CS-F407: Artificial Intelligence

13 Constraint Satisfaction

Dr. Kamlesh Tiwari
Associate Professor, Department of CSIS,
BITS Pilani, Pilani Campus, Rajasthan-333031 INDIA

Sept 11, 2024 M/W/F - 3:00 pm 6101 @ BITS-Pilani [Aug-Dec 2024]

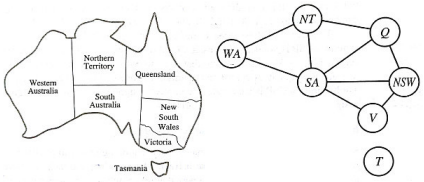
http://ktiwari.in/ai

Constraint Satisfaction

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

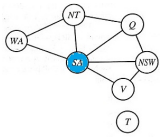
Example: Map Coloring

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≠WA, SA≠NT, SA≠Q, SA≠NSW, SA≠V, WA≠NT, NT≠Q, Q≠NSW, NSW≠V}
- There are many solution to the problem
- It is helpful to visualize as a **constraint graph** (node: variable, edge: participate in constraint)

Constraint Propagation

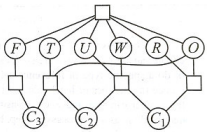


- Once you choose (SA=blue)
- you can conclude that none of its five members could take the color blue
- Without taking advantage of constraint propagation, search needs to consider $3^5 = 243$ assignments for the five neighbors.
- With constraint propagation, it needs $2^5 = 32$ only
- 87% reduction

Example: Cryptarithmethc 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 Alldiff{F,T,U,W,R,O} and
- Additional constraints are

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

Is it so 234+234=0468?

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 constraints [1]. 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). Note [2]

[1] If SA do not like green color then use {red,blue} instead of {red,green,blue}
[2] Generalization 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**
    $(X_i, X_j) \leftarrow$ REMOVE-FIRST($queue$)
    if REVISE(csp, X_i, X_j) **then**
      if size of $D_i = 0$ **then return** $false$
      for each X_k **in** X_i.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

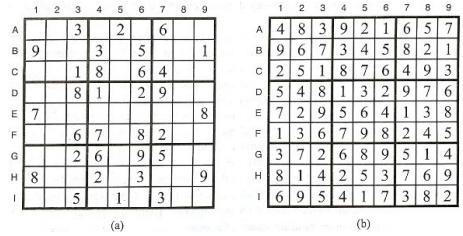
Inference in CSP: Path consistency

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 is it is k-consistent, k-1-consistent, k-2-consistent, ..., 1-consistent [3]
- **Global Constraints:** like *alldiff*. If all m variable involved in alldiff have only n possible values where $m > n$ then there is no solution.

[3] Finding such considition is hard

Example: Sudoku



$alldiff\{A1, A2, A3, A4, A5, A6, A7, A8, A9\}$
$alldiff\{B1, B2, B3, B4, B5, B6, B7, B8, B9\}$
$\vdots$
$alldiff\{A1, A2, A3, B1, B2, B3, C1, C2, C3\}$
$\vdots$

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 $!n.d^n$ leaves get generated (however valid assignments are only d^n)
- It is why we have ignored **commutativity** [4]
- So consider single variable at a node.
- Now we need to **backtrack**, if no legal value is left for assignment

[4] No 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** [5] 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.

[5] chose whose domain have fewer entries

Local Search for CSP

Complete state space formulation can also be used for search

function MIN-CONFLICTS(csp, max_steps) **returns** a solution or failure
inputs: csp, a constraint satisfaction problem
max_steps, the number of steps allowed before giving up

$current \leftarrow$ an initial complete assignment for csp
for $i = 1$ to max_steps **do**
if $current$ is a solution for csp **then return** $current$
$var \leftarrow$ a randomly chosen conflicted variable from csp.VARIABLES
$value \leftarrow$ the value v for var that minimizes CONFLICTS($var, v, current, csp$)
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 choses 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.[6]

[6] Airline wants to repair its schedule with minimum changes.

Exploit Problem Structure

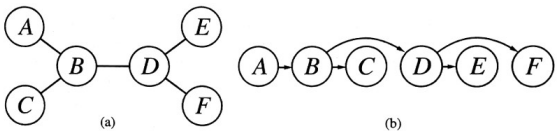
Sometime structure of the problem could help to find solution

- **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

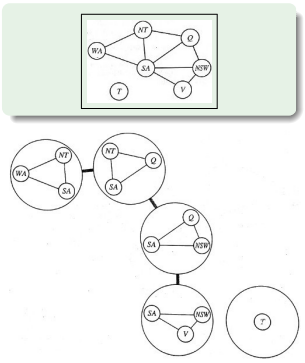


  - **Cut set conditioning:** Assign few, to get tree from remaining vars $O(d^c(n-c)d^2)$
  - Another approach is **tree decomposition**

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 ?

(Reference[7])

[7] 1) Book - *AIMA*, ch-06, Russell and Norvig.