

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**Đề tài**

**DEFINITION OF A CONSTRAINT SATISFACTION**

**PROBLEM**

**MÔN HỌC: TRÍ TUỆ NHÂN TẠO**

**GVHD: ThS. Lê Minh Tân**

**MÃ LỚP: ARIN330585\_22\_2\_02**

|  |  |
| --- | --- |
| ***Nhóm sinh viên thực hiện :*** |  |
| Nguyễn Trung Phiên | 21110593 |
| Ngô Quang Nghĩa | 21110559 |
| Lê Văn Vũ | 21110943 |
| Nguyễn Chí Thanh | 21110644 |
| Nguyễn Phạm Mạnh Hóa | 21110885 |
| Nguyễn Thành Nhơn | 21110907 |
| Đinh Quang Anh | 21110863 |

**TP Hồ Chí Minh , tháng 4 năm 2023**

**MỤC LỤC**

[A. PHẦN MỞ ĐẦU](#page3) [1](#page3)

[B. PHẦN NỘI DUNG](#page4) [2](#page4)

[CHƯƠNG 1: ĐỊNH NGHĨA BÀI TOÁN THỎA MÃN RÀNG BUỘC](#page4) [2](#page4)

1.1. Định nghĩa bài toán thỏa mãn ràng buộc 2

[1.2.](#page4) [Giải pháp cho bài toán thỏa mãn ràng buộc](#page4) [2](#page4)

[CHƯƠNG 2: MỘT SỐ VÍ DỤ VỀ CÁC BÀI TOÁN THỎA MÃN RÀNG BUỘC](#page5) [3](#page5)

[2.1. Bài toán tô màu đồ thị (Map Coloring)](#page5) [3](#page5)

[2.1.1. Giới thiệu sơ lược](#page5) [3](#page5)

[2.1.2. Định nghĩa bài toán dưới dạng CSP](#page5) [3](#page5)

[2.1.3. Lời giải cho bài toán](#page6) [4](#page6)

[2.1.4. Lý do nên xây dựng một vấn đề dưới dạng CSP](#page6) [4](#page6)

[2.2. Lập lịch sản xuất (Job-Shop Scheduling)](#page6) [5](#page6)

[2.2.1. Giới thiệu sơ lược](#page6) [5](#page6)

[2.2.2. Định nghĩa bài toán dưới dạng CSP](#page7) [5](#page7)

[2.2.3. Lời giải](#page8) [cho bài toán](#page8) [6](#page8)

[2.3 Bài toán giải mã (cryptarithmetic puzzles)](#page8) [6](#page8)

[2.3.1 Giới thiệu sơ lược](#page8) [6](#page8)

[2.3.2. Định nghĩa bài toán dưới dạng CSP](#page9) [7](#page9)

[2.3.3. Lời giải cho bài toán](#page9) [7](#page9)

[CHƯƠNG 3: CÁC BIẾN THỂ CỦA BÀI TOÁN THỎA MÃN RÀNG BUỘC](#page10) [8](#page10)

[3.1. Miền rời rạc và miền liên tục](#page10) [8](#page10)

[3.2. Các loại ràng buộc của bài toán thỏa mãn ràng buộc](#page10) [8](#page10)

[3.2.1. Ràng buộc tuyến tính và ràng buộc phi tuyến](#page10) [8](#page10)

[3.2.2. Ràng buộc đơn (unary constraint), ràng buộc nhị phân (binary constraint) và ràng buộc](#page10)

bậc cao (higher-order constraints) 9

[3.2.3 Ràng buộc toàn cục (global constraint)](#page10) [9](#page10)

[3.2.4 Ràng buộc ưu tiên (preference constraint)](#page10) [9](#page10)

[3.3. Chuyển đổi sang ràng buộc nhị phân](#page12) [10](#page12)

[C. PHẦN KẾT LUẬN](#page14) [12](#page14)

D. [TÀI LIỆU THAM KHẢO](#page15) [13](#page15)

**A. PHẦN MỞ ĐẦU**

Các bài toán thỏa mãn ràng buộc (CSPs) là các vấn đề toán học được định nghĩa là một tập hợp các đối tượng có trạng thái phải thỏa mãn một số ràng buộc hoặc giới hạn. CSP biểu diễn các thực thể trong một bài toán như một tập hợp đồng nhất các ràng buộc hữu hạn đối với các biến, được giải bằng các phương pháp thỏa mãn ràng buộc. Trong CSP, chúng ta có một tập hợp các biến với các miền đã biết và một tập hợp các ràng buộc áp đặt các hạn chế đối với các giá trị mà các biến đó có thể nhận. Nhiệm vụ của chúng ta là gán một giá trị cho mỗi biến để chúng thỏa mãn tất cả các ràng buộc. [CSP thường thể](https://en.wikipedia.org/wiki/Complexity_of_constraint_satisfaction) hiện độ phức tạp cao, đòi hỏi sự kết hợp giữa phương pháp phỏng [đoán](https://en.wikipedia.org/wiki/Heuristics) và [tìm kiếm tổ](https://en.wikipedia.org/wiki/Combinatorial_search) hợp để có thể được giải quyết trong một khoảng thời gian hợp lý.Các bài toán thỏa mãn ràng buộc (CSPs) thường được sử dụng để mô hình hóa các vấn đề trong thế giới thực trong AI, chẳng hạn như lập lịch trình, lập kế hoạch sản xuất và mã hóa…

Phần nội dung của báo cáo này sẽ trình bày về định nghĩa bài toán thỏa mãn ràng buộc (CSP), các ví dụ về các bài toán thỏa mãn ràng buộc như Map coloring, Job-shop scheduling, cryptarithmetic puzzles cũng như đề cập đến một số biến thể của CSP.

1

**B. PHẦN NỘI DUNG**

**CHƯƠNG 1: ĐỊNH NGHĨA BÀI TOÁN THỎA MÃN RÀNG BUỘC**

**1.1. Định nghĩa bài toán thỏa mãn ràng buộc**

Một vấn đề thỏa mãn ràng buộc bao gồm ba thành phần X,D và C:

X là một tập hợp các biến, *{X1,…,Xn}*

D là một tập hợp các miền, *{D1,…,Dn},* một miền cho mỗi biến.

C là một tập hợp các ràng buộc chỉ định các kết hợp giá trị cho phép.

Một miền Di bao gồm một tập hợp các giá trị cho phép {v1,…,vn} cho biến Xi . Ví dụ, một biến Boolean sẽ có miền {true,false}. Các biến khác nhau có thể có các miền khác nhau với các kích cỡ khác nhau.

Mỗi ràng buộc Cj bao gồm một cặp (scope, rel), trong đó scope là một bộ các biến tham gia vào ràng buộc và rel là một quan hệ xác định các giá trị mà các biến đó có thể nhận. Một quan hệ có thể được biểu diễn dưới dạng một tập hợp rõ ràng của tất cả các bộ giá trị thỏa mãn ràng buộc, hoặc dưới dạng một hàm mà có thể tính toán xem một bộ có phải là thành viên của mối quan hệ hay không. Ví dụ: nếu X1 và X2 cả hai đều có miền {1,2,3} , thì ràng buộc nói rằng X1 phải lớn hơn X2 có thể được viết là ((X1,X2),{(3,1),(3,2),(2,1)}) hoặc như ((X1,X2),X1>X2).

**1.2.** **Giải pháp cho bài toán thỏa mãn ràng buộc**

Các bài toán thỏa mãn ràng buộc sẽ được giải quyết thông qua các phép gán (assignment) các giá trị vào các biến {Xi=vi, … , Xn = vn }. Một phép gán không vi phạm bất kỳ ràng buộc nào được gọi là phép gán nhất quán hoặc hợp lệ. Phép gán hoàn chỉnh là phép gán trong đó mỗi biến được gán một giá trị.Phép gán một phần là phép gán mà một số biến không được gán. Giải pháp cho một bài toán thỏa mãn ràng buộc là một phép gán hoàn chỉnh và hợp lệ. Bên cạnh đó, giải pháp một phần cho bài toán là một phép gán một phần và hợp lệ

2

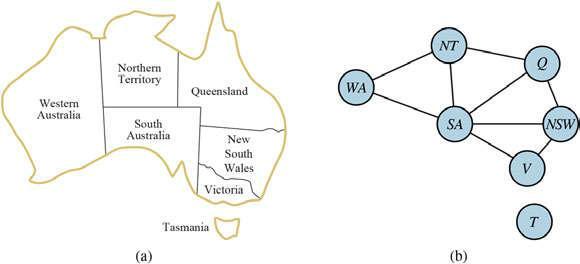
**CHƯƠNG 2: MỘT SỐ VÍ DỤ VỀ CÁC BÀI TOÁN THỎA MÃN RÀNG BUỘC**

**2.1. Bài toán tô màu đồ thị (Map Coloring)**

**2.1.1. Giới thiệu sơ lược**

Map coloring là một ví dụ điển hình về CSP. Bài toán map coloring được chỉ định bởi một tập hợp các màu và bản đồ hiển thị ranh giới giữa các vùng riêng biệt. Một giải pháp cho bài toán map coloring là việc gán một màu cho từng vùng của bản đồ sao cho không có bất kỳ vùng liền kề nào có hai màu giống nhau.

Ban đầu bài toán được biểu diễn bằng hình thực tế của bản đồ (Hình (a)), nhưng để có thể giải quyết bài toán một cách đơn giản và dễ dàng thì chúng ta sẽ biểu diễn chúng bằng đồ thị ràng buộc (constraint graph) (Hình (b)). Trong đó đồ thị ràng buộc (constraint graph) là một đồ thị có các nút biểu diễn các biến và các cạnh biểu diễn các ràng buộc giữa các biến.



**2.1.2. Định nghĩa bài toán dưới dạng CSP**

Bài toán map coloring có thể được biểu diễn dưới dạng CSP như sau:

* **Variables:** Mỗi biến trong CSP dùng để đại diện cho từng vùng khác nhau

trên bản đồ. Ở đây để đơn giản hơn, chúng ta sử dụng các tên được viết tắt cho các vùng : *WA=Western Australia, SA=Southern Australia, NT=Northern*

3

 *Territory, Q=Queensland, NSW=New South Wales, V=Victoria, T=Tasmania*. Từ đó ta có X={WA,NT,Q,NSW,V,SA,T}

* **Domains:** Miền giá trị của mỗi biến là Di = {red,green,blue} với i=1,2,…,7.
* **Constraints:** Các khu vực liền kề phải được tô màu khác nhau. Từ đó ta có 9 ràng buộc như sau C = {SA ≠ WA, SA ≠ NT, SA ≠ Q, SA ≠ NSW, SA ≠ V ,

WA ≠ NT, NT ≠ Q, Q ≠ NSW, NSW ≠ V }. Trong đó, SA≠WA là viết tắt của <(SA,WA),SA≠WA>, ta có thể liệt kê các cặp (SA, WA) lần lượt như sau {(red,green), (red,blue), (green,red), (green,blue), (blue,red), (blue,green)}.

**2.1.3. Lời giải cho bài toán**

Sau khi thực hiện gán giá trị cho các biến thì ta sẽ có được nhiều lời giải cho bài toán trên và {WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green} là một trong các lời giải đó.

Theo bài toán ta thấy biến SA có 5 ràng buộc; NT, Q, NSW có 3 ràng buộc; WA, V có 2 ràng buộc và T không có ràng buộc nào. Giả sử ta chọn màu lần lượt cho SA, NT, Q, NSW, WA, V, T với 3 màu red, blue, green thì ta sẽ có được số cách chọn màu lần lượt như sau 3,2,1,1,1,3. Từ đó suy ra bài toán này sẽ có tới 3 x 2 x 1 x 1 x 1 x 3 = 18 lời giải thỏa mãn.

**2.1.4. Lý do nên xây dựng một vấn đề dưới dạng CSP**

Việc sử dụng CSPs có thể giúp đơn giản hóa và tự nhiên hóa việc mô hình hóa các vấn đề phức tạp. Ngoài ra, CSPs còn giúp giảm thiểu thời gian và tài nguyên cần thiết được sử dụng để giải quyết các vấn đề, làm cho quá trình giải quyết các vấn đề này trở nên nhanh chóng và hiệu quả hơn.

Ví dụ trong bài toán map coloring ở trên, khi ta chọn {SA=blue}, ta có thể suy ra một ràng buộc rằng không có một trong năm biến lân cận nào có thể nhận giá trị blue. Nếu không có ràng buộc thì ta cần phải xem xét đến 35 = 243 trường hợp cho năm biến lân cận. Nhưng khi sử dụng ràng buộc thì chỉ cần 25 = 32 trường hợp cần được xem xét, điều này sẽ giảm tới gần 87% công việc của bài toán.

4

**2.2. Lập lịch sản xuất (Job-Shop Scheduling)**

**2.2.1. Giới thiệu sơ lược**

Job-Shop Scheduling là một vấn đề tối ưu hóa liên quan đến việc xác định thứ tự của một nhóm công việc sẽ được xử lý bởi một nhóm máy móc. Mỗi công việc bao gồm một loạt các nhiệm vụ cần được hoàn thành theo một thứ tự cụ thể. Đây có thể là một vấn đề phức tạp, vì mỗi công việc có thể có nhiều chuỗi tác vụ khác nhau và mỗi chuỗi có thể có thời gian xử lý khác nhau và ảnh hưởng đến thời gian xử lý của các công việc khác. Mục tiêu hướng đến ở đây là giảm thiểu khoảng thời gian cần thiết để hoàn thành tất cả các công việc.

**2.2.2. Định nghĩa bài toán dưới dạng CSP**

Xét bài toán lập lịch trình lắp ráp ô tô được biểu diễn dưới dạng CSP như sau:

 **Variables**: Chúng ta xem xét một phần nhỏ của việc lắp ráp ô tô bao gồm 15

nhiệm vụ: install axles(front and back) (lắp trục trước và sau), affix all four wheels (right and left, front and back) (gắn cả bốn bánh phải trái và trước sau) , tighten nuts for each wheel (siết chặt đai ốc cho từng bánh), affix

hubcaps (gắn nắp trục), and inspect the final assembly (kiểm tra lần cuối). Ta có thể biểu diễn nhiệm vụ với 15 biến: X={AxleF, AxleB, WheelRF, WheelLF, WheelRB, WheelLB, NutsRF, NutsLF, NutsRB, NutsLB, CapRF, CapLF, CapRB, CapLB, Inspect}.

 **Domains**: giả sử có yêu cầu hoàn thành toàn bộ công việc lắp ráp trong

30 phút. Chúng ta có thể đạt được điều đó bằng cách giới hạn miền của tất cả các biến: Di = {0,1,2,3,…,30}.

* **Constraints:** Chúng ta sử dụng các ràng buộc ưu tiên (Precedence constraint)

để đảm bảo thứ tự thực hiện giữa các tác vụ riêng lẻ. Khi một tác vụ T1 phải xảy ra trước tác vụ T2 và tác vụ T1 mất khoảng thời gian d1 để hoàn thành, ta sẽ thêm một ràng buộc có dạng: T1 + d1 <= T2.

Các trục phải được đặt đúng vị trí trước khi lắp bánh xe và phải mất 10 phút để lắp một trục:

AxleF + 10 <= WheelRF; AxleF + 10 <= WheelLF AxleB + 10 <= WheelRB; AxleB + 10 <= WheelLB

5

Đối với mỗi bánh xe chúng ta phải gắn bánh xe mất 1 phút, sau đó siết chặt đai ốc (2 phút) và cuối cùng là gắn nắp trục:

WheelRF + 1 <= NutsRF; NutsRF + 2 <= CapRF;

WheelLF + 1 <= NutsLF; NutsLF + 2 <= CapLF;

WheelRB + 1 <= NutsRB; NutsRB + 2 <= CapRB;

WheelLB + 1 <= NutsLB; NutsLB + 2 <= CapLB;

Giả sử Chúng ta có bốn công nhân để lắp bánh xe, nhưng họ phải dùng chung một công cụ giúp đặt trục vào vị trí. Chúng sử dụng ràng buộc không gian con (Disjunctive constraint) để đảm bảo rằng rằng AxleF và AxleB không được trùng nhau về thời gian, một trong hai sẽ được thực hiện trước:

(AxleF + 10 <= AxleB) or (AxleB + 10 <= AxleF)

Việc kiểm tra diễn ra sau cùng và mất 3 phút. Vì vậy đối với mọi biến ngoại trừ Inspect, chúng ta thêm một ràng buộc có dạng :

X + dX <= Inspect.

**2.2.3. Lời giải cho bài toán**

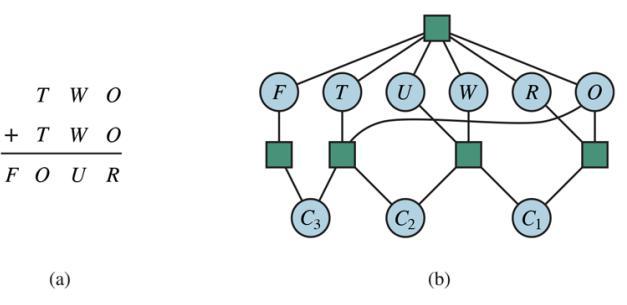
Sau khi thực hiện gán giá trị cho các biến thì ta sẽ thu được nhiều lời giải cho bài toán trong đó { AxleF = 1, AxleB = 11, WheelRF = 11, WheelLF = 11, WheelRB = 21, WheelLB = 21, NutsRF = 22, CapRF = 24, NutsLF = 22, CapLF = 24, NutsRB = 22, CapRB = 24, NutsLB = 22, CapLB = 24, Inspect= 27 } là một trong các lời giải đó.

**2.3 Bài toán giải mã (cryptarithmetic puzzles)**

**2.3.1 Giới thiệu sơ lược**

Câu đố mật mã (Hình (a)) là một trò chơi logic về toán học, trong đó mỗi chữ cái là viết tắt của một chữ số phân biệt, ta phải tìm các chữ số thay thế cho các chữ cái sao cho phép tính là đúng về mặt số học, và chữ cái đứng đầu không được phép thay thế bằng chữ số 0.

6



**2.3.2. Định nghĩa bài toán dưới dạng CSP**

* **Variables**: X = {F, T, U, W, R, O, C1, C2, C3}. Trong đó, C1, C2, C3 là các

biến phụ dùng để biểu diễn các số nhớ được chuyển sang các cột hàng chục, trăm hoặc ngàn.

* **Domains**: DU = DW = DR =DO=DC1=DC2=DC3={0,1,2,3,4,5,6,7,8,9}. DF = DT ={1,2,3,4,5,6,7,8,9}. F và T là các chữ cái đứng đầu và không được phép nhận giá trị 0
* **Constraints**: Tập ràng buộc sẽ bao gồm các ràng buộc sau:
  + Giá trị của các biến F, T, U, W, R, O phải khác nhau từng dôi một
  + O+O=R+10.C1
  + C1+W +W=U+10.C2
  + C2+T+T=O+10.C3
  + C3=F

Ngoài ra chúng ta còn có sử dụng ràng buộc siêu đồ thị (Constraint hypergraph) (Hình (b)) để biểu diễn các ràng buộc. Trong đó các node hình tròn (ordinary node) đại diện cho các biến, node hình vuông (hypernode) ở đỉnh thể hiện ràng buộc giá trị của các biến phải khác nhau từng đôi một và bốn node hình vuông ở giữa thể hiện các ràng buộc được bổ sung trên bốn cột để đảm bảo kết quả của phép tính là đúng.

**2.3.3. Lời giải cho bài toán**

Sau khi thực hiện phép gán thì ta sẽ thu được nhiều lời giải cho bài toán và{T=7, W=3, O=4, F=1, U=6, R=8} là một trong số các lời giải đó.

7

**CHƯƠNG 3: CÁC BIẾN THỂ CỦA BÀI TOÁN THỎA MÃN RÀNG BUỘC**

Các bài toán thỏa mãn ràng buộc có nhiều biến thể khác nhau và có thể được phân loại theo miền giá trị của các biến hoặc theo các loại ràng buộc. Ngoài ra ta cũng có thể chuyển đổi một bài toán có ràng buộc bất kì thành một bài toán có ràng buộc nhị phân.

**3.1. Miền rời rạc và miền liên tục**

Bài toán thỏa mãn ràng buộc có giá trị của các biến có thể thuộc miền rời rạc hoặc miền liên tục.

* **Miền rời rạc:** là tập hợp các giá trị đầu vào chỉ bao gồm các số cụ thể trong một khoảng. Miền rời rạc là một miền hữu hạn hoặc có thể là vô cùng.
* **Miền hữu hạn:** là tập hợp các giá trị đầu vào bao gồm một số lượng giá trị hữu hạn, tức là chỉ bao gồm một số hữu hạn các giá trị riêng lẻ.

Ví dụ: các bài toán tô màu đồ thị, lập thời gian biểu có giới hạn thời gian hay bài toán 8 quân hậu đều là các vấn đề ràng buộc có giá trị của các biến thuộc miền này.

* **Miền vô cùng:** là tập hợp các giá trị đầu vào mở rộng trên một phạm vi giá trị

không có giới hạn trên hoặc dưới .

Ví dụ: Bài toán tìm các giá trị nguyên x,y sao cho 2x + y = 3. Ở đây miền giá trị hai biến x, y là tập hợp các số nguyên cũng là miền rời rạc gồm vô hạn các giá trị.

* **Miền liên tục:** là một tập hợp các giá trị đầu vào bao gồm tất cả các số trong một khoảng. Dạng bài toán nổi tiếng nhất của CSP miền liên tục là các bài toán quy hoạch tuyến tính, trong đó các ràng buộc phải là đẳng thức hoặc bất

đẳng thức tuyến tính.

Ví dụ: Tìm các giá trị thực của x, y, z thuộc [0, 1], sao cho x + y + z <= 3

**3.2. Các loại ràng buộc của bài toán thỏa mãn ràng buộc**

**3.2.1. Ràng buộc tuyến tính và ràng buộc phi tuyến**

* **Ràng buộc tuyến tính:** là một dạng ràng buộc mà có thể biểu diễn bởi một hoặc nhiều bất phương trình tuyến tính với các biến và hệ số tuyến tính, chỉ bao gồm các biến số bậc nhất.

8

Ví dụ: x + y ≤ 30

* **Ràng buộc phi tuyến tính :** khác với ràng buộc tuyến tính, ràng buộc phi tuyến bao gồm các mối quan hệ trong đó các biến được bình phương, lập phương, lên một số mũ khác ngoài một, hoặc nhân hoặc chia cho nhau.

Ví dụ: x 2+ y2 ≤ 30, x 2 - y ≥ 8

**3.2.2. Ràng buộc đơn (unary constraint), ràng buộc nhị phân (binary constraint) và ràng buộc bậc cao (higher-order constraints)**

* **Ràng buộc đơn:** là một loại ràng buộc liên quan đến một biến duy nhất. Nó chỉ định một hạn chế đối với các giá trị có thể có mà biến có thể nhận mà

không cần tham chiếu đến bất kỳ biến nào khác.

Ví dụ, trong bài toán tô màu bản đồ, có thể xảy ra trường hợp South Australians sẽ không chấp nhận màu xanh lá cây, chúng ta có thể diễn đạt điều đó bằng ràng buộc đơn <(SA),SA ≠green>.

* **Ràng buộc nhị phân:** là một loại ràng buộc liên quan đến hai biến . Nó chỉ định một hạn chế đối với các giá trị có thể được gán đồng thời cho một cặp biến.

Ví dụ <(SA,NSW),SA ≠NSW> là một ràng buộc nhị phân.

* **Ràng buộc bậc cao**: là các loại ràng buộc liên quan đến ba biến hoặc nhiều

hơn.

Ví dụ: ((X,Y,Z),X <Y < Z)

**3.2.3. Ràng buộc toàn cục (global constraint)**

Ràng buộc toàn cục là một loại ràng buộc được áp dụng đồng thời cho nhiều biến tùy ý. Ràng buộc toàn cục phổ biến nhất là ràng buộc Alldiff . Trong ràng buộc Alldiff thì tất cả các biến liên quan phải có giá trị khác nhau.

Ví dụ: Trong trò chơi Sudoku thì ta phải điền các số từ 1-9 sao cho mỗi số trên hàng dọc, hàng ngang hoặc khối 3x3 không được trùng nhau.

9

**3.2.4. Ràng buộc ưu tiên (preference constraint)**

Ràng buộc ưu tiên là ràng buộc sử dụng để biểu diễn sự ưu tiên giữa các giá trị của các biến**.** Điều này cho phép người dùng chỉ định các giá trị được ưa thích hơn so với các giá trị khác trong các ràng buộc.

Ví dụ: Khi lập lịch phân công giảng dạy thì giảng viên A có thể dạy buổi sáng hoặc buổi chiều vào thứ 7 trong tuần. Giảng viên A thích dạy vào buổi sáng hơn buổi chiều nên ta có thể dùng ràng buộc ưu tiên cho trường hợp này.

Ràng buộc ưu tiên thường được mã hóa dưới dạng chi phí cho các phân bổ biến riêng lẻ. Trong ví dụ trên, phân công cho giảng viên A dạy vào buổi chiều sẽ có chi phí là 2 điểm cho hàm mục tiêu, trong khi phân công vào buổi sáng sẽ có chi phí là 1 điểm.

**3.3. Chuyển đổi sang ràng buộc nhị phân**

Để chuyển đổi một bài toán có ràng buộc nhiều biến sang ràng buộc nhị phân thì ta sẽ đưa về một bài toán thỏa ràng buộc mới có các biến sao cho mỗi một biến ứng với mỗi ràng buộc của bài toán ban đầu và một ràng buộc nhị phân ứng với mỗi cặp ràng buộc trong bài toán ban đầu mà có chia sẻ các biến chung.

Ví dụ: Bài toán thỏa ràng buộc có các biến X = {x, y, z}, mỗi biến có miền giá trị là {1,2,3,4,5} và hai Ràng buộc C1: ((x, y, z), x + y = z) và C2: ((x, y), x + 1 = y). Sau khi chuyển đổi, bài toán mới có các biến X = {C1, C2}. Miền giá trị của biến C1 là tập các bộ ba{(xi, yj, zk)} từ ràng buộc C1 ban đầu, miền giá trị của C2 là tập các bộ hai {(xi, yj)} từ ràng buộc C2 ban đầu. Ràng buộc nhị phân ((C1, C2), R1), trong đó R1 = {((1,2,3),(1,2)) , ((2,3,5), (2,3))} là một quan hệ mới xác định ràng buộc giữa C1 và C2.

Tuy nhiên, chúng ta có thể ưu tiên sử dụng ràng buộc toàn cục hơn là một tập các ràng buộc nhị phân vì hai lý do. Đầu tiên, việc mô tả vấn đề bằng ràng buộc toàn cục thì dễ dàng hơn và ít mắc lỗi hơn. Thứ hai, các ràng buộc toàn cục

10

thường chứa nhiều thông tin hơn về các giá trị hợp lệ của các biến và các bộ giá trị của chúng. Các thuật toán suy luận đặc biệt có thể sử dụng thông tin này để loại bỏ các giá trị không hợp lệ và giảm thiểu không gian tìm kiếm, từ đó cải thiện hiệu suất của thuật toán.

11

1. **PHẦN KẾT LUẬN**

Qua việc phân tích các nội dung trong báo cáo, chúng ta đã hiểu rõ hơn về định nghĩa Constraint Satisfaction Problem (CSP). Chúng ta đã điểm qua các ví dụ thực tế về CSP như Map coloring, Job-shop scheduling, cryptarithmetic puzzles cũng như tìm hiểu một số biến thể về hình thức của CSP để giải quyết những vấn đề phức tạp hơn.

Bằng cách áp dụng CSP, chúng ta có thể tìm ra các giải pháp tối ưu cho các vấn đề thực tế khác nhau như lập lịch kinh doanh, điều phối vận tải, lập kế hoạch phân bổ tài nguyên,… Đối với các bài toán lớn và phức tạp, việc áp dụng CSP có thể giúp giải quyết bài toán một cách nhanh chóng và chính xác hơn.

Tóm lại CSP là một bài toán quan trọng trong lĩnh vực trí tuệ nhân tạo. CSP được ứng dụng rộng rãi trong các lĩnh vực khác nhau và đã trở thành một công cụ quan trọng trong việc giải quyết các vấn đề có ràng buộc nhằm tối ưu hóa giải pháp và giảm thiểu chi phí.

12

**D. TÀI LIỆU THAM KHẢO**

1/ Stuart Russell, Artificial Intelligence: A Modern Approach, 4th edition ( Sách bản PDF )

2/ <http://aima.cs.berkeley.edu/newchap05>

3/ [https://courses.cs.washington.edu/courses/csep573/11wi/lectures/07-csp.pdf](https://courses.cs.washington.edu/courses/csep573/11wi/lectures/07-csp)

[4/https://www.youtube.com/watch?v=nKDaifJjGsg&list=PL13VD7ErsVSDDAvW0PO](https://www.youtube.com/watch?v=nKDaifJjGsg&list=PL13VD7ErsVSDDAvW0POwOIfUvR4jAyZVR&ab_channel=SACHLETHANH) wOIfUvR4jAyZVR&ab\_channel=SACHLETHANH

13