Nhập môn TTNT – Tóm tắt

#by DTD for 2021 classes (128702 and 126523)

# L1. Giới thiệu

Lĩnh vực liên quan TTNT và một số thành công gần đây

* Các lĩnh vực liên quan TTNT, ảnh hưởng của TTNT
* Các thành công gần đây (đặc điểm, kết quả nổi bật)

Khái niệm TTNT

* Các quan điểm về TTNT
* Kiểm chứng Turing, tác tử hợp lý

Các nền tảng TTNT, lịch sử TTNT

* Các nền tảng (lĩnh vực) liên quan
* Các sự kiện quan trọng, các giai đoạn phát triển (giai đoạn thành công, khó khăn)

# **L2. Tác tử thông minh**

## **1, Khái niệm tác tử và các khái niệm liên quan**

### **- Định nghĩa tác tử**

*Tác tử* là bất cứ cái gì (con người, người máy, software robots, các bộ ổn nhiệt,...) có khả năng *cảm nhận (nhận biết) môi trường* xung quanh nó thông qua *các bộ phận cảm biến* (sensors) và *hành động* phù hợp theo môi trường đó thông qua *các bộ phận hoạt động* (actuators)

### **- Các thành phần, hàm, chương trình, kiến trúc của tác tử**

Hàm tác tử: là hàm ánh xạ từ lịch sử nhận thức tới các hành động:

*f*: P\* → A

Chương trình tác tử: hoạt động (chạy) dựa trên kiến trúc thực tế của hàm *f*

Tác tử = Kiến trúc + Chương trình

## **2, Phân biệt các môi trường công việc, các kiểu tác tử**

### **- Môi trường công việc PEAS**

PEAS:

• Performance measure: Tiêu chí đánh giá hiệu quả hoạt động

• Environment: Môi trường xung quanh

• Actuators: Các bộ phận hành động

• Sensors: Các bộ phận cảm biến

Để thiết kế một tác tử thông minh (hợp lý), trước tiên cần phải xác định (thiết lập) các giá trị của các thành phần của PEAS

### **- Đặc điểm, phân biệt các kiểu môi trường công việc**

Đặc điểm: Môi trường trong thực tế thường có các đặc điểm: chỉ có thể quan sát được một phần, ngẫu nhiên, liên tiếp, thay đổi (động), liên tục, đa tác tử

Phân biệt các kiểu môi trường:

**+, Có thể quan sát được hoàn toàn** (hay có thể quan sát được một phần)?

• Các bộ cảm biến của một tác tử cho phép nó truy cập tới *trạng thái đầy đủ* của môi trường tại mỗi thời điểm

**+, Xác định** (hay ngẫu nhiên)?

• Trạng thái tiếp theo của môi trường được xác định hoàn toàn dựa trên trạng thái hiện tại và hành động của tác tử (tại trạng thái hiện tại này)

• Nếu một môi trường là xác định, ngoại trừ đối với các hành động của các tác tử khác, thì gọi là *môi trường chiến lược*

**+, Phân đoạn** (hay liên tiếp)?

• Lịch sử kinh nghiệm của tác tử được chia thành *các giai đoạn* (chương/hồi)

• Mỗi giai đoạn bao gồm việc nhận thức của tác tử và hành động mà nó thực hiện

• Ở mỗi giai đoạn, việc lựa chọn hành động để thực hiện chỉ phụ thuộc vào giai đoạn đó (không phụ thuộc vào các giai đoạn khác)

**+, Tĩnh** (hay động)?

• Môi trường không thay đổi trong khi tác tử cân nhắc (xem nên đưa ra hành động nào)

• Môi trường *bán động* (semi-dynamic) là môi trường mà khi thời gian trôi qua thì nó (môi trường) không thay đổi, nhưng hiệu quả hoạt động của tác tử thì thay đổi

• Ví dụ: Các chương trình trò chơi có tính giờ

**+, Rời rạc** (hay liên tục)?  
• Tập các nhận thức và các hành động là hữu hạn, được định nghĩa phân biệt rõ ràng

+,**Tác tử đơn lẻ** (hay đa tác tử)?

• Một tác tử hoạt động độc lập (không phụ thuộc / liên hệ với các tác tử khác) trong một môi trường

### **- Đặc điểm, phân biệt các kiểu tác tử**

4 kiểu tác tử cơ bản

**• Tác tử phản xạ đơn giản (simple reflex agents)**

→Hành động theo một quy tắc (luật) có điều kiện phù hợp với trạng thái hiện thời (của môi trường)

**• Tác tử phản xạ dựa trên mô hình (model-based reflex agents)**

→Sử dụng một mô hình nội bộ để giám sát trạng thái hiện tại của môi trường

→Lựa chọn hành động: giống như đối với tác tử phản xạ đơn giản

**• Tác tử dựa trên mục tiêu (goal-based agents)**

→Biết về trạng thái hiện tại của môi trường: chưa đủ → Cần biết thêm thông tin về mục tiêu

• Trạng thái hiện tại của môi trường: Ở một ngã tư, xe taxi có thể rẽ trái, rẽ phải, hoặc đi thẳng

• Thông tin về mục tiêu: xe taxi cần đi tới đích đến của hành khách

→Tác tử dựa trên mục tiêu:

• Theo dõi trạng thái hiện tại của môi trường

• Lưu giữ một tập các mục tiêu (cần đạt được)

• Chọn hành động cho phép (rốt cuộc) sẽ đạt đến các mục tiêu

**• Tác tử dựa trên lợi ích (utility-based agents)**

→Trong nhiều môi trường, thông tin về các mục tiêu không đủ để đánh giá hiệu quả của các hành động

• Có rất nhiều chuỗi các hành động cho phép taxi đi đến đích (tức là đạt đến mục tiêu)

• Nhưng: chuỗi hành động nào nhanh hơn, an toàn hơn, đáng tin cậy hơn, chi phí thấp hơn?

→Cần sự đánh giá lợi ích đối với tác tử

• Hàm lợi ích (utility function)

→Ánh xạ từ **chuỗi** các trạng thái của môi trường tới một giá trị số thực (thể hiện mức lợi ích đối với tác tử)

Tài liệu, demo (IT3160 L2 WE.docx)

* Ví dụ về 6 cặp môi trường làm việc
* Ví dụ về các kiểu tác tử

# **L3. Tìm kiếm cơ bản (Uninformed search)**

## **1, Giải quyết vấn đề bằng tìm kiếm**

* Giải quyết vấn đề bằng tìm kiếm, phát biểu bài toán trạng thái đơn
* Cây tìm kiếm, giải thuật tìm kiếm theo cấu trúc cây

## **2, Phân biệt các chiến lược tìm kiếm cơ bản (BFS, UCS, DFS, LDS, IDS)**

* Nguyên tắc phát triển các nút (độ sâu thấp nhất trước, chi phí thấp nhất trước, theo các nhánh)
* Tập các node sẽ được mở rộng fringe (hàng đợt FIFO hay ngăn xếp LIFO)
* So sánh về thực thi các chiến lược tìm kiếm

### **2.1, Thuật toán BFS (Tìm kiếm theo chiều rộng)**

+, Nguyên tắc phát triển các nút: phát triển các nút chưa xét theo chiều rộng – Các nút được xét theo thứ tự độ sâu tăng dần

+, fringe là một cấu trúc kiểu hàng đợi FIFO (các nút mới được bổ sung vào cuối của fringe)

+, Đặc điểm của BFS:

Diagram

Description automatically generated

(d là điểm tìm thấy lời giải)

Hoàn chỉnh khi b hữu hạn: nếu b là hữu hạn thì bài toán sẽ luôn tìm đc giải pháp

Chi phí = 1 cho mỗi bước là tối ưu:

### **2.2, Thuật toán UCS (Tìm kiếm với chi phí cực tiểu)**

+, Nguyên tắc phát triển các nút: Phát triển các nút chưa xét có chi phí thấp nhất – Các nút được xét theo thứ tự chi phí (từ nút gốc đến nút đang xét) tăng dần

+, fringe là một cấu trúc hàng đợi, trong đó các phần tử được sắp xếp theo chi phí đường đi

=> Trở thành phương pháp tìm kiếm theo chiều rộng, nếu các chi phí ở mỗi bước (mỗi cạnh của cây tìm kiếm) là như nhau

+, Đặc điểm của UCS:

Graphical user interface, text, application

Description automatically generated

### **2.3, Thuật toán DFS (Tìm kiếm theo chiều sâu)**

+, Nguyên tắc phát triển các nút: Phát triển các nút chưa xét theo chiều sâu – Các nút được xét theo thứ tự độ sâu giảm dần

+, fringe là một cấu trúc kiểu ngăn xếp LIFO (Các nút mới được bổ sung vào đầu của fringe)

+, Đặc điểm của DFS:

Text

Description automatically generated

### **2.4, Thuật toán DLS (Tìm kiếm giới hạn độ sâu)**

+, Nguyên tắc phát triển các nút: Phương pháp tìm kiếm theo chiều sâu (DFS) + Sử dụng giới hạn về độ sâu l trong quá trình tìm kiếm

→ các nút ở độ sâu l không có nút con

### **2.5, Thuật toán IDS (Tìm kiếm sâu dần)**

+, Giải thuật tìm kiếm sâu dần

* Áp dụng giải thuật DFS đối với các đường đi (trong cây) có độ dài <=1
* Nếu thất bại (không tìm được lời giải), tiếp tục áp dụng giải thuật DFS đối với các đường đi có độ dài <=2
* Nếu thất bại (không tìm được lời giải), tiếp tục áp dụng giải thuật DFS đối với các đường đi có độ dài <=3
* ...(tiếp tục như trên, cho đến khi: 1) tìm được lời giải, hoặc 2) toàn bộ cây đã được xét mà không tìm được lời giải)

+, Đặc điểm:

Text, letter

Description automatically generated

### **2.6, So sánh DLS và IDS**

• Với độ sâu *d* và hệ số phân nhánh *b*, thì số lượng các nút được sinh ra trong giải thuật tìm kiếm giới hạn độ sâu là:

*NDLS = b0 + b1 + b2 + ... + bd-2 + bd-1 + bd*• Số lượng các nút được sinh ra trong giải thuật tìm kiếm sâu dần là:  
NIDS = (d+1).b0 + d.b1 + (d-1).b2 + ... + 3.bd-2 +2.bd-1 + 1.bd

Ví dụ với *b = 10*, *d = 5*:  
• NDLS = 1 + 10 + 100 + 1,000 + 10,000 + 100,000 = 111,111

• NIDS =6+50+400+3,000+20,000+100,000=123,456

• Lãng phí = (123,456 - 111,111)/111,111 = 11%

### **2.7, So sánh giữa các giải thuật tìm kiếm cơ bản**

Table

Description automatically generated

Tài liệu, demo: chương trình minh họa khác nhau giữa các giải thuật BFS, UCS, DFS, LDS, IDS

* Python code: main10-14.py (Tóm tắt kết quả chạy chương trình: IT3160 L3-L6 demo.docx)

# **L4. Tìm kiếm với tri thức bổ sung (Informed search)**

## **1, Các giải thuật best-first search (Greedy best-first search và A\*)**

* Hàm đánh giá, ước lượng chấp nhận được
* Đánh giá thuật toán (tính tối ưu).
* So sánh với uniformed search (UFS)

### **1.1, Thuật toán Greedy best-first search**

• Hàm đánh giá *f(n)* là hàm *heuristic h(n)*• Hàm heuristic *h(n)* đánh giá chi phí để đi từ nút hiện tại *n* đến nút đích (mục tiêu)

• Ví dụ: Trong bài toán tìm đường đi từ Arad đến Bucharest, sử dụng: *hSLD(n)* = Ước lượng khoảng cách đường thẳng (“chim bay”) từ thành phố hiện tại *n* đến Bucharest

• Phương pháp tìm kiếm Greedy best-first search sẽ xét (phát triển) nút “có vẻ” gần với nút đích (mục tiêu) nhất

• Đặc điểm:

Graphical user interface, text, application

Description automatically generated

### **1.2, Thuật toán A\* search**

• Ý tưởng: Tránh việc xét (phát triển) các nhánh tìm kiếm đã xác định (cho đến thời điểm hiện tại) là có chi phí cao

• Sử dụng hàm đánh giá *f(n) = g(n)* + *h(n)*

* *g(n)* = chi phí từ nút gốc cho đến nút hiện tại *n*
* *h(n)* = chi phí ước lượng từ nút hiện tại *n* tới đích
* *f(n)* = chi phí tổng thể ước lượng của đường đi qua nút hiện tại *n* đến đích

• Đặc điểm:

Text

Description automatically generated

## **2, Các giải thuật tìm kiếm cục bộ (hill-climbing, simulated annealing, local beam, genetic algorithm)**

* Các cá thể (giả thuyết) trong tìm kiếm cục bộ (khác biệt với các PP tìm kiếm ở trên)
* Đặc điểm, phương thức mở rộng node (các nodes), đánh giá các thuật toán
* Các toán tử di truyền

Tài liệu, demo: chương trình minh họa thuật toán di truyền (GA)

* Python code: mainGA.py (Tóm tắt kết quả chạy chương trình: IT3160 L3-L6 demo.docx)

# **L5 Tìm kiếm có đối thủ**

## **1, Giải thuật tìm kiếm có đối thủ (minimax, cắt tỉa α-β)**

* Bài toán tìm kiếm có đối thủ
* Đánh giá thuật toán
* Tri thức bổ sung trong tìm kiếm có đổi thủ

### **1.1, Thuật toán Minimax**

Diagram

Description automatically generated

- Lượt của Max: Chọn giá trị max của các con

- Lượt của Min: Chọn giá trị min của các con

- Đặc điểm:

Graphical user interface, text, application

Description automatically generated

### **1.2, Thuật toán cắt tỉa alpha-beta**

Text

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram, radar chart

Description automatically generated

Diagram

Description automatically generated

Diagram, radar chart

Description automatically generated

Tài liệu, demo: chương trình minh họa cờ ca-ro, cờ vua

* Python code (mainMmTictac.py, mainMmChessLib.py, mainMmChessNolib.py)
* Tóm tắt kết quả chạy chương trình demo (IT3160 L3-L6 demo.docx)

# L6 Tìm kiếm thỏa mãn ràng buộc

Khái niệm

* Ràng buộc (constraint), bài toán CSP, tìm kiếm bằng kiểm thử,
* Tìm kiếm quay lui (back tracking)
* Cải thiện hiệu quả của tìm kiếm quay lui (thứ tự xét, kiểm tra tiến, AC-3)
* Tìm kiếm cục bộ cho CSP

Tài liệu, demo: tìm kiếm quay lui

* Python code (mainCSP\_BT.py, CSP\_FC.py)
* Tóm tắt kết quả chạy chương trình demo (IT3160 L3-L6 demo 2.0.docx)

# L8 Biểu diễn tri thức (giới một phần)

Khái niệm, các phương pháp biểu diễn trí thức:

* Biểu diễn tri thức bằng luật (production rules), khung (frames)
  + Đặc điểm phương pháp biểu diễn, sử dụng luật trong suy diễn, thủ tục trong biểu diễn khung
* Mạng ngữ nghĩa, ontology: không thi

# L9-11 Logic và suy diễn

Các khái niệm liên quan logic, suy diễn logic

Logic định đề

## **1, Toán tử logic**

¬S1 là đúng, khi và chỉ khi S1 là sai

S1 ∧ S2 là đúng, khi và chỉ khi S1 là đúng và S2 là đúng

S1 ∨ S2 là đúng, khi và chỉ khi S1 là đúng hoặc S2 là đúng

S1 ⇒ S2 là đúng, khi và chỉ khi S1 là sai hoặc S2 là đúng; là sai, khi và chỉ khi S1 là đúng và S2 là sai

S1 ⇔ S2 là đúng, khi và chỉ khi S1⇒S2 là đúng và S2⇒S1 là đúng

## **2, Tương đương logic**

Text

Description automatically generated

## **3, Chứng minh bằng bảng chân lý, luật suy diễn**

Table

Description automatically generated

## **4, Luật suy diễn**

Graphical user interface, text, application

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

## **5, Dạng chuẩn**

### **5.1, Các dạng chuẩn**

Graphical user interface, text

Description automatically generated

### **5.2, Chuyển đổi về dạng chuẩn CNF**

Text

Description automatically generated

Hợp giải, suy diễn

1, Chứng minh bằng hợp giải

Text

Description automatically generated

Text

Description automatically generated

Diagram

Description automatically generated

### **5.3, Suy diễn tiến, lùi**

Logic vị từ

* Các ký hiệu (hằng, biến, hàm, vị từ), lượng từ (tồn tại, với mọi)
* biểu diễn câu với logic vị từ

Tham khảo

* Logic và suy diễn: tóm tắt, bài tập (L9 practice)

# L12 Học máy

Khái niệm chung

* Bài toán học máy, quá trình học máy, các vấn đề trong học máy (over-fitting, under-fitting)
* Học có giám sát – không giám sát
* Một số phương pháp học máy ở (sau đây): là các phương pháp phân lớp, học máy có giám sát

Láng giềng gần nhất (K-nearest neighbors k-NN)

* Hàm tính khoảng cách
* Ưu – nhược điểm

## **Naive-bayes**

* Xác xuất tiên nghiệm, có điều kiện, xác xuất hậu nghiệm
* Xác định phân lớp dựa trên xác suất (giải thuật naive-bayes)

Giải bài tập Naïve-bayes:

Text

Description automatically generated

Bài 1:

Với C={true, false}

+,P(c1=true) = 6/10

+, P(c2=false) = 4/10

Xác xuất có thể xảy ra của ví dụ zj đối với phân lớp c1:

P(zj(A=1) | c1) = 2/6

P(zj(B=0) | c1) = 1/6

P(zj(C=1) | c1) = 3/6

=> P(zj | c1) = 2/6 \* 1/6 \* 3/6

Xác xuất có thể xảy ra của ví dụ zj đối với phân lớp c2:

P(zj(A=1) | c2) = 3/4

P(zj(B=0) | c2) = 2/4

P(zj(C=1) | c2) = 4/4

=> P(zj | c2) = 3/4 \* 2/4 \* 4/4

Xác định phân lớp có thể nhất:

P(c1)\*P(zj|c1) = 6/10 \* 2/6 \* 1/6 \* 3/6=0.01667

P(c2)\*P(zj|c2) = 4/10 \* 3/4 \* 2/4 \* 4/4=0.15

=> True

Bài 2:

Ta có C={Yes, No}

+,P(c1=yes) = 5/10

+, P(c2=no) = 5/10

Xác xuất có thể xảy ra của ví dụ zj đối với phân lớp c1:

P(zj(Color=Red) | c1) = 3/5

P(zj(Type=SUV) | c1) = 1/5

P(zj(Origin=Domestic) | c1) = 2/5

=> P(zj | c1) = 3/5 \* 1/5 \* 2/5

Xác xuất có thể xảy ra của ví dụ zj đối với phân lớp c1:

P(zj(Color=Red) | c2) = 2/5

P(zj(Type=SUV) | c2) = 3/5

P(zj(Origin=Domestic) | c2) = 3/5

=> P(zj | c1) = 2/5 \* 3/5 \* 3/5

Xác định phân lớp có thể nhất:

P(c1)\*P(zj|c1) = 5/10 \* 3/5 \* 1/5 \* 2/5=0.024

P(c2)\*P(zj|c2) = 5/10 \* 2/5 \* 3/5 \* 3/5=0.0.72

=> Xe sẽ bị trộm

Cây quyết định

* Thuật toán ID3
* Entropy – Information gain

Tài liệu, chương trình demo: L12\_ML.py, L12 demo.docx

# Các hình thức thi, kiểm tra

Thi trắc nghiệm: Vận dụng kiến thức đã học để đánh giá đúng-sai, lựa chọn các phương án phù hợp

Thi đề mở: Vận dụng các kiến thức đã học vào giải quyết các vấn đề (bài tập) đặt ra