Constraint Satisfaction Problems

Tushar B. Kute,
http://tusharkute.com

Search Algorithm

- The objective of every problem-solving technique is one, i.e., to find a solution to reach the goal.
- Although, in adversarial search and local search, there were no constraints on the agents while solving the problems and reaching to its solutions.

Constraint satisfaction

- Constraint satisfaction is a technique where a problem is solved when its values satisfy certain constraints or rules of the problem. Such type of technique leads to a deeper understanding of the problem structure as well as its complexity.
- Constraint satisfaction depends on three components, namely:
  - X: It is a set of variables.
  - D: It is a set of domains where the variables reside. There is a specific domain for each variable.
  - C: It is a set of constraints which are followed by the set of variables.

Constraint satisfaction

- In constraint satisfaction, domains are the spaces where the variables reside, following the problem specific constraints.
- These are the three main elements of a constraint satisfaction technique. The constraint value consists of a pair of {scope, rel}.
- The scope is a tuple of variables which participate in the constraint and rel is a relation which includes a list of values which the variables can take to satisfy the constraints of the problem.

Constraint satisfaction

- Solving Constraint Satisfaction Problems
- The requirements to solve a constraint satisfaction problem (CSP) is:
  - A state-space
  - The notion of the solution.
- A state in state-space is defined by assigning values to some or all variables such as

  {X1=v1, X2=v2, and so on…}.

Constraint satisfaction

- An assignment of values to a variable can be done in three ways:
  - Consistent or Legal Assignment: An assignment which does not violate any constraint or rule is called Consistent or legal assignment.
  - Complete Assignment: An assignment where every variable is assigned with a value, and the solution to the CSP remains consistent. Such assignment is known as Complete assignment.
  - Partial Assignment: An assignment which assigns values to some of the variables only. Such type of assignments are called Partial assignments.

Types of Domains in CSP

- There are following two types of domains which are used by the variables :
  - Discrete Domain: It is an infinite domain which can have one state for multiple variables. For example, a start state can be allocated infinite times for each variable.
  - Finite Domain: It is a finite domain which can have continuous states describing one domain for one specific variable. It is also called a continuous domain.

Constraint Types in CSP

- With respect to the variables, basically there are following types of constraints:
  - Unary Constraints: It is the simplest type of constraints that restricts the value of a single variable.
  - Binary Constraints: It is the constraint type which relates two variables. A value x2 will contain a value which lies between x1 and x3.
  - Global Constraints: It is the constraint type which involves an arbitrary number of variables.

Constraint Types in CSP

- Some special types of solution algorithms are used to solve the following types of constraints:
  - Linear Constraints: These type of constraints are commonly used in linear programming where each variable containing an integer value exists in linear form only.
  - Non-linear Constraints: These type of constraints are used in non-linear programming where each variable (an integer value) exists in a non-linear form.

Constraint Propagation

- In local state-spaces, the choice is only one, i.e., to search for a solution. But in CSP, we have two choices either:
  - We can search for a solution or
  - We can perform a special type of inference called constraint propagation.
- Constraint propagation is a special type of inference which helps in reducing the legal number of values for the variables.
- The idea behind constraint propagation is local consistency.

Local consistency

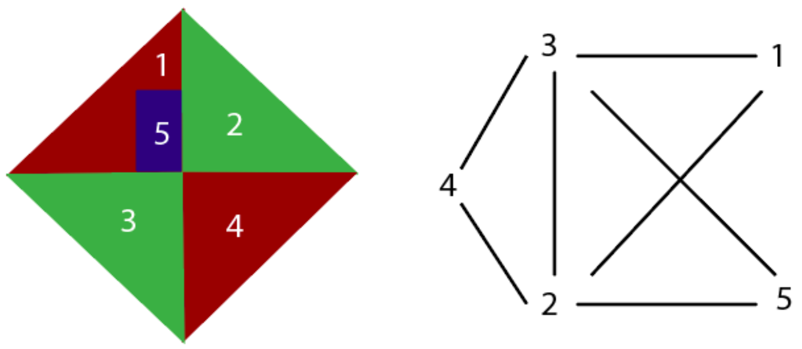
- In local consistency, variables are treated as nodes, and each binary constraint is treated as an arc in the given problem. There are following local consistencies which are discussed below:
  - Node Consistency: A single variable is said to be node consistent if all the values in the variable's domain satisfy the unary constraints on the variables.
  - Arc Consistency: A variable is arc consistent if every value in its domain satisfies the binary constraints of the variables.

Local consistency

- Path Consistency: When the evaluation of a set of two variable with respect to a third variable can be extended over another variable, satisfying all the binary constraints. It is similar to arc consistency.
- k-consistency: This type of consistency is used to define the notion of stronger forms of propagation. Here, we examine the k-consistency of the variables.

CSP Problems

- Constraint satisfaction includes those problems which contains some constraints while solving the problem. CSP includes the following problems:
- Graph Coloring: The problem where the constraint is that no adjacent sides can have the same color.

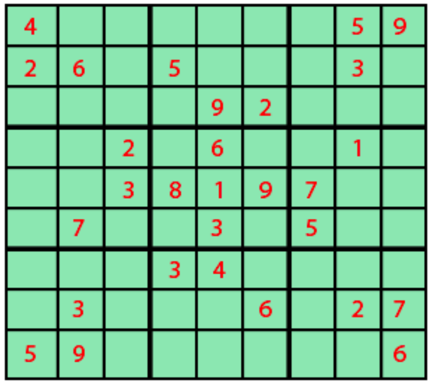


Graph Coloring

CSP Problems

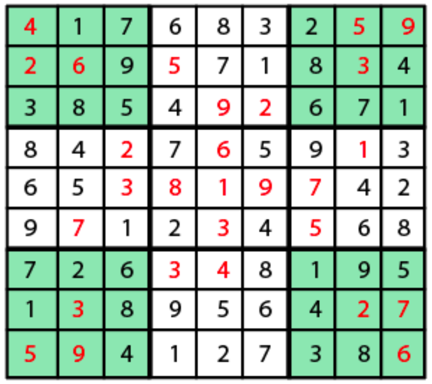
- Sudoku Playing: The gameplay where the constraint is that no number from 0-9 can be repeated in the same row or column.

SUDOKU



4							5	9
2	6		5				3	
				9	2			
		2		6			1	
		3	8	1	9	7		
	7			3		5		
			3	4				
	3				6		2	7
5	9							6

Puzzle



4	1	7	6	8	3	2	5	9
2	6	9	5	7	1	8	3	4
3	8	5	4	9	2	6	7	1
8	4	2	7	6	5	9	1	3
6	5	3	8	1	9	7	4	2
9	7	1	2	3	4	5	6	8
7	2	6	3	4	8	1	9	5
1	3	8	9	5	6	4	2	7
5	9	4	1	2	7	3	8	6

Solution

Inference in CSPs

- A number of inference techniques use the constraints to infer which variable/value pairs are consistent and which are not. These include node, arc, path, and k-consistent.
- constraint propagation: Using the constraints to reduce the number of legal values for a variable, which in turn can reduce the legal values for another variable, and so on.
- local consistency: If we treat each variable as a node in a graph and each binary constraint as an arc, then the process of enforcing local consistency in each part of the graph causes inconsistent values to be eliminated throughout the graph.

Inference in CSPs

- Node consistency
  - A single variable (a node in the CSP network) is node-consistent if all the values in the variable’s domain satisfy the variable’s unary constraint.
  - We say that a network is node-consistent if every variable in the network is node-consistent.

Inference in CSPs

- Arc consistency
  - A variable in a CSP is arc-consistent if every value in its domain satisfies the variable’s binary constraints.
  - Xi is arc-consistent with respect to another variable Xj if for every value in the current domain Di there is some value in the domain Dj that satisfies the binary constraint on the arc (Xi, Xj).
  - A network is arc-consistent if every variable is arc-consistent with every other variable.
  - Arc consistency tightens down the domains (unary constraint) using the arcs (binary constraints).

Inference in CSPs

- Path consistency
  - Path consistency: A two-variable set {Xi, Xj} is path-consistent with respect to a third variable Xm if, for every assignment {Xi = a, Xj = b} consistent with the constraint on {Xi, Xj}, there is an assignment to Xm that satisfies the constraints on {Xi, Xm} and {Xm, Xj}.
  - Path consistency tightens the binary constraints by using implicit constraints that are inferred by looking at triples of variables.

Inference in CSPs

- K-consistency
  - K-consistency: A CSP is k-consistent if, for any set of k-1 variables and for any consistent assignment to those variables, a consistent value can always be assigned to any kth variable.
  - 1-consistency = node consistency; 2-consisency = arc consistency; 3-consistensy = path consistency.
  - A CSP is strongly k-consistent if it is k-consistent and is also (k - 1)-consistent, (k – 2)-consistent, … all the way down to 1-consistent.
  - A CSP with n nodes and make it strongly n-consistent, we are guaranteed to find a solution in time O(n2d). But algorithm for establishing n-consitentcy must take time exponential in n in the worse case, also requires space that is exponential in n.

Inference in CSPs

- Global constraints
  - A global constraint is one involving an arbitrary number of variables (but not necessarily all variables).
  - Global constraints can be handled by special-purpose algorithms that are more efficient than general-purpose methods.

Global constraints

- 1) inconsistency detection for Alldiff constraints
  - A simple algorithm: First remove any variable in the constraint that has a singleton domain, and delete that variable’s value from the domains of the remaining variables. Repeat as long as there are singleton variables. If at any point an empty domain is produced or there are more vairables than domain values left, then an inconsistency has been detected.
  - A simple consistency procedure for a higher-order constraint is sometimes more effective than applying arc consistency to an equivalent set of binary constrains.

Global constraints

- 2) inconsistency detection for resource constraint (the atmost constraint)
- We can detect an inconsistency simply by checking the sum of the minimum of the current domains;
- e.g. Atmost(10, P1, P2, P3, P4): no more than 10 personnel are assigned in total.
- If each variable has the domain {3, 4, 5, 6}, the Atmost constraint cannot be satisfied.
- We can enforce consistency by deleting the maximum value of any domain if it is not consistent with the minimum values of the other domains.
- e.g. If each variable in the example has the domain {2, 3, 4, 5, 6}, the values 5 and 6 can be deleted from each domain.

Global constraints

- 3) inconsistency detection for bounds consistent
- For large resource-limited problems with integer values, domains are represented by upper and lower bounds and are managed by bounds propagation.
- e.g. suppose there are two flights F1 and F2 in an airline-scheduling problem, for which the planes have capacities 165 and 385, respectively. The initial domains for the numbers of passengers on each flight are
- D1 = [0, 165] and D2 = [0, 385].

Global constraints

- Now suppose we have the additional constraint that the two flight together must carry 420 people: F1 + F2 = 420. Propagating bounds constraints, we reduce the domains to

  D1 = [35, 165] and D2 = [255, 385].
- A CSP is bounds consistent if for every variable X, and for both the lower-bound and upper-bound values of X, there exists some value of Y that satisfies the constraint between X and Y for every variable Y.

Sudoku

- A Sudoku puzzle can be considered a CSP with 81 variables, one for each square. We use the variable names A1 through A9 for the top row (left to right), down to I1 through I9 for the bottom row.
- The empty squares have the domain {1, 2, 3, 4, 5, 6, 7, 8, 9} and the pre-filled squares have a domain consisting of a single value.

Sudoku

- There are 27 different Alldiff constraints: one for each row, column, and box of 9 squares:
- Alldiff(A1, A2, A3, A4, A5, A6, A7, A8, A9)
- Alldiff(B1, B2, B3, B4, B5, B6, B7, B8, B9)
- …
- Alldiff(A1, B1, C1, D1, E1, F1, G1, H1, I1)
- Alldiff(A2, B2, C2, D2, E2, F2, G2, H2, I2)
- …
- Alldiff(A1, A2, A3, B1, B2, B3, C1, C2, C3)
- Alldiff(A4, A5, A6, B4, B5, B6, C4, C5, C6)
- …

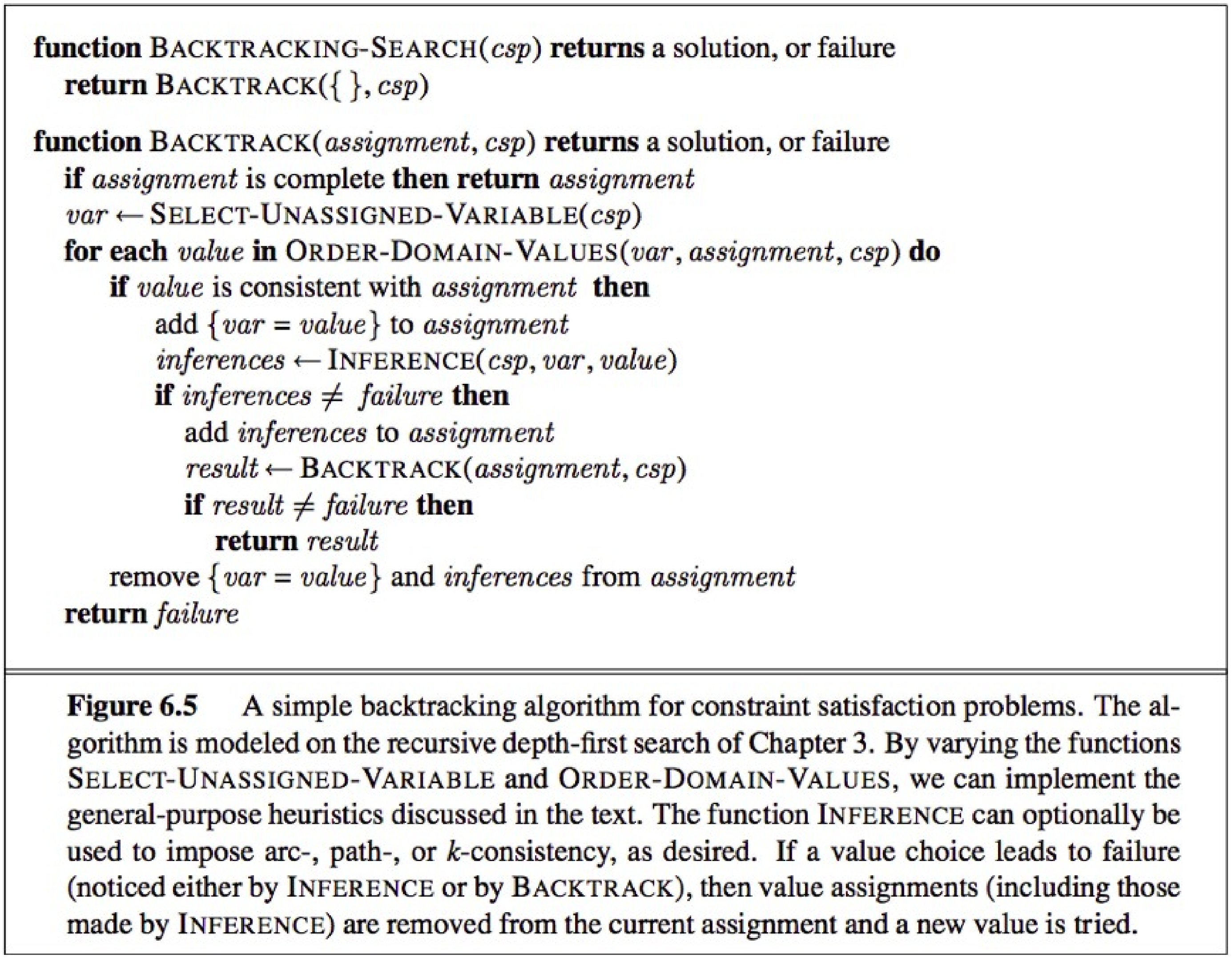
Backtracking search for CSPs

- Backtracking search, a form of depth-first search, is commonly used for solving CSPs. Inference can be interwoven with search.
- Commutativity: CSPs are all commutative. A problem is commutative if the order of application of any given set of actions has no effect on the outcome.
- Backtracking search: A depth-first search that chooses values for one variable at a time and backtracks when a variable has no legal values left to assign.

Backtracking search for CSPs

- Backtracking algorithm repeatedly chooses an unassigned variable, and then tries all values in the domain of that variable in turn, trying to find a solution. If an inconsistency is detected, then BACKTRACK returns failure, causing the previous call to try another value.
- There is no need to supply BACKTRACKING-SEARCH with a domain-specific initial state, action function, transition model, or goal test.
- BACKTRACKING-SARCH keeps only a single representation of a state and alters that representation rather than creating a new ones.

Backtracking search for CSPs



```
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 ← 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 ← INFERENCE(csp, var, value)
        if inferences ≠ failure then
           add inferences to assignment
           result ← BACKTRACK(assignment, csp)
           if result ≠ failure then
              return result
     remove {var = value} and inferences from assignment
  return failure
```

Figure 6.5 A simple backtracking algorithm for constraint satisfaction problems. The algorithm is modeled on the recursive depth-first search of Chapter 3. By varying the functions SELECT-UNASSIGNED-VARIABLE and ORDER-DOMAIN-VALUES, we can implement the general-purpose heuristics discussed in the text. The function INFERENCE can optionally be used to impose arc-, path-, or *k*-consistency, as desired. If a value choice leads to failure (noticed either by INFERENCE or by BACKTRACK), then value assignments (including those made by INFERENCE) are removed from the current assignment and a new value is tried.

Backtracking search for CSPs

- To solve CSPs efficiently without domain-specific knowledge, address following questions:
- 1)function SELECT-UNASSIGNED-VARIABLE: which variable should be assigned next?
- function ORDER-DOMAIN-VALUES: in what order should its values be tried?
- 2)function INFERENCE: what inferences should be performed at each step in the search?
- 3)When the search arrives at an assignment that violates a constraint, can the search avoid repeating this failure?

Intelligent backtracking

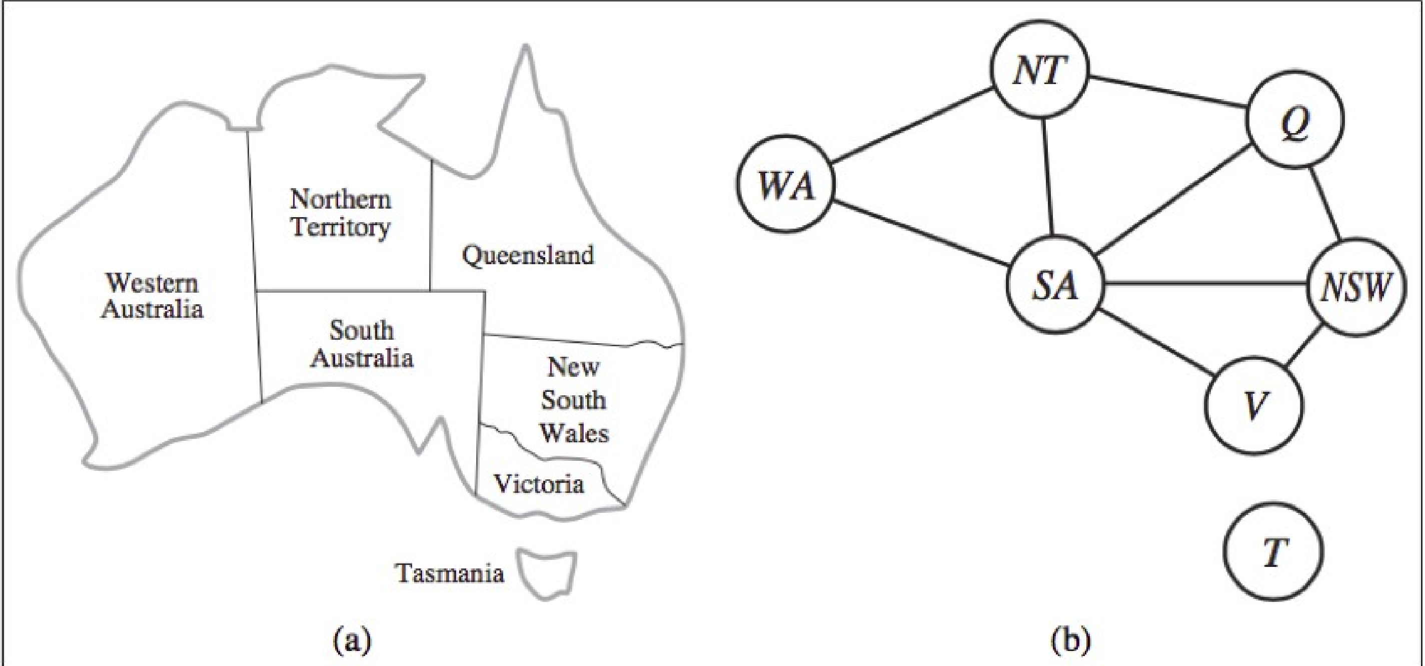
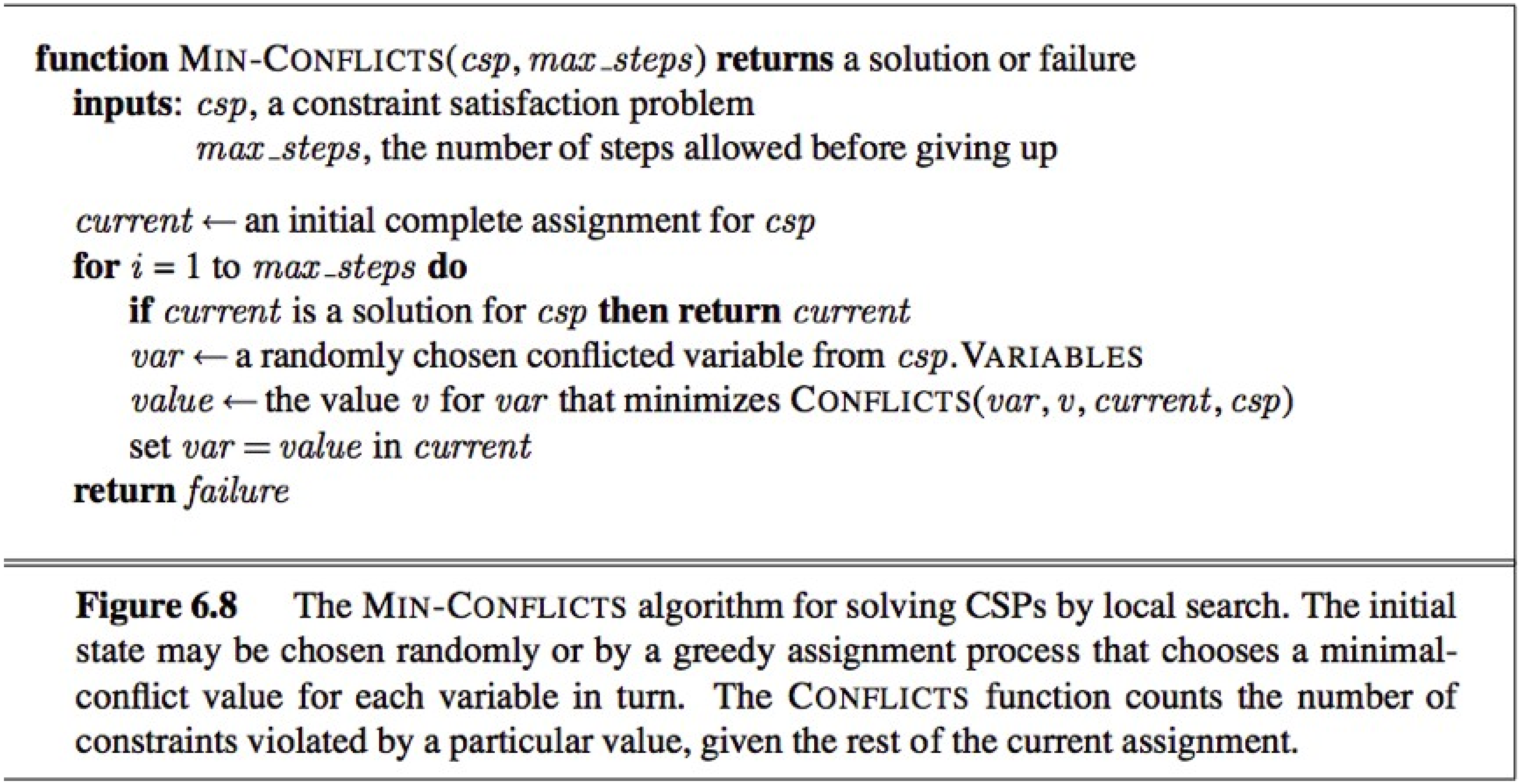


Figure 6.1 (a) The principal states and territories of Australia. Coloring this map can be viewed as a constraint satisfaction problem (CSP). The goal is to assign colors to each region so that no neighboring regions have the same color. (b) The map-coloring problem represented as a constraint graph.

Local search for CSPs

- Local search algorithms for CSPs use a complete-state formulation: the initial state assigns a value to every variable, and the search change the value of one variable at a time.
- The min-conflicts heuristic: In choosing a new value for a variable, select the value that results in the minimum number of conflicts with other variables.

Local search for CSPs



```
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 ← an initial complete assignment for csp
  for i = 1 to max_steps do
     if current is a solution for csp then return current
     var ← a randomly chosen conflicted variable from csp.VARIABLES
     value ← the value v for var that minimizes CONFLICTS(var, v, current, csp)
     set var = value in current
  return failure
```

Figure 6.8 The MIN-CONFLICTS algorithm for solving CSPs by local search. The initial state may be chosen randomly or by a greedy assignment process that chooses a minimal-conflict value for each variable in turn. The CONFLICTS function counts the number of constraints violated by a particular value, given the rest of the current assignment.

Local search for CSPs

- The landscape of a CSP under the mini-conflicts heuristic usually has a series of plateau. Simulated annealing and Plateau search (i.e. allowing sideways moves to another state with the same score) can help local search find its way off the plateau.
- This wandering on the plateau can be directed with tabu search: keeping a small list of recently visited states and forbidding the algorithm to return to those tates.

Local search for CSPs

- Constraint weighting: a technique that can help concentrate the search on the important constraints.
- Each constraint is given a numeric weight Wi, initially all 1.
- At each step, the algorithm chooses a variable/value pair to change that will result in the lowest total weight of all violated constraints.
- The weights are then adjusted by incrementing the weight of each constraint that is violated by the current assignment.
- Local search can be used in an online setting when the problem changes, this is particularly important in scheduling problems.

Structure of Problem

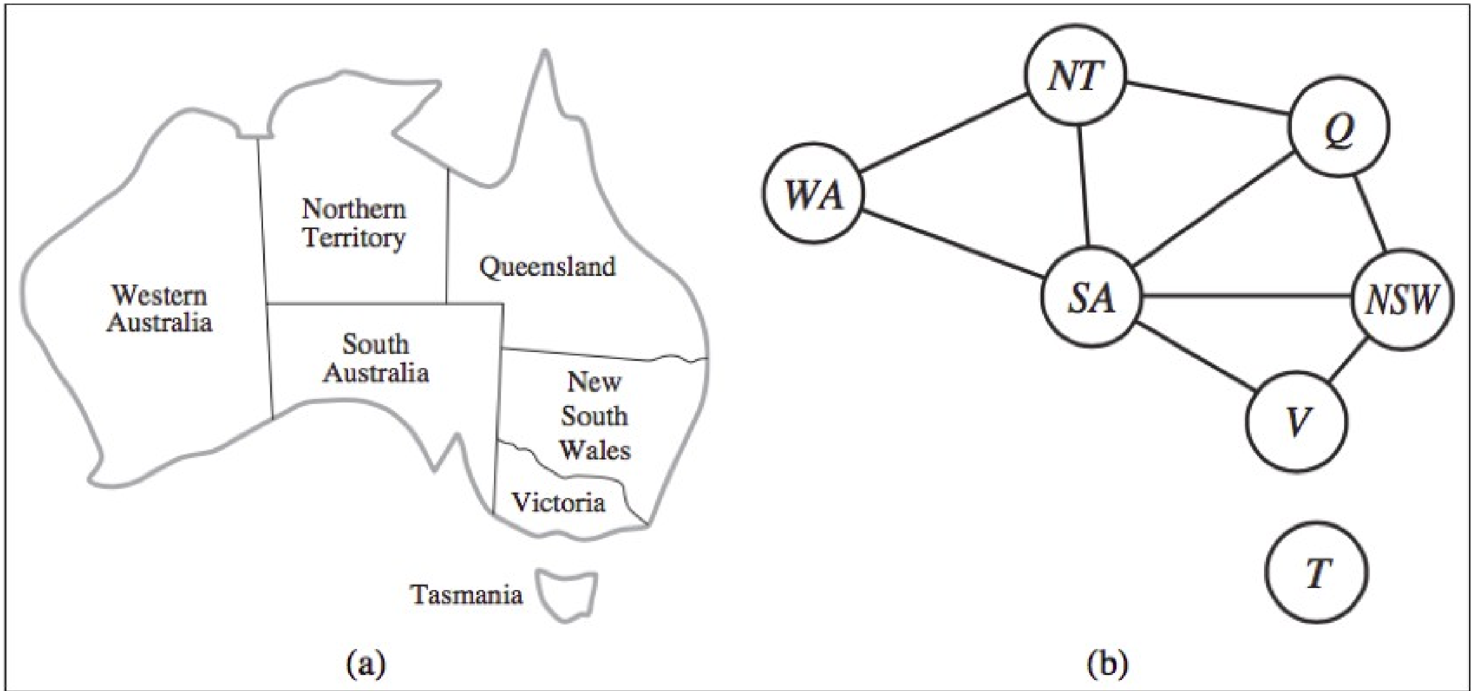


Figure 6.1 (a) The principal states and territories of Australia. Coloring this map can be viewed as a constraint satisfaction problem (CSP). The goal is to assign colors to each region so that no neighboring regions have the same color. (b) The map-coloring problem represented as a constraint graph.

The structure of constraint graph

- The structure of the problem as represented by the constraint graph can be used to find solution quickly.
- e.g. The problem can be decomposed into 2 independent subproblems: Coloring T and coloring the mainland.
- Tree: A constraint graph is a tree when any two varyiable are connected by only one path.
- Directed arc consistency (DAC): A CSP is defined to be directed arc-consistent under an ordering of variables X1, X2, … , Xn if and only if every Xi is arc-consistent with each Xj for j>i.
- By using DAC, any tree-structured CSP can be solved in time linear in the number of variables.

The structure of constraint graph

- Pick any variable to be the root of the tree;
- Choose an ordering of the variable such that each variable appears after its parent in the tree. (topological sort)
- Any tree with n nodes has n-1 arcs, so we can make this graph directed arc-consistent in O(n) steps, each of which must compare up to d possible domain values for 2 variables, for a total time of O(nd2)
- Once we have a directed arc-consistent graph, we can just march down the list of variables and choose any remaining value.
- Since each link from a parent to its child is arc consistent, we won’t have to backtrack, and can move linearly through the variables.

The structure of constraint graph

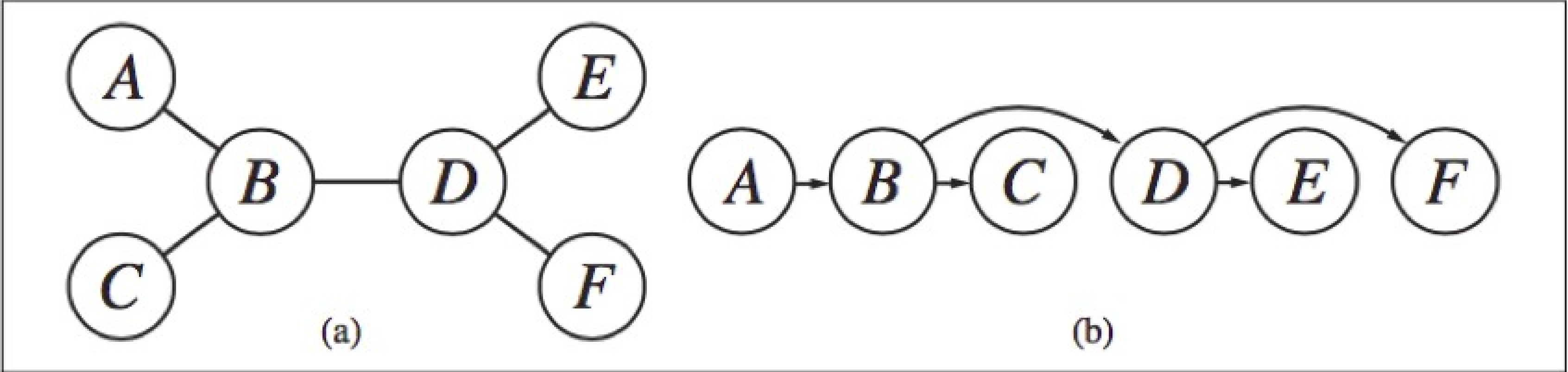
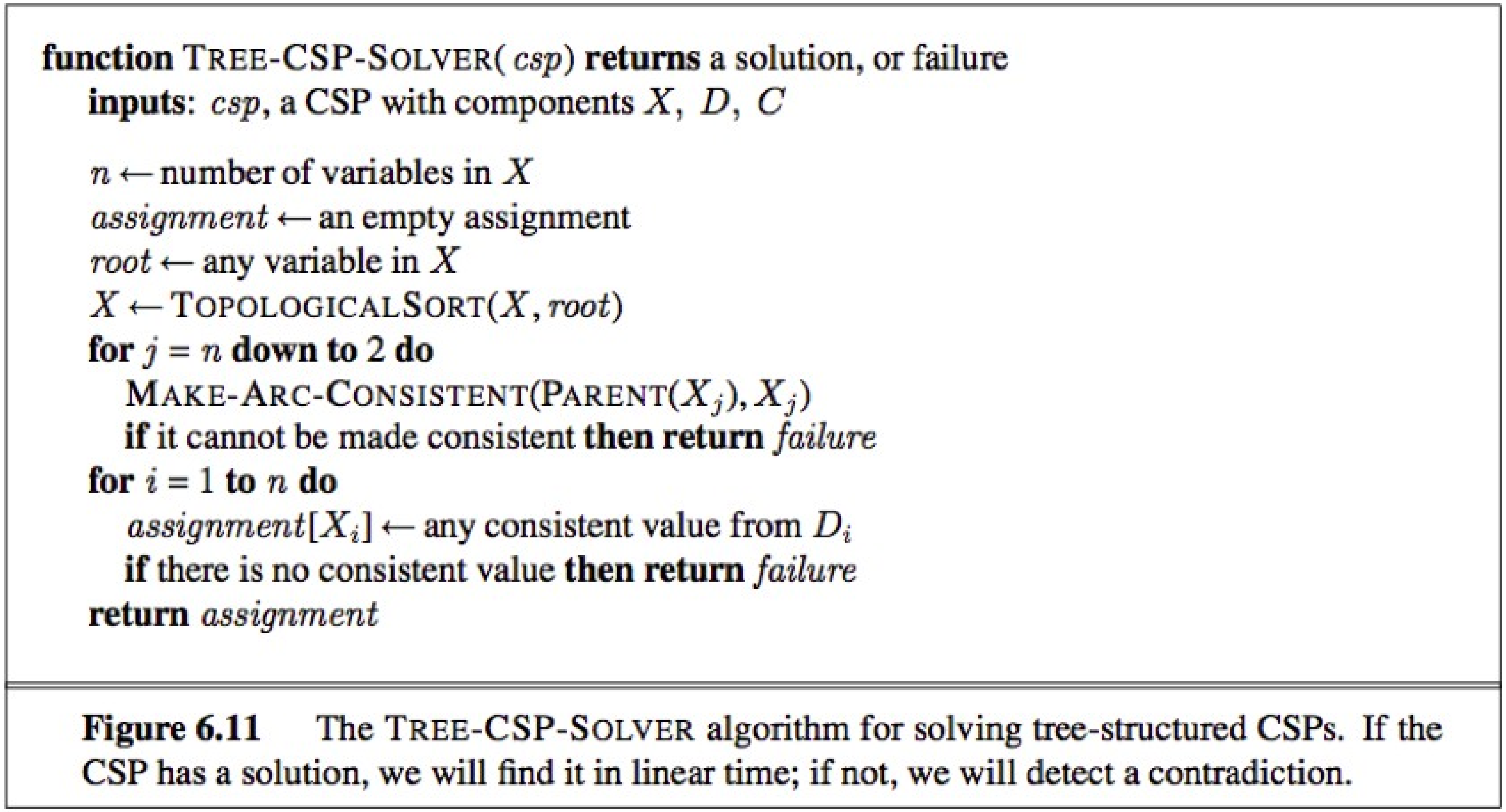


Figure 6.10 (a) The constraint graph of a tree-structured CSP. (b) A linear ordering of the variables consistent with the tree with A as the root. This is known as a **topological sort** of the variables.

The structure of constraint graph



```
function TREE-CSP-SOLVER( csp) returns a solution, or failure
   inputs: csp, a CSP with components $X$, $D$, $C$

   $n \leftarrow$ number of variables in $X$
   assignment $\leftarrow$ an empty assignment
   root $\leftarrow$ any variable in $X$
   $X \leftarrow$ TOPOLOGICALSORT($X$, root)
   for $j = n$ down to 2 do
      MAKE-ARC-CONSISTENT(PARENT($X_j$), $X_j$)
      if it cannot be made consistent then return failure
   for $i = 1$ to $n$ do
      assignment[$X_i$] $\leftarrow$ any consistent value from $D_i$
      if there is no consistent value then return failure
   return assignment
```

Figure 6.11 The TREE-CSP-SOLVER algorithm for solving tree-structured CSPs. If the CSP has a solution, we will find it in linear time; if not, we will detect a contradiction.

Thank you

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com