

IS-ZC444: ARTIFICIAL INTELLIGENCE

Lecture-09: Constraint Satisfaction



Dr. Kamlesh Tiwari

Assistant Professor

Department of Computer Science and Information Systems,
BITS Pilani, Pilani, Jhunjhunu-333031, Rajasthan, INDIA

October 03, 2018

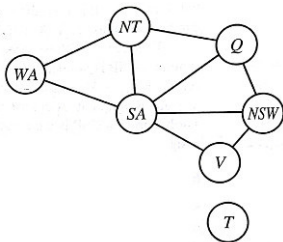
(WILP @ BITS-Pilani Jul-Nov 2018)

Constraint Satisfaction

- A Constraint Satisfaction Problem (CSP) has three components
 - 1 X as a set of variables $\{X_1, X_2, X_3, \dots, X_n\}$
 - 2 D as a set of domains $\{D_1, D_2, D_3, \dots, D_n\}$
 - 3 C set of constraints that specify allowable combinations of values
- Each D_i contains allowable set of values $\{v_1, v_2, v_3, \dots, v_k\}$ for X_i
- Each C_i contains a pair $\langle \text{scope}, \text{relation} \rangle$ such as $\langle (X_1, X_3), X_1 \neq X_3 \rangle$
- To solve CSP it needs **state space** and a notion of **solution**
- An assignment of variables such as $(X_1 = v_1, X_2 = v_2, \dots, X_n = v_n)$ that does not violates any constraints is called **consistent** or legal solution.
- Our target to have complete assignment that is consistent.

Example: Map Coloring

Given three colors {red, green, blue}, can you color following map such that no two neighboring region have same color.



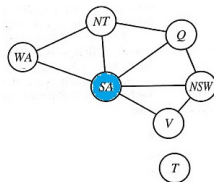
- $X = \{WA, NT, Q, NSW, V, SA, T\}$
- $C = \{SA \neq WA, SA \neq NT, SA \neq Q, SA \neq NSW, SA \neq V, WA \neq NT, NT \neq Q, Q \neq NSW, NSW \neq V\}$
- There are many solution to the problem
- It is helpful to visualize as a **constraint graph** (node: variable, edge: participate in constraint)

Graph Coloring

- With number of available colors equal to the highest degree of the node in the graph, then you could obviously color it
- Any planer graph ¹ can be colored with 5 colors.
- In fact with 4 colors (proof is hard)
- However some spacial graphs, such as a bipartite graph could be colored with only 2 colors
- Given a graph, finding what is its chromatic number (minimum number of colors required to color) is a NP-hard problem

¹When you draw nodes on a sheet and edges do not cross. Three house three utility graph is not planer

Constraint Propagation

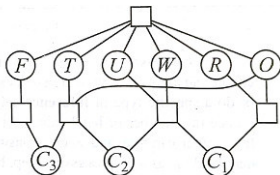


- Once you choose (SA=blue)
- you can infer that none of its five members could take the color blue
- Without taking advantage of constraint propagation, search needs to consider $3^5 = 243$ assignments for the five neighbors.
- With constraint propagation, it needs $2^5 = 32$ only
- 87% reduction

Example: Cryptarithmic Puzzle

Consider following addition (we have to find digits)

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



- Each letter represents a different digit
- We need $\text{Alldiff}\{F, T, U, W, R, O\}$ and
- Additional constraints are

$$\begin{aligned} O + O &= R + 10 \times C_1 \\ C_1 + W + W &= U + 10 \times C_2 \\ C_2 + T + T &= O + 10 \times C_3 \\ C_3 &= F \end{aligned}$$

Preferential Constraints

- Many real world CSPs include **preference constraints**
- Indicating some solution are preferred over other
- Consider university class scheduling problem
 - ▶ Apart from absolute constraints such as no professor could simultaneously teach two classes
 - ▶ There are some preferential constraints such as Prof. A prefer teaching in morning whereas Prof. B prefer teaching on evening.
 - ▶ A solution that schedules Prof. A in evening and Prof. B in morning is still ok
 - ▶ But, we do not prefer it
- Such problems are sometimes called **constraint optimization problem** (COP)

Inference in CSP

- In CSP, an algorithm can either
 - 1 Search or
 - 2 Do inference: constraint propagation (that reduces number of legal values for another variable)

Enforcing **local consistency** in each part of the graph can cause inconsistency elimination throughout the graph.

- **Node consistency:** if all the values in variable's domain satisfy the variable's unary constraints². It is always possible to eliminate all unary constraints by applying *Node consistency*
- **Arc consistency:** X_i is arc consistent with X_j if for every value in current domain D_i there is some value in D_j satisfying binary constraint on (X_i, X_j) . Note³

²If SA do not like green color then use {red,blue} instead of {red,green,blue}

³Generalization is possible: using more than two variables in a constraint ▶

AC-3 Algorithm

function AC-3(*csp*) **returns** false if an inconsistency is found and true otherwise

inputs: *csp*, a binary CSP with components (X, D, C)

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

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

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

if REVISE(*csp*, X_i, X_j) **then**

if size of $D_i = 0$ **then return** false

for each X_k **in** $X_i.\text{NEIGHBORS} - \{X_j\}$ **do**

 add (X_k, X_i) to *queue*

return true

function REVISE(*csp*, X_i, X_j) **returns** true iff we revise the domain of X_i

revised \leftarrow false

for each x **in** D_i **do**

if no value y in D_j allows (x, y) to satisfy the constraint between X_i and X_j **then**

 delete x from D_i

revised \leftarrow true

return *revised*

Takes $O(cd^3)$ time in worst case

Inference in CSP: Path consistency

Arc consistency can help if some domain becomes empty, or size of every domain reduces to 1

- **Path consistency:** two variable set $\{X_i, X_j\}$ is path consistent wrt X_m if for every $\{X_i = a, X_j = b\}$ consistent with constraints on $\{X_i, X_j\}$ there is an assignment to X_m that satisfies constraints on $\{X_i, X_m\}$ and $\{X_m, X_j\}$
- **K-consistency:** for any set of $k - 1$ variables and for any constraint assignment to those variables, a consistent value can always be assigned to any k^{th} variable.
A CSP is strongly k -consistent if it is k -consistent, $k-1$ -consistent, $k-2$ -consistent, ..., 1-consistent ⁴
- **Global Constraints:** like *alldiff*. If all m variable involved in *alldiff* have only n possible values where $m > n$ then there is no solution.

⁴Finding such condition is hard

Example: Sudoku

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

(a)

Example: Sudoku

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

(a)

	1	2	3	4	5	6	7	8	9
A	4	8	3	9	2	1	6	5	7
B	9	6	7	3	4	5	8	2	1
C	2	5	1	8	7	6	4	9	3
D	5	4	8	1	3	2	9	7	6
E	7	2	9	5	6	4	1	3	8
F	1	3	6	7	9	8	2	4	5
G	3	7	2	6	8	9	5	1	4
H	8	1	4	2	5	3	7	6	9
I	6	9	5	4	1	7	3	8	2

(b)

Example: Sudoku

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

(a)

	1	2	3	4	5	6	7	8	9
A	4	8	3	9	2	1	6	5	7
B	9	6	7	3	4	5	8	2	1
C	2	5	1	8	7	6	4	9	3
D	5	4	8	1	3	2	9	7	6
E	7	2	9	5	6	4	1	3	8
F	1	3	6	7	9	8	2	4	5
G	3	7	2	6	8	9	5	1	4
H	8	1	4	2	5	3	7	6	9
I	6	9	5	4	1	7	3	8	2

(b)

$alldiff\{A1, A2, A3, A4, A5, A6, A7, A8, A9\}$

$alldiff\{B1, B2, B3, B4, B5, B6, B7, B8, B9\}$

\vdots

$alldiff\{A1, A2, A3, B1, B2, B3, C1, C2, C3\}$

\vdots

Backtracking Search for CSP

- When inference only do not work, use search
- CSP with n variable and d domain size can have branching factor nd at top level. Then $(n - 1)d$ in next level and so on.
- Tree with $n \cdot d^n$ leaves get generated (however valid assignments are only d^n)
- It is why we have ignored **commutativity**⁵
- So consider single variable at a node.
- Now we need to **backtrack**, if no legal value is left for assignment

⁵No effect of assignment order

Backtracking Search

function BACKTRACKING-SEARCH(*csp*) **returns** a solution, or failure
return BACKTRACK({ }, *csp*)

function BACKTRACK(*assignment*, *csp*) **returns** a solution, or failure
if *assignment* is complete **then return** *assignment*
var \leftarrow SELECT-UNASSIGNED-VARIABLE(*csp*)
for each *value* **in** ORDER-DOMAIN-VALUES(*var*, *assignment*, *csp*) **do**
 if *value* is consistent with *assignment* **then**
 add { *var* = *value* } to *assignment*
 inferences \leftarrow INFERENCE(*csp*, *var*, *value*)
 if *inferences* \neq failure **then**
 add *inferences* to *assignment*
 result \leftarrow BACKTRACK(*assignment*, *csp*)
 if *result* \neq failure **then**
 return *result*
 remove { *var* = *value* } and *inferences* from *assignment*
return failure

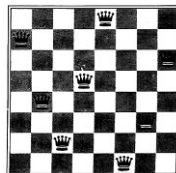
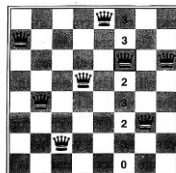
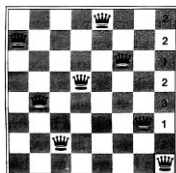
Backtracking Search

- Which variable to choose next? **minimum remaining value**⁶ or “fail first”
- **Degree heuristic**, choose one which is involved in many constraints
- Which value to choose? **least constrained value**
- **Forward checking**: interleaving search and inference would help. Whenever a variable X is assigned, forward checking establishes arc consistency for it.
- **Intelligent Backtracking**: Some time it is needed to backtrack upward more than a single step to resolve the inconsistency. Conflicting set is used to find most suitable node.

⁶chose whose domain have fewer entries

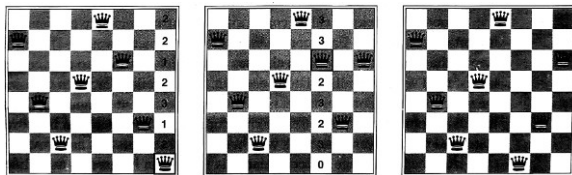
Local Search for CSP

Complete state space formulation can also be used for search



Local Search for CSP

Complete state space formulation can also be used for search



function MIN-CONFLICTS(*esp*, *max_steps*) **returns** a solution or failure

inputs: *esp*, a constraint satisfaction problem

max_steps, the number of steps allowed before giving up

current \leftarrow an initial complete assignment for *esp*

for $i = 1$ to *max_steps* **do**

if *current* is a solution for *esp* **then return** *current*

var \leftarrow a randomly chosen conflicted variable from *esp*.VARIABLES


value \leftarrow the value v for *var* that minimizes CONFLICTS(*var*, v , *current*, *esp*)

set *var* = *value* in *current*

return *failure*

Local Search for CSP

- **Min-conflict:** can solve Million Queen problem in 50 steps on an average
- **Tabu-search:** keeps a small list of recently visited states and forbidding algorithm to return to these states.
- **Constraint weighting:** starting from weight 1 for each variable; in each step, algorithm chooses variable/value such that sum of weights is minimized. Weights of violated variables are incremented.
- Big advantage that the local search can be implemented in online environment.⁷

⁷Airline wants to repair its schedule with minimum changes. 

Exploit Problem Structure

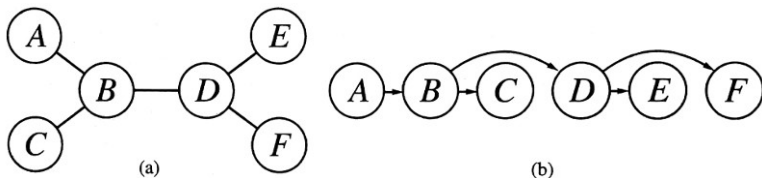
Sometime structure of the problem could help to find solution

- **Independent Subproblems:** see that Tasmania is not connected to mainland in Australia map.

Compare $O(d^c n/c)$ with $O(d^n)$ it is linear

where each sub problem has c variables

- **Tree structured CSP** is solvable in linear time $O(nd^2)$ with topological sort

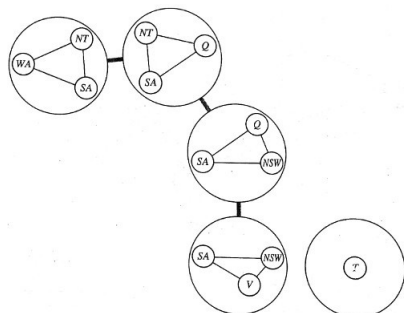
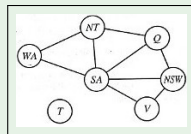


- ▶ **Cut set conditioning:** Assign few, to get tree from remaining vars $O(d^c(n-c)d^2)$
- ▶ Another approach is **tree decomposition**

Tree Decomposition for CSP

Divide the problem in sub-problems

- Every variable in original problem appears at least one of the subproblems
- If two variables are connected by a constraint in original problem, then they must appear together in at least one of the subproblem
- If a variable appears in two subproblems in the tree, it must appear in every subproblem along the path connecting those subproblems



Thank You!

Thank you very much for your attention!

Queries ?

(Reference⁸)

⁸1) Book - *AIMA*, ch-06, Russell and Norvig.