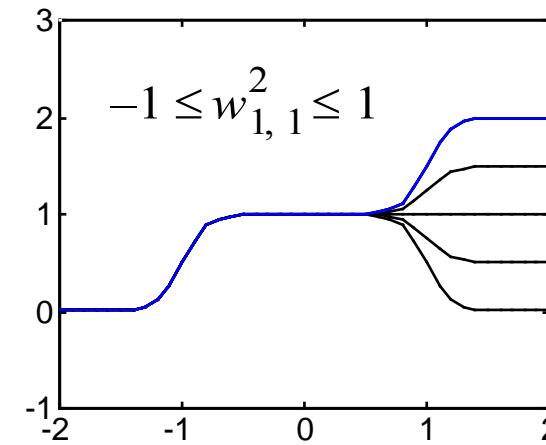
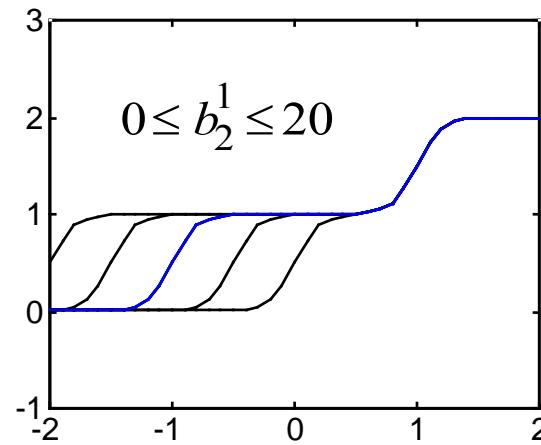
$$w_{1,1}^1 = 10 \quad w_{2,1}^1 = 10 \quad b_1^1 = -10 \quad b_2^1 = 10$$

$$w_{1,1}^2 = 1 \quad w_{1,2}^2 = 1 \quad b^2 = 0$$

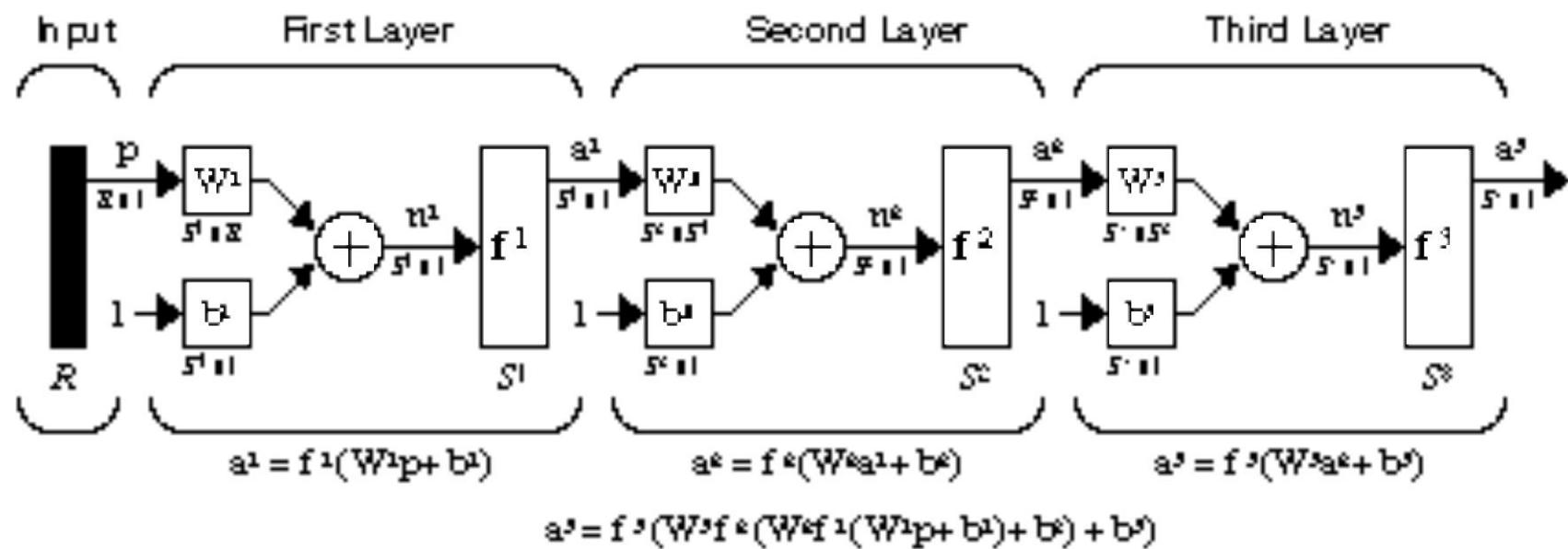
Đồ thị "hàm" của mạng



Thay đổi giá trị các tham số



Mạng Đa Lớp



$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1} \mathbf{a}^m + \mathbf{b}^{m+1}) \quad m = 0, 2, \dots, M-1$$

$$\mathbf{a}^0 = \mathbf{p}$$

$$\mathbf{a} = \mathbf{a}^M$$

Hàm Lỗi

Tập huấn luyện

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \dots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

MSE

$$F(\mathbf{x}) = E[e^2] = E[(t - a)^2]$$

Dạng Vector

$$F(\mathbf{x}) = E[\mathbf{e}^T \mathbf{e}] = E[(\mathbf{t} - \mathbf{a})^T (\mathbf{t} - \mathbf{a})]$$

Xấp xỉ MSE

$$\hat{F}(\mathbf{x}) = (\mathbf{t}(k) - \mathbf{a}(k))^T (\mathbf{t}(k) - \mathbf{a}(k)) = \mathbf{e}^T(k) \mathbf{e}(k)$$

Approximate Steepest Descent

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha \frac{\partial \hat{F}}{\partial w_{i,j}^m} \quad b_i^m(k+1) = b_i^m(k) - \alpha \frac{\partial \hat{F}}{\partial b_i^m}$$

Đạo Hàm Hàm Hợp

$$\frac{df(n(w))}{dw} = \frac{df(n)}{dn} \times \frac{dn(w)}{dw}$$

Ví dụ

$$f(n) = \cos(n) \quad n = e^{2w} \quad f(n(w)) = \cos(e^{2w})$$

$$\frac{df(n(w))}{dw} = \frac{df(n)}{dn} \times \frac{dn(w)}{dw} = (-\sin(n))(2e^{2w}) = (-\sin(e^{2w}))(2e^{2w})$$

Tính Gradient

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial w_{i,j}^m}$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial b_i^m}$$

TÍNH GRADIENT

$$n_i^m = \sum_{j=1}^{S^{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m$$

$$\frac{\partial n_i^m}{\partial w_{i,j}^m} = a_j^{m-1} \quad \frac{\partial n_i^m}{\partial b_i^m} = 1$$

Độ nhạy

$$s_i^m \equiv \frac{\partial \hat{F}}{\partial n_i^m}$$

Gradient

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = s_i^m a_j^{m-1} \quad \frac{\partial \hat{F}}{\partial b_i^m} = s_i^m$$

Steepest Descent

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha s_i^m a_j^{m-1} \quad b_i^m(k+1) = b_i^m(k) - \alpha s_i^m$$

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T \quad \mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m$$

$$\mathbf{s}^m \equiv \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \begin{bmatrix} \frac{\partial \hat{F}}{\partial n_1^m} \\ \frac{\partial \hat{F}}{\partial n_2^m} \\ \vdots \\ \frac{\partial \hat{F}}{\partial n_{S^m}^m} \end{bmatrix}$$

Ma trận Jacobian

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} = \begin{bmatrix} \frac{\partial n_1^{m+1}}{\partial n_1^m} & \frac{\partial n_1^{m+1}}{\partial n_2^m} & \dots & \frac{\partial n_1^{m+1}}{\partial n_{S^m}^m} \\ \frac{\partial n_2^{m+1}}{\partial n_1^m} & \frac{\partial n_2^{m+1}}{\partial n_2^m} & \dots & \frac{\partial n_2^{m+1}}{\partial n_{S^m}^m} \\ \vdots & \vdots & & \vdots \\ \frac{\partial n_{S^m+1}^{m+1}}{\partial n_1^m} & \frac{\partial n_{S^m+1}^{m+1}}{\partial n_2^m} & \dots & \frac{\partial n_{S^m+1}^{m+1}}{\partial n_{S^m}^m} \end{bmatrix}$$

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} = \mathbf{W}^{m+1} \mathbf{F}'^m(\mathbf{n}^m)$$

$$\frac{\partial n_i^{m+1}}{\partial n_j^m} = \frac{\partial}{\partial n_j^m} \left(\sum_{l=1}^{S^m} w_{i,l}^{m+1} a_l^m + b_i^{m+1} \right) = w_{i,j}^{m+1} \frac{\partial a_j^m}{\partial n_j^m}$$

$$\frac{\partial n_i^{m+1}}{\partial n_j^m} = w_{i,j}^{m+1} \frac{\partial f^m(n_j^m)}{\partial n_j^m} = w_{i,j}^{m+1} f'^m(n_j^m)$$

$$f'^m(n_j^m) = \frac{\partial f^m(n_j^m)}{\partial n_j^m}$$

$$\mathbf{F}'^m(\mathbf{n}^m) = \begin{bmatrix} f'^m(n_1^m) & 0 & \dots & 0 \\ 0 & f'^m(n_2^m) & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & f'^m(n_{S^m}^m) \end{bmatrix}$$

Backpropagation (Sensitivities)

$$\mathbf{s}^m = \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \left(\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} \right)^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}} = \mathbf{F}'^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}}$$

$$\mathbf{s}^m = \mathbf{F}'^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1}$$

Độ nhạy được tính từ tầng cuối cùng và lan truyền ngược lại cho đến tầng đầu.

$$\mathbf{s}^M \rightarrow \mathbf{s}^{M-1} \rightarrow \dots \rightarrow \mathbf{s}^2 \rightarrow \mathbf{s}^1$$

Khởi đầu (Last Layer)

$$s_i^M = \frac{\partial \hat{F}}{\partial n_i^M} = \frac{\partial (\mathbf{t} - \mathbf{a})^T (\mathbf{t} - \mathbf{a})}{\partial n_i^M} = \frac{\partial \sum_{j=1}^{S^M} (t_j - a_j)^2}{\partial n_i^M} = -2(t_i - a_i) \frac{\partial a_i}{\partial n_i^M}$$

$$\frac{\partial a_i}{\partial n_i^M} = \frac{\partial a_i^M}{\partial n_i^M} = \frac{\partial f^M(n_i^M)}{\partial n_i^M} = f^M(n_i^M)$$

$$s_i^M = -2(t_i - a_i) f^M(n_i^M)$$

$$\mathbf{s}^M = -2 \mathbf{f}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a})$$

Tổng kết thuật toán

Lan truyền Xuôi

$$\mathbf{a}^0 = \mathbf{p}$$

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^m + \mathbf{b}^{m+1}) \quad m = 0, 2, \dots, M-1$$

$$\mathbf{a} = \mathbf{a}^M$$

Lan truyền ngược

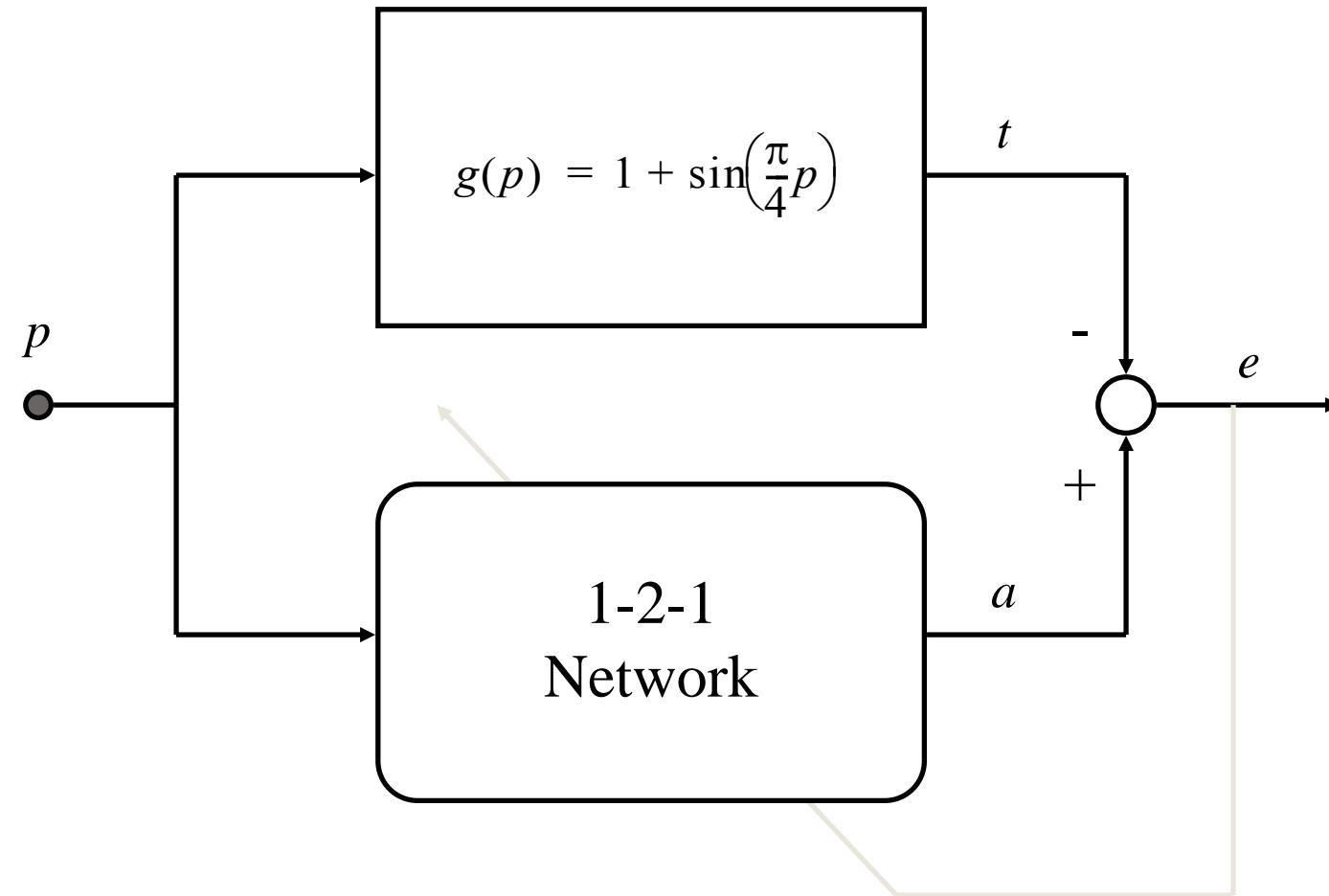
$$\mathbf{s}^M = -2\tilde{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a})$$

$$\mathbf{s}^m = \tilde{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1} \quad m = M-1, \dots, 2, 1$$

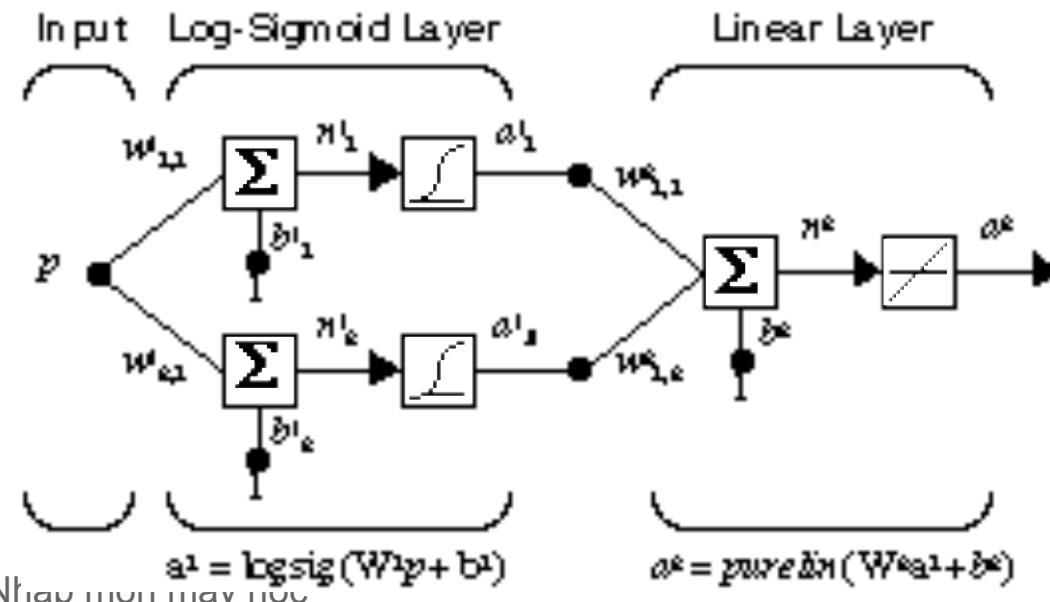
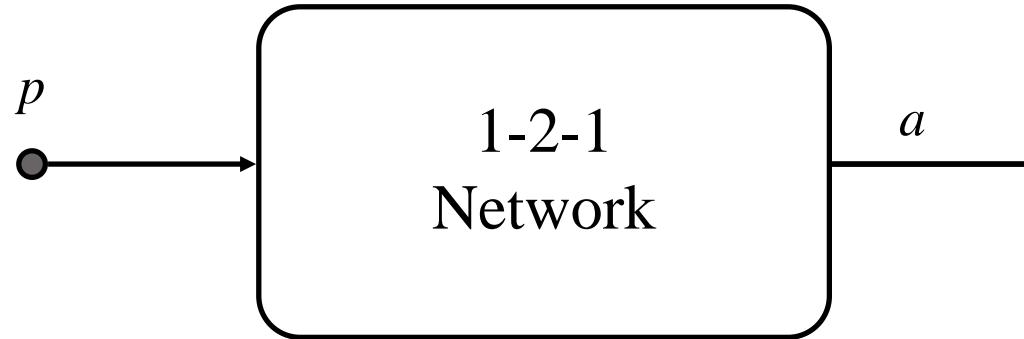
Cập nhật trọng số

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T \quad \mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m$$

Ví dụ: Regression

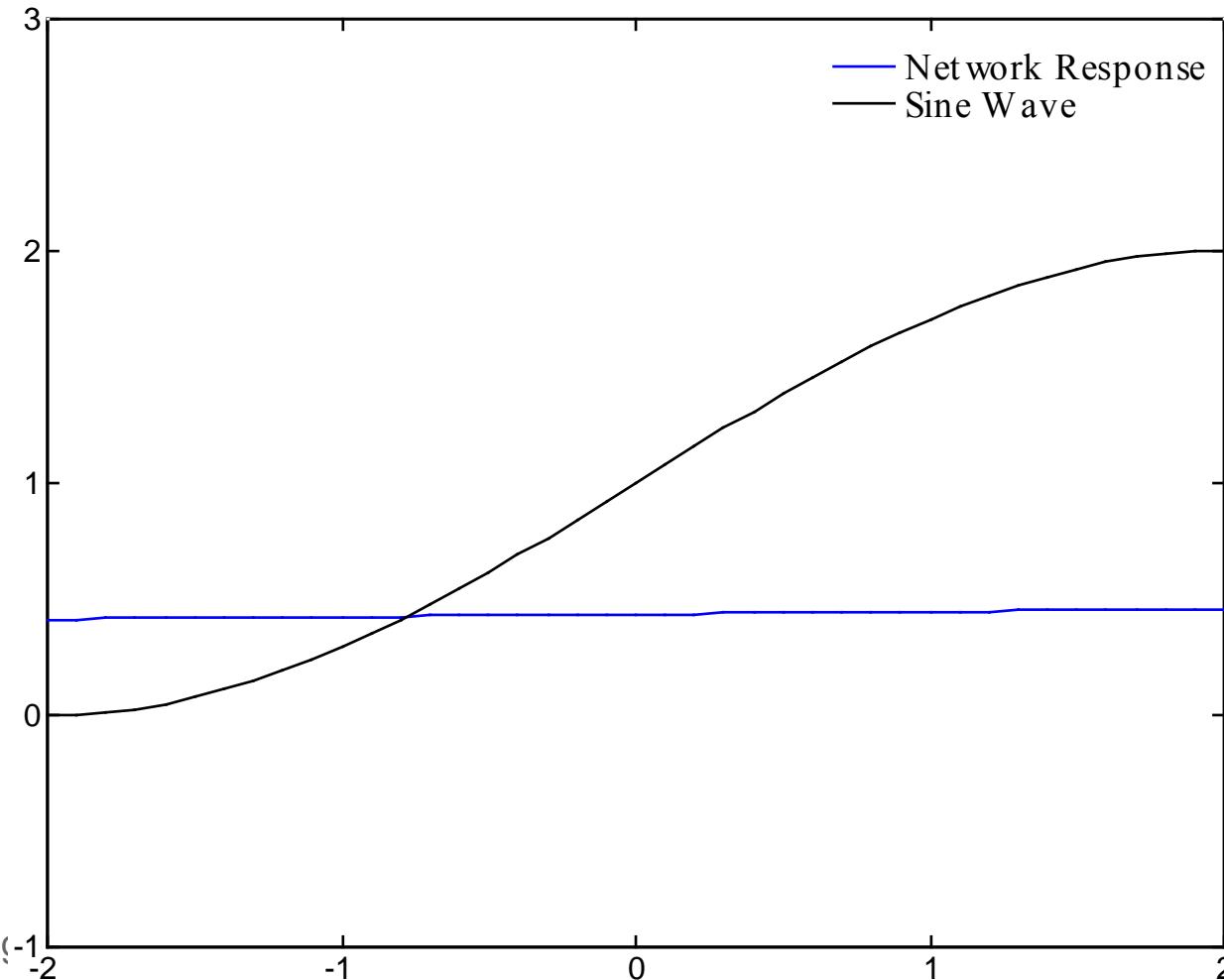


Network



Khởi tạo ban đầu

$$\mathbf{W}^1(0) = \begin{bmatrix} -0.27 \\ -0.41 \end{bmatrix} \quad \mathbf{b}^1(0) = \begin{bmatrix} -0.48 \\ -0.13 \end{bmatrix} \quad \mathbf{W}^2(0) = \begin{bmatrix} 0.09 & -0.1 \end{bmatrix} \quad \mathbf{b}^2(0) = \begin{bmatrix} 0.48 \end{bmatrix}$$



Lan Truyền Xuôi

$$a^0 = p = 1$$

$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{W}^1 \mathbf{a}^0 + \mathbf{b}^1) = \text{logsig}\left(\begin{bmatrix} -0.27 \\ -0.41 \end{bmatrix} + \begin{bmatrix} -0.48 \\ -0.13 \end{bmatrix}\right) = \text{logsig}\left(\begin{bmatrix} -0.75 \\ -0.54 \end{bmatrix}\right)$$

$$\mathbf{a}^1 = \begin{bmatrix} \frac{1}{1+e^{0.75}} \\ \frac{1}{1+e^{0.54}} \end{bmatrix} = \begin{bmatrix} 0.32 \\ 0.368 \end{bmatrix}$$

$$a^2 = \mathbf{f}^2(\mathbf{W}^2 \mathbf{a}^1 + \mathbf{b}^2) = \text{purelin} (\begin{bmatrix} 0.09 & -0.17 \\ 0.368 \end{bmatrix} \begin{bmatrix} 0.32 \\ 1 \end{bmatrix} + \begin{bmatrix} 0.48 \end{bmatrix}) = \begin{bmatrix} 0.446 \end{bmatrix}$$

$$e = t - a = \left\{ 1 + \sin\left(\frac{\pi}{4}p\right) \right\} - a^2 = \left\{ 1 + \sin\left(\frac{\pi}{4}1\right) \right\} - 0.446 = 1.261$$

Đạo hàm hàm chuyển

$$f^1(n) = \frac{d}{dn} \left(\frac{1}{1 + e^{-n}} \right) = \frac{e^{-n}}{(1 + e^{-n})^2} = \left(1 - \frac{1}{1 + e^{-n}} \right) \left(\frac{1}{1 + e^{-n}} \right) = (1 - a^1)(a^1)$$

$$f^2(n) = \frac{d}{dn}(n) = 1$$

Lan Truyền Ngược

$$\mathbf{s}^2 = -2 \mathbf{F}^2(\mathbf{n}^2)(\mathbf{t} - \mathbf{a}) = -2 \left[f^2(n^2) \right] (1.26 \mathbf{i}) = -2 [1] (1.26 \mathbf{i}) = -2.522$$

$$\mathbf{s}^1 = \mathbf{F}^1(\mathbf{n}^1)(\mathbf{W}^2)^T \mathbf{s}^2 = \begin{bmatrix} (1-a_1^1)(a_1^1) & 0 \\ 0 & (1-a_2^1)(a_2^1) \end{bmatrix} \begin{bmatrix} 0.09 \\ -0.1 \end{bmatrix} \begin{bmatrix} -2.52 \\ 2 \end{bmatrix}$$

$$\mathbf{s}^1 = \begin{bmatrix} (1 - 0.32)(0.32) & 0 \\ 0 & (1 - 0.36)(0.36) \end{bmatrix} \begin{bmatrix} 0.09 \\ -0.1 \end{bmatrix} \begin{bmatrix} -2.52 \\ 2 \end{bmatrix}$$

$$\mathbf{s}^1 = \begin{bmatrix} 0.218 & 0 \\ 0 & 0.233 \end{bmatrix} \begin{bmatrix} -0.22 \\ 0.429 \end{bmatrix} = \begin{bmatrix} -0.0495 \\ 0.0997 \end{bmatrix}$$

Cập nhật trọng số

$$\alpha = 0.1$$

$$\mathbf{W}^2(1) = \mathbf{W}^2(0) - \alpha \mathbf{s}^2(\mathbf{a}^1)^T = \begin{bmatrix} 0.09 & -0.1 \end{bmatrix} - 0.1 \begin{bmatrix} -2.52 \\ 2 \end{bmatrix} \begin{bmatrix} 0.32 & 10.36 \end{bmatrix}$$

$$\mathbf{W}^2(1) = \begin{bmatrix} 0.17 & 1 - 0.077 \end{bmatrix}$$

$$\mathbf{b}^2(1) = \mathbf{b}^2(0) - \alpha \mathbf{s}^2 = \begin{bmatrix} 0.48 \end{bmatrix} - 0.1 \begin{bmatrix} -2.52 \end{bmatrix} = \begin{bmatrix} 0.73 \end{bmatrix}$$

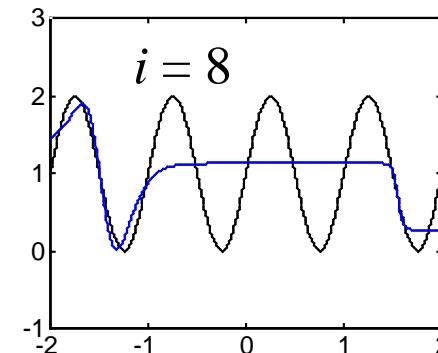
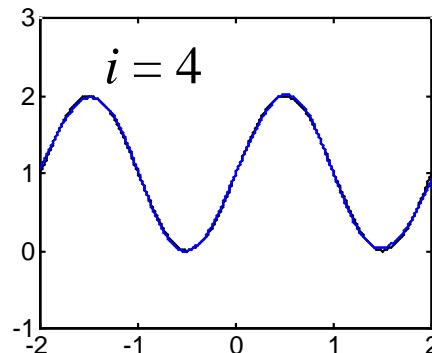
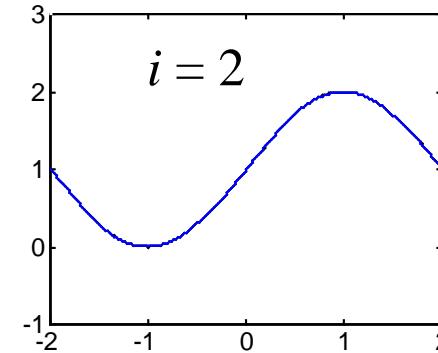
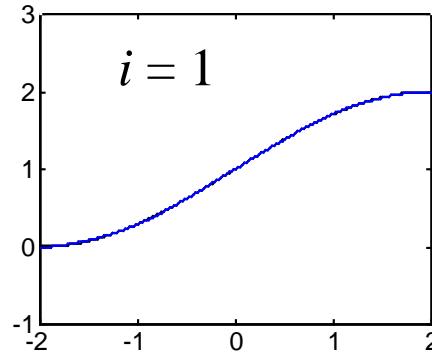
$$\mathbf{W}^1(1) = \mathbf{W}^1(0) - \alpha \mathbf{s}^1(\mathbf{a}^0)^T = \begin{bmatrix} -0.27 \\ -0.41 \end{bmatrix} - 0.1 \begin{bmatrix} -0.0495 \\ 0.0997 \end{bmatrix} \begin{bmatrix} 1 \end{bmatrix} = \begin{bmatrix} -0.265 \\ -0.420 \end{bmatrix}$$

$$\mathbf{b}^1(1) = \mathbf{b}^1(0) - \alpha \mathbf{s}^1 = \begin{bmatrix} -0.48 \\ -0.13 \end{bmatrix} - 0.1 \begin{bmatrix} -0.0495 \\ 0.0997 \end{bmatrix} = \begin{bmatrix} -0.475 \\ -0.140 \end{bmatrix}$$

Lựa chọn Cấu trúc mạng

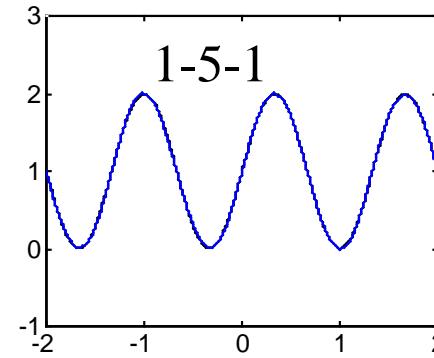
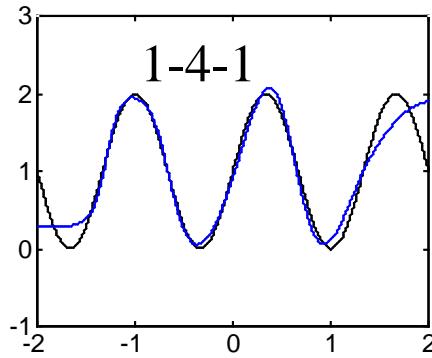
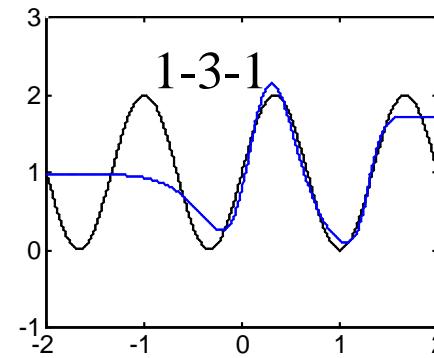
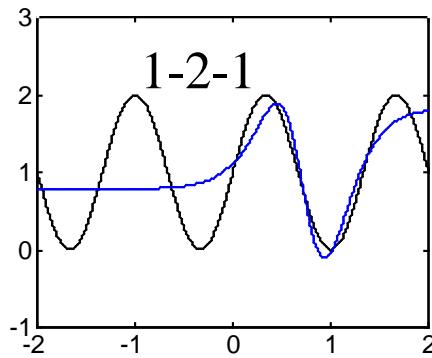
$$g(p) = 1 + \sin\left(\frac{i\pi}{4}p\right)$$

1-3-1 Network



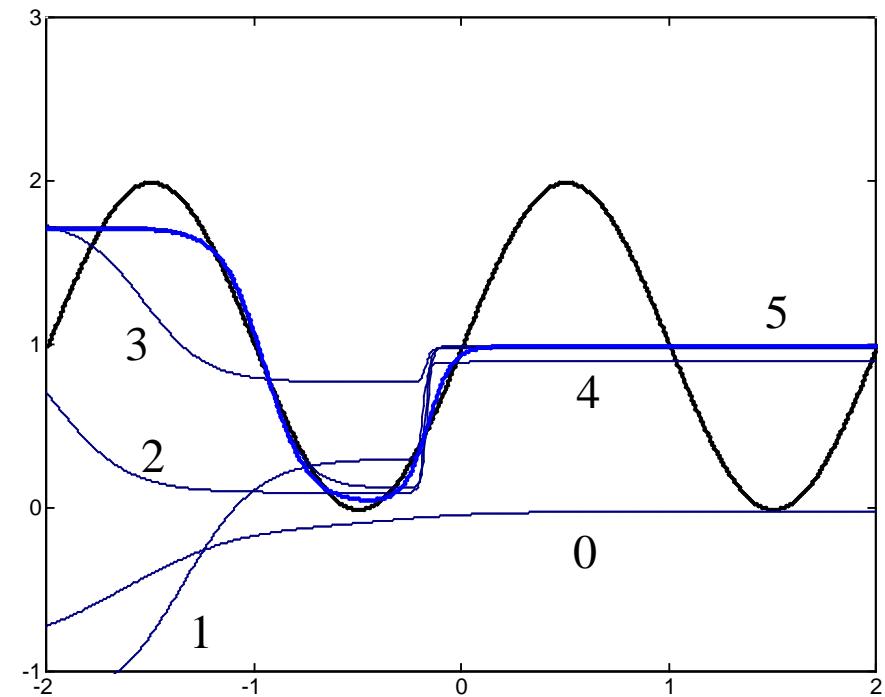
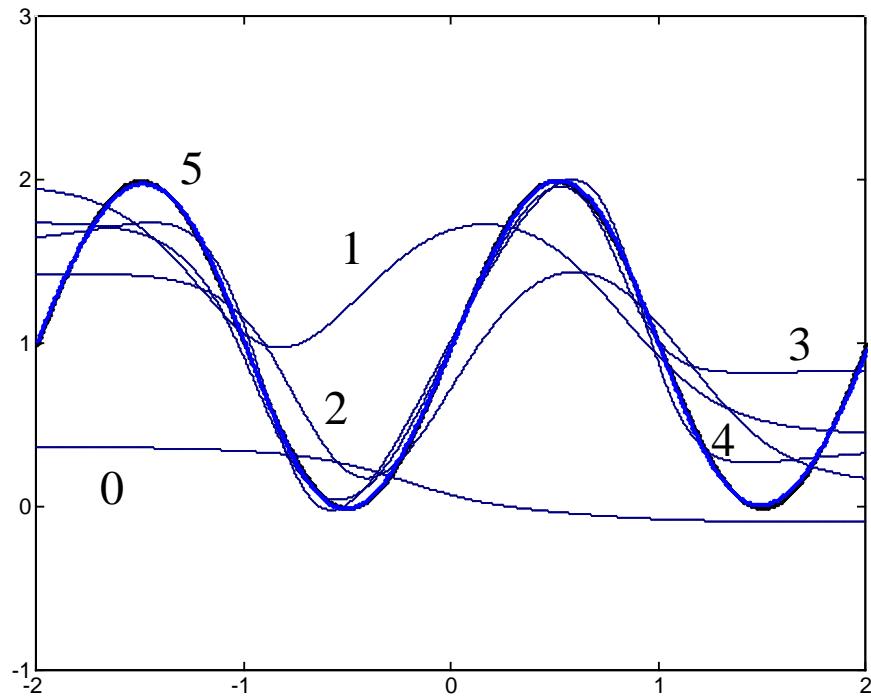
Lựa chọn cấu trúc mạng

$$g(p) = 1 + \sin\left(\frac{6\pi}{4}p\right)$$



Hội tụ

$$g(p) = 1 + \sin(\pi p)$$



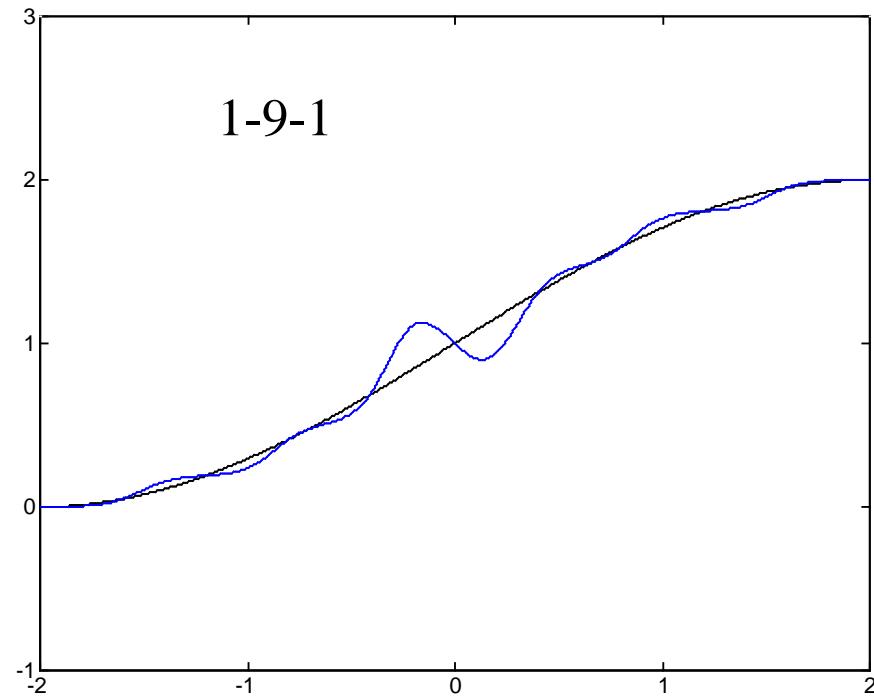
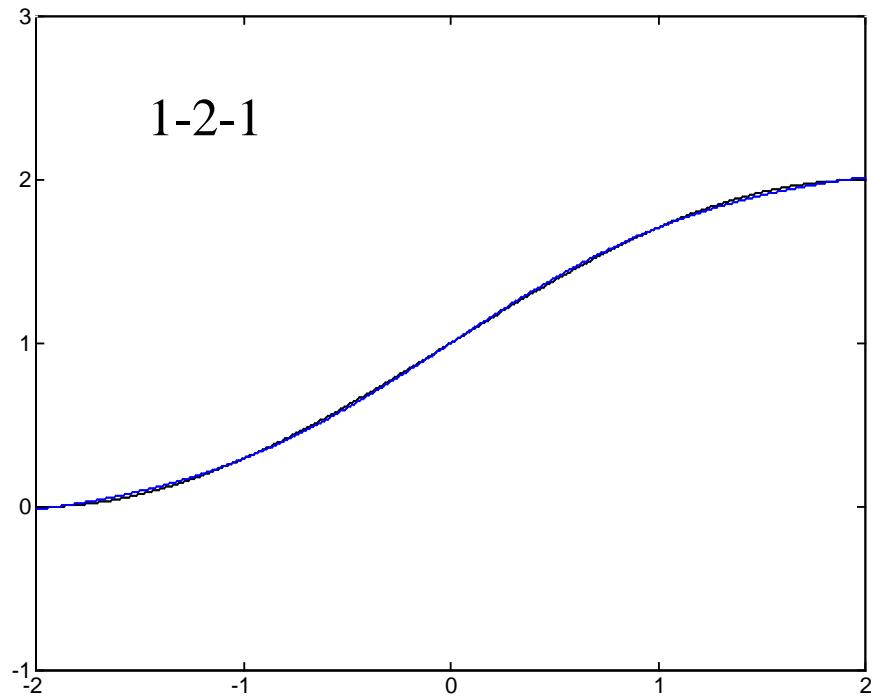
Hội tụ đến cực trị toàn cục

Hội tụ đến cực trị địa phương

Khái quát hóa

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \dots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

$$g(p) = 1 + \sin\left(\frac{\pi}{4}p\right) \quad p = -2, -1.6, -1.2, \dots, 1.6, 2$$



Định Lý Kolmogov và Độ Phức Tạp Học

- Bài toán số 13 của David Hilbert "Nghiệm của đa thức bậc 7 không thể biểu diễn bằng chồng hàm của các hàm 2 biến cụ thể là đa thức sau đây:
 $f^7 + xf^3 + yf^2 + zf + 1 = 0$ không thể giải được bằng các hàm hai biến".
- Ví dụ về chồng hàm hai biến để giải phương trình bậc 2.

Định Lý Kolmogorov và Độ Phức Tập Học

- Năm 1957, Kolmogorov (Arnold, Lorenz) chứng minh giả thiết đưa ra trong bài toán của Hilbert là sai. Thậm chí chứng minh kết quả mạnh hơn: mọi hàm liên tục đều biểu diễn được bằng chồng các hàm một biến chỉ dùng phép toán nhân và cộng.
- Định Lý Kolmogorov: $f:[0,1]^n \rightarrow [0,1]$ là hàm liên tục thì tồn tại các hàm một biến $g, h_i, i=1,2,\dots,2n+1$ và các hằng số λ_i sao cho:
 - $f(x_1, x_2, \dots, x_n) = \sum_{j=1,2n+1} g(\sum_{i=1,n} \lambda_i h_j(x_i))$
 - Định lý cho mạng Neural (Baron, 1993):
 - *Mạng Perceptron hướng tiến một lớp ẩn dùng hàm chuyển sigmoid có thể xấp xỉ bất cứ hàm khả tích Lobe nào trên khoảng $[0, 1]$.*

Định Lý Kolmogorov và Độ Phức Tập Học

- Mặc dù vậy các định lý chỉ đưa ra sự tồn tại mà không đưa ra được thuật toán cho việc xác định cấu trúc mạng (số neuron trong tầng ẩn) hay các trọng số.
- Định lý NP về học cho mạng Neural (Judd, 1990):
- *Bài toán tìm các trọng số tối ưu cho mạng Neural đa lớp có hàm chuyển hardlims là NP đầy đủ.*
- Lưu ý:
 - - do đó thuật toán BP không đảm bảo tìm được nghiệm tối ưu, thậm chí không đảm bảo sự hội tụ.
 - - Việc xác định cấu trúc mạng và một số yếu tố của thuật toán học còn mang tính kinh nghiệm, Heuristic.

Một số Heuristics cho BP

- Cập nhật theo chế độ tuần tự (online) hay batch (epoch):
 - Thường việc học theo chế độ tuần tự giúp BP hội tụ nhanh hơn, đặc biệt khi dữ liệu lớn và dư thừa.
- Chuẩn hoá giá trị đầu ra:
 - Đảm bảo giá trị đầu ra nằm trong miền giá trị của hàm chuyển trên các neuron đầu ra tương ứng (thường là nằm trong khoảng $[a+\varepsilon, b-\varepsilon]$).
- Chuẩn hoá giá trị đầu vào:
 - Đảm bảo giá trị trung bình gần 0 hoặc nhỏ so với độ lệch tiêu chuẩn (stdev). Các giá trị tốt nhất phải độc lập với nhau.
- Khởi tạo giá trị trong số:
 - Uniform random in $[-\alpha, \alpha]$; $[-\alpha, -\beta] \cup [\alpha, \beta]$
- Kết thúc sớm:
 - Khi liên tiếp n epoch training mà không có sự cải thiện đáng kể lỗi hay không có sự thay đổi đáng kể của các trọng số.

Một số Heuristics cho BP

- Tốc độ học:
 - *Tốc độ học của các Neuron nên đều nhau vì vậy, neuron tầng sau (thường có gradient lớn hơn tầng trước) nên có tốc độ học nhỏ hơn tầng trước, Neuron có ít input nên có tốc độ học lớn hơn Neuron có nhiều input.*
- Kiểm tra chéo (crossvalidation):
 - *Tách tập dữ liệu ra làm hai tập độc lập (training and testing). Tỷ lệ thường là 2/3:1/3. Thực hiện việc học trên tập training và kiểm tra khả năng khai quát hóa của mạng trên tập testing.*
- Luật phân lớp tối ưu:
 - *Dùng M neuron đầu ra cho bài toán phân M lớp sử dụng luật cạnh tranh winner-take-all.*
- Xác định số neuron lớp ẩn:
 - *Các ước lượng cận thô (số lượng dữ liệu, chiều VC). Phương pháp: Incremental, Decremental.*

Đọc thêm

1. M.T. Hagan, H.B. Demuth, and M. Beale, *Neural Network Design*, PWS Publishing.
2. Giáo trình, chương 19.
3. MIT Courseware: ch8, ch9.

Câu Hỏi ôn tập

1. Trình bày cấu trúc mạng Neuron Perceptron đa lớp?
2. Trình bày giải thuật học lan truyền ngược cho MLP.
3. Trình bày định lý Kolmogorov và ứng dụng cho MLP.
4. Cài đặt thuật giải BP cho MLP.
5. Ứng dụng MLP để giải các bài toán như: Classification và Regression.