

Constraint Satisfaction Problems

305450: Lecture 4

7/15/2009

305450 - Constraint Satisfaction

1

Constraint satisfaction problems (CSPs)

- CSP is defined by:
 - a set of **variables**: X_1, X_2, \dots, X_n
 - Each variable X_i has a nonempty **domain** D_i of possible **values**
 - a set of **constraints**: C_1, C_2, \dots, C_m
 - Each constraint involves some subset of the variables and specifies the allowable combinations of values for that subset
 - A state of the problem is defined by an **assignment** of values to some or all of the variables
 - e.g., $\{X_i = v_i, X_j = v_j, \dots\}$

7/15/2009

305450 - Constraint Satisfaction

2

-Constraint Satisfaction Problem นิยามโดย

- เซ็ตของตัวแปร โดยแต่ละตัวแปรจะมีค่าอยู่ในโดเมนๆหนึ่ง
- เซ็ตของเงื่อนไข ซึ่งแต่ละเงื่อนไขอาจเกี่ยวข้องกับตัวแปร หนึ่งตัว หรือมากกว่าหนึ่งตัวก็ได้
- State** ของปัญหานิยามด้วยการให้ค่ากับตัวแปรบางตัวหรือทั้งหมด

Constraint satisfaction problems (CSPs)

- An assignment that does not violate any constraints is called a **consistent** or legal assignment
- A complete assignment is one in which every variable is mentioned
- A **solution** to a CSP is a complete assignment that satisfies all the constraints

7/15/2009

305450 - Constraint Satisfaction

3

- การให้ค่าต้องไม่ขัดแย้งกับเงื่อนไขที่กำหนด
- การให้ค่าที่สมบูรณ์คือการที่ตัวแปรทุกตัวถูกให้ค่าหมดแล้ว
- คำตอบสำหรับปัญหา CSP นั้นจะต้องเป็นการให้ค่าที่สมบูรณ์และไม่ขัดแย้งกับเงื่อนไขที่กำหนด

Example CSPs (Map Coloring)



- Variables WA, NT, Q, NSW, V, SA, T
- Domains $D_i = \{\text{red, green, blue}\}$
- Constraints: adjacent regions must have different colors
- e.g., $WA \neq NT$, or (WA, NT) in $\{(\text{red, green}), (\text{red, blue}), (\text{green, red}), (\text{green, blue}), (\text{blue, red}), (\text{blue, green})\}$

7/15/2009

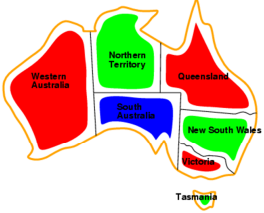
305450 - Constraint Satisfaction

4

So what all these mean?

- ตัวอย่างนี้แสดงปัญหาการใส่สีให้กับ state ของประเทศออสเตรเลีย
- ตัวแปรของปัญหาจะเป็น states ของประเทศออสเตรเลีย
- ส่วนโดเมนจะเป็นสีสามสีที่ให้มา
- เงื่อนไขคือว่า state ที่อยู่ติดกันจะต้องไม่มีสีเดียวกัน

Example: Map-Coloring



- Solutions are **complete** and **consistent** assignments, e.g., WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green

7/15/2009 305450 - Constraint Satisfaction 5

-คำตอบสำหรับปัญหานี้ต้องเป็นการให้ค่าที่สมบูรณ์ (คือตัวแปรทุกตัวถูกให้ค่า) และ **consistent** (ไม่ขัดแย้งกับเงื่อนไขที่กำหนด)

Varieties of constraints

- Unary** constraints involve a single variable,
 - e.g., SA \neq green
- Binary** constraints involve pairs of variables,
 - e.g., SA \neq WA
- Higher-order** constraints involve 3 or more variables,
 - can be reduced to binary CSP

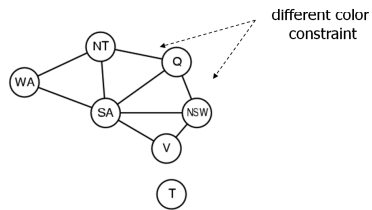
7/15/2009 305450 - Constraint Satisfaction 6

-เงื่อนไขมีได้หลายแบบ

- เกี่ยวข้องกับตัวแปรเดียว เช่น **southern australia** ต้องไม่เป็นสีเขียว
- เกี่ยวข้องกับตัวแปรสองตัว เช่น สีของ **southern australia** กับ **western australia** จะต้องไม่เป็นสีเขียวกัน
- ส่วนเงื่อนไขที่เกี่ยวกับตัวแปรตั้งแต่สามตัวขึ้นไปจะสามารถลดลงมาอยู่ในรูปของเงื่อนไขที่เกี่ยวกับสองตัวแปรได้
- ดังนั้นในบทนี้เราพิจารณาเฉพาะ **unary and binary constraints**

Constraint graph

- **Binary CSP:** each constraint relates two variables
- **Constraint graph:** nodes are variables, arcs are constraints



7/15/2009

305450 - Constraint Satisfaction

7

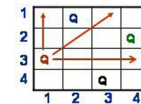
-Constraint graph ประกอบด้วยโหนดที่แทนตัวแปรในปัญหาที่เรากำลังพิจารณา

- ส่วน edge หรือเรียกว่า arc นั้นจะแสดงเงื่อนไขระหว่างตัวแปรในปัญหา

- เช่นในตัวอย่างของเราในดจะแสดง state ของประเทศ australia ส่วน arc จะแสดงเงื่อนไขว่าสีของแต่ละ state จะต้องต่างกัน

Example CSPs

Place N queens on a NxN chessboard so that none can attack the other



- Variables: board positions in NxN chessboard
- Domains: Queen or Blank
- Constraints: Two positions on a line (vertical, horizontal, diagonal) cannot both be Q

7/15/2009

305450 - Constraint Satisfaction

8

-ตัวอย่างอีกอันของ CSP คือ N Queens Problem ซึ่งจะต้องวาง queens จำนวน n ตัวใน board ที่มีขนาด nxn โดยที่ไม่ให้ queens นั้นสามารถ attack กันได้

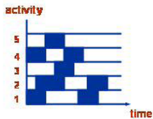
- ตัวแปรในปัญหานี้คือ ตำแหน่งใน board

- โดเมนคือว่า ตำแหน่งนั้นว่างหรือมี queen อยู่

- เงื่อนไขคือว่าสองตำแหน่งที่อยู่บนเส้นแนวตั้ง แนวทแยง หรือทแยงจะต้องไม่เป็น queens ทั้งคู่

Example CSPs

Choose time for activities, e.g.,
pick up products from providers,
deliver products to customers,
bill customers for the ordered products



- Variables: activities
- Domains: set of start times
- Constraints: activities that use same resources cannot overlap in time

7/15/2009 305450 - Constraint Satisfaction 9

- อีกตัวอย่างคือการจัดเวลาให้กับการทำกิจกรรม เช่น ไปเอาสินค้าจากผู้ผลิต ส่งสินค้าให้ลูกค้า เก็บเงินลูกค้า
- ตัวแปรคือ กิจกรรมต่างๆ
- โดเมนคือเวลาเริ่มต้นในการทำแต่ละกิจกรรม
- เงื่อนไขคือว่ากิจกรรมที่ใช้ **resources** เดียวกันจะมีความเหลื่อมล้ำในเรื่องของเวลาไม่ได้

CSPs formulation

States are defined by the values assigned so far

- Initial state: the empty assignment $\{ \}$
- Successor function: assign a value to an unassigned variable that does not conflict with current assignment
→ fail if no legal assignments
- Goal test: the current assignment is complete

7/15/2009 305450 - Constraint Satisfaction 10

- การสร้างปัญหา CSP ให้อยู่ในรูปแบบของ search problem
- Initial state จะเริ่มจากการที่ยังไม่ได้ให้ค่ากับตัวแปรใดๆทั้งสิ้น
- Successor function จะเป็นการให้ค่ากับตัวแปรที่ยังไม่ได้ให้ค่า โดยการให้ค่าต้องไม่ขัดแย้งกับเงื่อนไขที่กำหนด — consistency
- goal test คือการดูว่าเราให้ค่ากับตัวแปรทุกตัวครบหรือยัง

Solving CSPs

- Solving CSPs involves some combination of:
 - Constraint Propagation, to eliminate values that could not be part of any solution
 - Search, to explore valid assignments

7/15/2009

305450 - Constraint Satisfaction

11

-การแก้ปัญหา CSP นั้นจะประกอบด้วย

- **constraint propagation** เพื่อที่จะกำจัดค่าที่ไม่สามารถเป็นส่วนหนึ่งของคำตอบได้
- **search** เพื่อพิจารณาการให้ค่าที่เป็นไปได้ทั้งหมด

Constraint Propagation (aka Arc Consistency)

- Arc consistency eliminates values from domain of variable that can never be part of a consistent solution
- $X \rightarrow Y$ is consistent iff
 - for **every** value x of X there is **some** allowed y
- We can achieve consistency on arc by deleting values from D_i that fail this condition

7/15/2009

305450 - Constraint Satisfaction

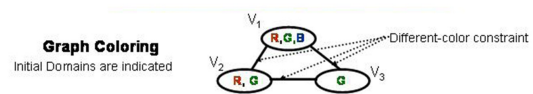
12

-Arc consistency กำจัดค่าที่ไม่สามารถเป็นส่วนหนึ่งของคำตอบได้ (อย่าสับสนกับเรื่อง A* search)

-เงื่อนไข consistency ต้องพิจารณาจากว่าสำหรับค่าทุกค่าใน X ต้องมีอย่างน้อยหนึ่งค่าใน Y ที่ไม่ขัดกับเงื่อนไข เดียวตัวอย่าง

Constraint Propagation

Let's look at a trivial example of graph coloring.
We have three variables with the domains indicated.
Each variable is constrained to have values different from its neighbors.



7/15/2009

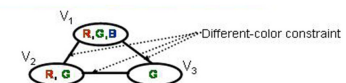
305450 - Constraint Satisfaction

13

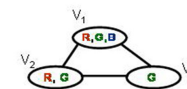
- เช่นตัวอย่างการให้สีกับเมืองสามเมือง
- ค่าโดเมนเริ่มต้นของแต่ละเมืองในปัญหานี้ไม่เท่ากัน
 - เมือง V1 สีที่สามารถให้ได้คือ แดง เขียว ฟ้า
 - เมือง V2 สีที่สามารถให้ได้คือ แดง เขียว
 - เมือง V3 สีที่สามารถให้ได้คือ เขียว

Constraint Propagation

Graph Coloring
Initial Domains are indicated



Arc examined	Value deleted



Each undirected constraint arc is really two directed constraint arcs, the effects shown above are from examining BOTH arcs.

7/15/2009

305450 - Constraint Satisfaction

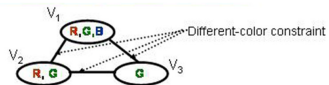
14

- เนื่องจากปัญหานี้อธิบายด้วย undirected graph ดังนั้น arc consistency constraint ต้องพิจารณาสองทิศทาง เช่นจาก V1 to V2 and V2 to V1

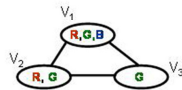
Constraint Propagation

Graph Coloring

Initial Domains are indicated



Arc examined	Value deleted
$V_1 - V_2$	none



7/15/2009

305450 - Constraint Satisfaction

15

-เริ่มจาก arc ระหว่าง V_1 - V_2 เราพบว่า

- ถ้า V_1 เป็นสีแดง V_2 เป็นสีเขียวได้
- ถ้า V_1 เป็นสีเขียว V_2 เป็นสีแดงได้
- ถ้า V_1 เป็นสีฟ้า V_2 เป็นสีแดงหรือเขียวก็ได้

-ในทางกลับกัน

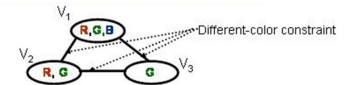
- ถ้า V_2 เป็นสีแดง V_1 เป็นสีเขียวหรือฟ้าก็ได้
- ถ้า V_2 เป็นสีเขียว V_1 เป็นสีแดงหรือฟ้าก็ได้

-ดังนั้น V_1 and V_2 are consistent

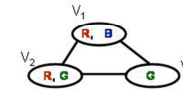
Constraint Propagation

Graph Coloring

Initial Domains are indicated



Arc examined	Value deleted
$V_1 - V_2$	none
$V_1 - V_3$	$V_1(\text{G})$



7/15/2009

305450 - Constraint Satisfaction

16

- ในกรณีนี้เราจะมีปัญหาว่าถ้า V_1 เป็นสีเขียว V_3 จะเป็นสีเขียวไม่ได้ เราเลยกำจัด สีเขียวออกจาก โดเมนของ V_1

Constraint Propagation

Graph Coloring
Initial Domains are indicated

Different-color constraint

Arc examined	Value deleted
$V_1 - V_2$	none
$V_1 - V_3$	$V_1(\mathbf{G})$
$V_2 - V_3$	$V_2(\mathbf{G})$

Note that although we looked at all the arcs once, we need to keep going since we have changed the domains for V_1 and V_2 . It is this phenomenon that accounts for the name "constraint propagation"

7/15/2009 305450 - Constraint Satisfaction 17

- เช่นเดียวกันถ้า V_2 เป็นสีเดียวกับ V_3 จะเป็นสีเดียวกันไม่ได้ เราเลยกำจัด สีเขียวออกจาก โดเมนของ V_2
- แม้ว่าเราจะพิจารณาทุก arc แล้ว แต่เนื่องจากเราเปลี่ยนแปลงค่าโดเมนของแต่ละตัวแปร ดังนั้นเราต้องเช็คต่อไปเรื่อยๆจนกว่าจะไม่มี ความขัดแย้งเหลืออยู่
- ลักษณะการกระจายของ constraint checking เช่นนี้ทำให้เราเรียกสิ่งนี้ว่า constraint propagation

Constraint Propagation

Graph Coloring
Initial Domains are indicated

Different-color constraint

Arc examined	Value deleted
$V_1 - V_2$	none
$V_1 - V_3$	$V_1(\mathbf{G})$
$V_2 - V_3$	$V_2(\mathbf{G})$
$V_1 - V_2$	$V_1(\mathbf{R})$

7/15/2009 305450 - Constraint Satisfaction 18

- ในกรณีนี้เราจะมีปัญหาว่าถ้า V_1 เป็นสีแดง V_2 จะเป็นสีแดงไม่ได้ เราเลยกำจัด สีแดงออกจาก โดเมนของ V_1

Constraint Propagation

Graph Coloring
Initial Domains are indicated

Arc examined	Value deleted
$V_1 - V_2$	none
$V_1 - V_3$	$V_1(\mathbf{G})$
$V_2 - V_3$	$V_2(\mathbf{G})$
$V_1 - V_2$	$V_1(\mathbf{R})$
$V_1 - V_3$	none

7/15/2009 305450 - Constraint Satisfaction 19

- ไม่มีความขัดแย้งหลงเหลือ

Search

- Constraint propagation is not enough
- In general, if there is more than one value in the domain of any of the variables, we do not know whether there is zero, one, or more than one answer that is globally consistent. We have to search for an answer to actually know for sure.

7/15/2009 305450 - Constraint Satisfaction 20

- เราพบว่า **constraint propagation** เพียงอย่างเดียวไม่เพียงพอสำหรับการหาคำตอบให้กับ CSP

- เนื่องจากว่าคำตอบอาจไม่มี มีหนึ่งค่า หรือไม่มีเลยก็ได้ ดังนั้นเพื่อช่วยในการพิจารณาหาคำตอบของ CSP เราใช้เรื่องของ **search** เข้ามาช่วย

Backtracking search

- To search for solutions to a CSP problem, any of the search methods we have studied is applicable.
- All we need to realize is that the space of assignments of values to variables can be viewed as a tree in which all the assignments of values to the first variable are descendants of the first node and all the assignments of values to the second variable form the descendants of those nodes and so forth
- The classic approach to search such a tree is backtracking (i.e., depth-first search)
- Variable assignments are **commutative**, i.e.,
[WA = red then NT = green] same as [NT = green then WA = red]

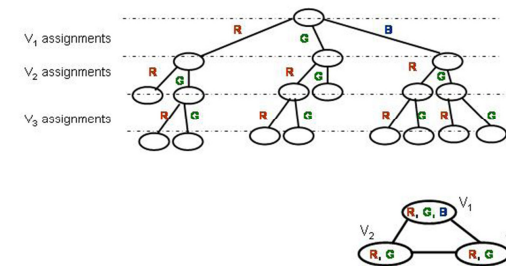
7/15/2009

305450 - Constraint Satisfaction

21

- เราสามารถใช้ **search algorithms** ใดๆที่เราเรียนไปได้
- แต่ถ้าเป็น **search** ที่มี **cost** มาเกี่ยวข้องเราต้องพิจารณาเรื่องของ **preference** แต่ในบทนี้เราสนใจแค่ CSP โดยทั่วไป ดังนั้นเราใช้ **DFS** หรือ **BFS**
- ตัวที่ใช้มากที่สุดคือ **DFS** เพราะถ้าเป็น **BFS** นั้นจะทำให้ต้องพิจารณาจำนวน **branch** มาก
- **Note** ว่าลำดับการให้ค่ากับตัวแปรไม่มีความสำคัญในการหาคำตอบให้กับ CSP

Backtracking example

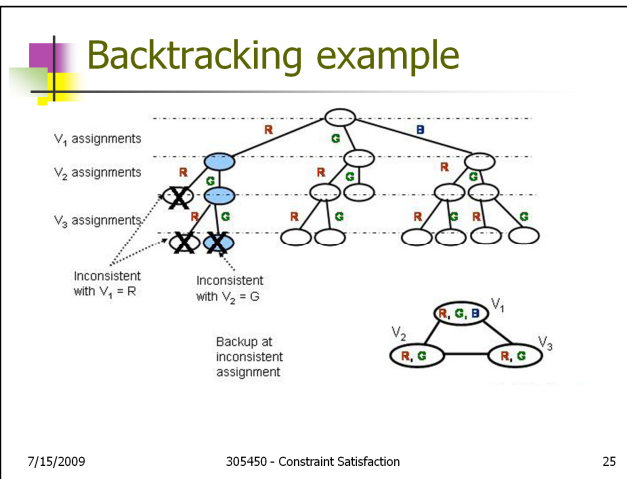


7/15/2009

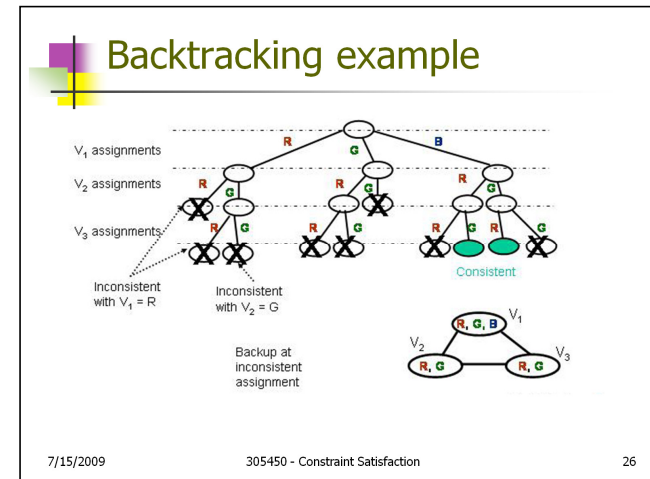
305450 - Constraint Satisfaction

22

- การสร้าง **search tree** สำหรับ CSP problem นั้น เราเริ่มจาก **root node** แล้ว **level 1** เป็นการให้ค่ากับตัวแปรตัวที่ 1 และ **level 2** เป็นการให้ค่ากับตัวแปรตัวที่สอง เป็นอย่างนี้ไปเรื่อยๆ



- แต่ $V_3=G$ is inconsistent with $V_2=G$, ดังนั้นเราย้อนกลับ, แต่ไม่มีค่าอื่นๆที่ต้องพิจารณาสำหรับ V_3 และ V_2 ดังนั้นเราย้อนกลับไป level ของ V_1



- สุดท้ายเราได้ 2 paths ที่เป็นคำตอบให้กับปัญหานี้ได้

Constraint Propagation & BT

- We can use some form of backtracking search to solve CSP independent of any form of constraint propagation.
- However, it is natural to consider combining them.
 - For example, during a BT search where we have a partial assignment, where a subset of all the variables each has unique values assigned, we could then propagate these assignments throughout the constraint graph to obtain reduced domains for the remaining variables.
 - This is, in general, advantageous since it decreases the effective branching factor of the search tree.

7/15/2009

305450 - Constraint Satisfaction

27

- เราสามารถใช้ **backtracking search** เดี่ยวๆ โดยไม่ใช้ **constraint propagation** ก็ได้
- แต่โดยปกติของสิ่งนี้มักจะใช้ด้วยกัน เพราะจะช่วยลดจำนวน **branching factor** ใน **search tree**

Constraint Propagation & BT

- But, how much propagation should we do? Is it worth doing the full arc-consistency propagation we described earlier?
- The answer is USUALLY no.
 - It is generally sufficient to only propagate to the immediate neighbors of variables that have unique values (the ones assigned earlier in the search).
 - That is, we eliminate from consideration any values for future variables that are inconsistent with the values assigned to past variables.
 - This process is known as **forward checking** (FC) because one checks values for future variables (forward in time), as opposed to standard backtracking which checks value of past variables (backwards in time, hence back-checking).

7/15/2009

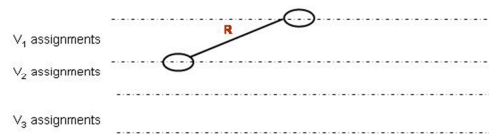
305450 - Constraint Satisfaction

28

- แต่ปัญหาคือถ้าเราต้องการ **propagate** ไปไกลแค่ไหน ต้องทำทุกๆ **arc** เลยหรือไม่
- ซึ่งโดยทั่วไปมันไม่จำเป็น เราสามารถทำการ **propagate** ค่า **constraints** ไปที่โหนดที่อยู่ติดกับมันโดยตรงเลยก็เพียงพอ (**immediate neighbors** ซึ่งคือมี **arc** ต่อถึงกันโดยตรง)
- วิธีการนี้เป็นการ **check** ค่า **arc consistency** ล่วงหน้าก่อนที่จะใส่ โหนดเข้าไปใน **tree** ดังนั้นจึงเรียกว่า **forward checking** ซึ่งต่างจาก **backtracking** ที่พิจารณาค่าที่ผ่านมาแล้ว จึงเรียกว่า **back-checking**

Backtracking with Forward Checking

When examining assignment $V_i = d_{ij}$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.



7/15/2009

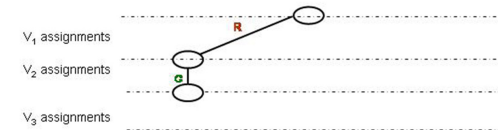
305450 - Constraint Satisfaction

29

- ตัวอย่าง ให้ค่า $V_1 = R$ เราจะต้อง propagate constraint ไปที่ V_2 and V_3
- ดังนั้น V_2 and V_3 ไม่สามารถเป็นสีแดงได้
- ลบ R from domains of V_2 and V_3

Backtracking with Forward Checking

When examining assignment $V_i = d_{ij}$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.



7/15/2009

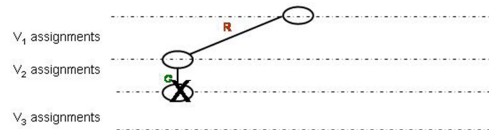
305450 - Constraint Satisfaction

30

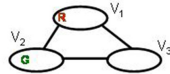
- ให้ค่า $V_2 = G$
- ดังนั้น propagate constraint ไปที่ V_3 ซึ่งทำให้ V_3 เป็นสีเขียวไม่ได้ ดังนั้น โดเมนของ V_3 จะไม่มีค่าที่สามารถใช้ได้อีกต่อไป

Backtracking with Forward Checking

When examining assignment $V_i = d_{ij}$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.



We have a conflict whenever a domain becomes empty.



7/15/2009

305450 - Constraint Satisfaction

31

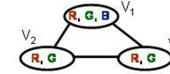
- ดังนั้น path นี้ fail

Backtracking with Forward Checking

When examining assignment $V_i = d_{ij}$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.



When backing up, need to restore domain values, since deletions were done to reach consistency with tentative assignments considered during search.



7/15/2009

305450 - Constraint Satisfaction

32

- ให้พิจารณาย้อนกลับขึ้นมา โดยให้คืนค่า initial domain values ให้กับแต่ละตัวแปร

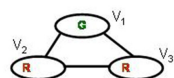
Backtracking with Forward Checking

When examining assignment $V_i = d_{i,j}$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.

V_1 assignments

V_2 assignments

V_3 assignments



7/15/2009

305450 - Constraint Satisfaction

33

- ทำให้ $V1 = G$
- $V2$ and $V3$ cannot be G

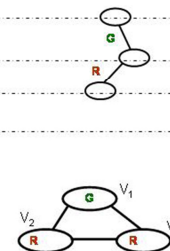
Backtracking with Forward Checking

When examining assignment $V_i = d_{i,j}$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.

V_1 assignments

V_2 assignments

V_3 assignments



7/15/2009

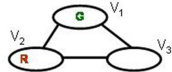
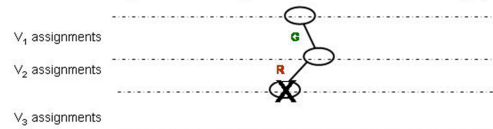
305450 - Constraint Satisfaction

34

- Remove G from domains of $V2$ and $V3$

Backtracking with Forward Checking

When examining assignment $V_i = d_{i,j}$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.



7/15/2009

305450 - Constraint Satisfaction

35

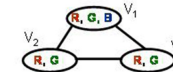
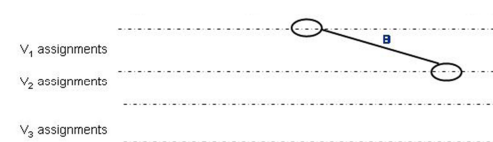
-ให้ $V_2 = R$

-โดเมนของ V_3 จะไม่มีค่าเหลือให้ใช้ได้อีกต่อไป

-ดังนั้น path นี้ fail

Backtracking with Forward Checking

When examining assignment $V_i = d_{i,j}$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.



7/15/2009

305450 - Constraint Satisfaction

36

-ถ้า ให้ $V_1 = B$ จะไม่มีผลกระทบกับ domain ของ V_2 and V_3

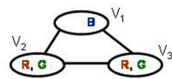
Backtracking with Forward Checking

When examining assignment $V_i = d_{i,j}$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.

V_1 assignments

V_2 assignments

V_3 assignments



7/15/2009

305450 - Constraint Satisfaction

37

- ให้ $V_2 = R$ ดังนั้น V_3 เป็นได้แค่ G

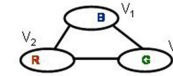
Backtracking with Forward Checking

When examining assignment $V_i = d_{i,j}$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.

V_1 assignments

V_2 assignments

V_3 assignments



7/15/2009

305450 - Constraint Satisfaction

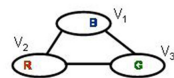
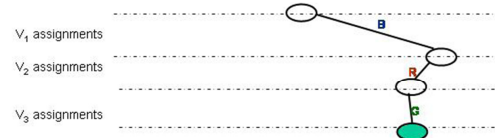
38

- ถ้าให้ V_2 เป็น R ดังนั้น V_3 เป็นได้แค่ G



Backtracking with Forward Checking

When examining assignment $V_i = d_{i,j}$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.



7/15/2009

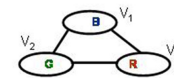
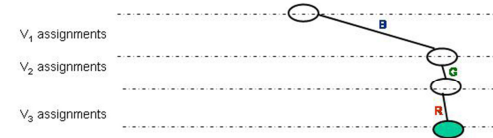
305450 - Constraint Satisfaction

39



Backtracking with Forward Checking

When examining assignment $V_i = d_{i,j}$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.



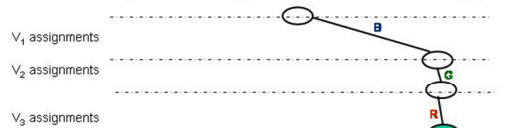
7/15/2009

305450 - Constraint Satisfaction

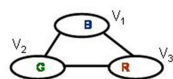
40

Backtracking with Forward Checking

When examining assignment $V_i = d_{ij}$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.



No need to check previous assignments



Generally preferable to pure BT

7/15/2009

305450 - Constraint Satisfaction

41

- เวลาที่เราทำ **forward checking** ไม่จำเป็นต้อง **check new assignments against previous assignments**. เพราะว่าค่าใดๆที่มีแนวโน้มว่าจะไม่ **consistent** จะถูกกำจัดโดยการ **propagation**.

- การใช้ **BT** ร่วมกับ **FC** โดยปกติจะนิยมมากกว่าการใช้ **BT** อย่างเดียว เนื่องจากว่าการพิจารณา **assignment** ที่ไม่ **consistent** จะทำครั้งเดียวและใช้กับทุกตัว แทนที่จะต้องทำการเช็คซ้ำไปซ้ำมาในแต่ละส่วนของ **tree**

- เช่น ในการใช้ **BT** อย่างเดียว การให้ค่ากับ **V3** ที่ไม่ **consistent** กับค่าของ **V1** นั้นจะถูกพิจารณาสำหรับทุกค่าของ **V2** แต่การใช้ ค่าที่ไม่ **consistent** นั้นจะถูกลบออกจากโดเมนของ **V3** ในทันทีซึ่งไม่ต้องมาพิจารณาอีกสำหรับค่าอื่นๆของ **V2**

Variable and value ordering

- **Most constrained variable:**
 - choose the variable with smallest valid domain
 - This will have the effect of reducing the average branching factor in the tree
- **Least constraining value:**
 - choose value that rules out the fewest values from neighboring domains
 - As each value of a variable impact the domains of its neighbors. The value with the largest minimum resulting domain size would be one that least constraints the remaining choices and is least likely to lead to failure
 - However, value ordering is only worth doing if we are looking for a single answer to the problem. Of we want all answers, then all values will have to be tried eventually.

7/15/2009

305450 - Constraint Satisfaction

42

- ที่ผ่านมามาไม่ได้พิจารณาเรื่องลำดับของการให้ค่า หรือการเลือกตัวแปร

- ถ้าเราพิจารณาสิ่งนี้เราจะสามารถช่วยทำให้การหาคำตอบของ **CSP** เร็วขึ้นได้

- การเลือกตัวแปร (**Variable**)

- สำหรับ การให้ค่ากับตัวแปร ให้เราเลือกตัวแปรที่มีเงื่อนไขมากที่สุด นั่นคือเลือกตัวแปรที่มีค่าในโดเมนน้อยที่สุด

- ซึ่งจะลดค่า **branching factor** ของ **tree**

- การให้ค่ากับตัวแปร

- ให้เลือกค่าที่มีผลกระทบกับตัวแปรข้างเคียงน้อยที่สุด

- ซึ่งจะลดโอกาสของการเป็น **failure paths**

- แต่การพิจารณาลำดับของการให้ค่า จะมีผลก็ต่อเมื่อเราต้องการคำตอบเพียงค่าเดียว (ซึ่งคือค่าแรกที่เราพบ) แต่ถ้าเราต้องการคำตอบทั้งหมด ลำดับของการให้ค่า ก็ไม่มีผล

Example

Colors: R, G, B, Y

A=Green
B=Blue
C=Red

Which country should we color next →
What color should we pick for it? →

7/15/2009 305450 - Constraint Satisfaction 43

This example of the 4-color map-coloring problem illustrates a simple situation for variable and value ordering. Here, A is colored Green, B is colored Blue and C is colored Red. What country should we color next, D or E or F?

Example

Colors: R, G, B, Y

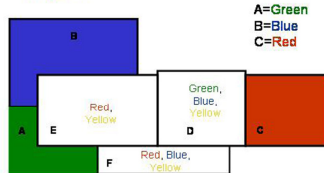
A=Green
B=Blue
C=Red

Which country should we color next → E most-constrained variable (smallest domain)
What color should we pick for it? →

7/15/2009 305450 - Constraint Satisfaction 44

Example

Colors: R, G, B, Y



A=Green
B=Blue
C=Red

Which country should we color next → E most-constrained variable (smallest domain)
What color should we pick for it? → RED least-constraining value (eliminates fewest values from neighboring domains)

7/15/2009

305450 - Constraint Satisfaction

45

Summary

- CSPs are a special kind of problem:
 - states defined by values of a fixed set of variables
 - goal test defined by constraints on variable values
- Backtracking = depth-first search with one variable assigned per node
- Variable ordering and value selection heuristics help significantly
- Forward checking prevents assignments that guarantee later failure

7/15/2009

305450 - Constraint Satisfaction

46