GITAM
(DEEMED TO BE UNIVERSITY)
VISAKHAPATNAM • HYDERABAD • BENGALURU

INTRODUCTION TO ARTIFICIAL INTELLIGENCE AND ITS APPLICATIONS

Code:19EAI232

Dr. Praveen Gupta
Asst. Professor
Department of Computer Science and Engineering,
GITAM Institute of Technology,
GITAM (Deemed to be University), Visakhapatnam(AP), India
Email Id: pgupta@gitam.edu , M: 9783750898

MODULE III: ROADMAP

- Constraint Satisfaction Problems:
  - ✓ Defining Constraint Satisfaction Problems,
  - ✓ Constraint Propagation:
    - Inference in CSPs,
    - Backtracking Search for CSPs,
    - Local Search for CSPs,
    - The Structure of Problems.
- Local Agents:
  - ✓ Knowledge-based Agents,
  - ✓ The Wumpus World, Logic,
  - ✓ Propositional Logic:
    - A Very Simple Logic,
    - Propositional Theorem Proving.

LEARNING OUTCOMES:

After completion of this module, the student will be able to:

- define constraint satisfaction problems (L1)
- illustrate inference in constraint satisfaction problems (L2)
- contrast backtracking search and local search for constraint satisfaction problems (L2)
- explain knowledge - based agent (L2)
- apply propositional logic for real time problems (L3)

MODULE III: ROADMAP

- Constraint Satisfaction Problems:
  - ✓ Defining Constraint Satisfaction Problems,
  - ✓ Constraint Propagation:
    - Inference in CSPs,
    - Backtracking Search for CSPs,
    - Local Search for CSPs,
    - The Structure of Problems.
- Local Agents:
  - ✓ Knowledge-based Agents,
  - ✓ The Wumpus World, Logic,
  - ✓ Propositional Logic:
    - A Very Simple Logic,
    - Propositional Theorem Proving.

CONSTRAINT SATISFACTION PROBLEMS

- Constraint satisfaction problems (CSPs) are **mathematical questions defined** as a set of objects whose state must satisfy a number of constraints or limitations.
- CSPs represent the entities in a problem as a homogeneous collection of finite constraints over variables, which is solved by constraint satisfaction methods.
- In artificial intelligence and operations research, constraint satisfaction is the process of finding a solution to a set of constraints that impose conditions that the variables must satisfy.

CONSTRAINT SATISFACTION PROBLEMS

- A constraint satisfaction problem (CSP) is a problem that requires its solution within some limitations or conditions also known as constraints.
- It consists of the following:

  ➢ A finite set of variables which stores the solution (V = {V1, V2, V3,....., Vn})

  ➢ A set of discrete values known as domain from which the solution is picked (D = {D1, D2, D3,.....,Dn})

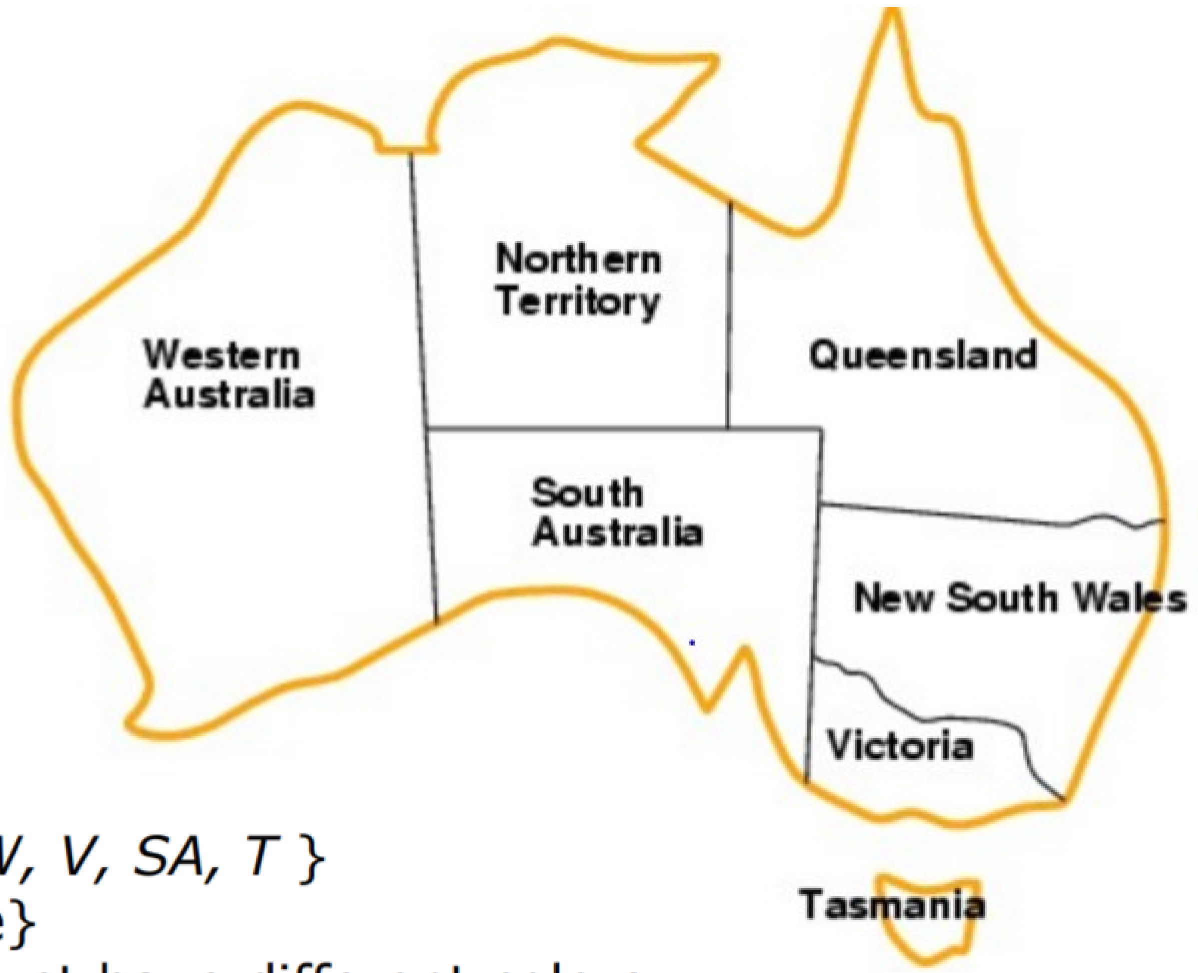
  ➢ A finite set of constraints (C = {C1, C2, C3,......, Cn})

Please note, that the elements in the domain can be both continuous and discrete but in AI, we generally only deal with discrete values.

CONSTRAINT SATISFACTION PROBLEMS

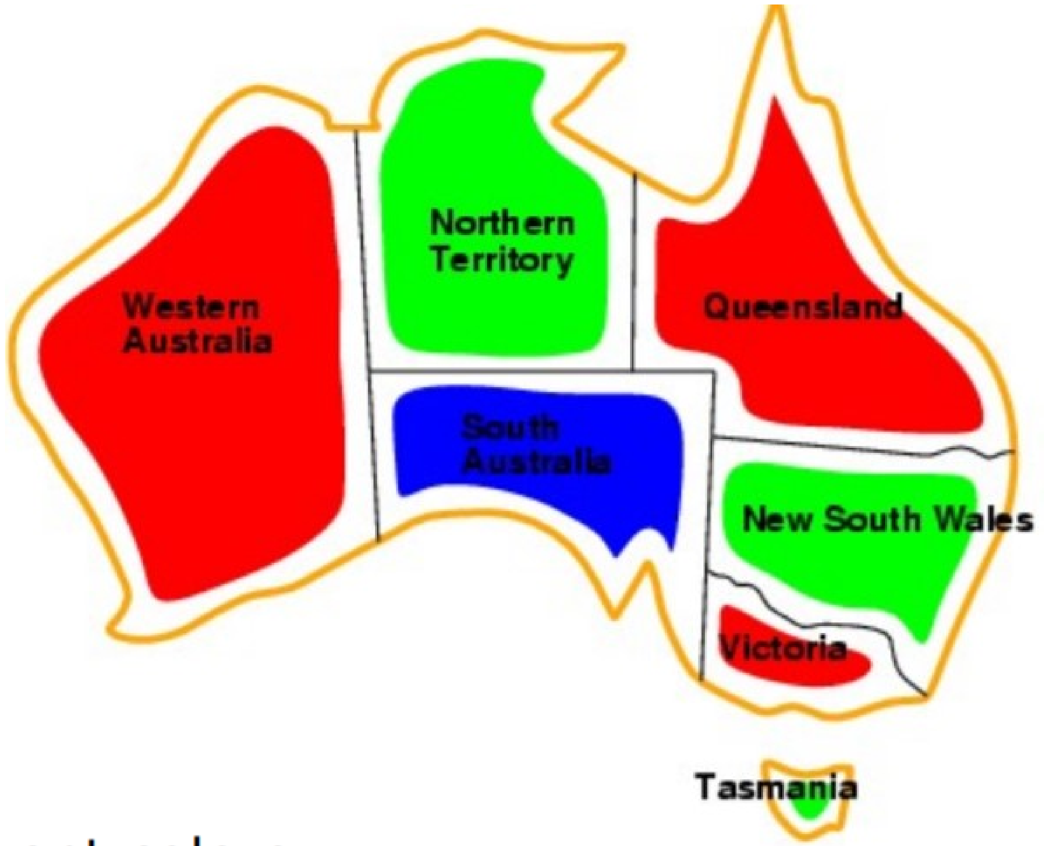
- Each state in a CSP is defined by an assignment of values to some or all of the variables
- An assignment that does not violate any constraints is called a consistent or legal assignment
- A complete assignment is one in which every variable is assigned
- A solution to a CSP is consistent and complete assignment
- Allows useful general-purpose algorithms with more power than standard search algorithms

EXAMPLE: MAP COLORING



Variables: X = {WA, NT, Q, NSW, V, SA, T }
Domains: D_i = {red, green, blue}
Constraints: adjacent regions must have different colors
Solution?

EXAMPLE: MAP COLORING



Variables: X = {*WA, NT, Q, NSW, V, SA, T* }
Domains: D_i = {red, green, blue}
Constraints: adjacent regions must have different colors
Solution? {WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = red}.

TYPES OF VARIABLES

- Discrete variables
  - ❖ finite domains:
    - ➢ n variables, domain size d →O (d n) complete assignments
    - ➢ e.g., Boolean CSPs, such as 3-SAT (NP-complete)
    - ➢ Worst case, can’t solve finite-domain CSPs in less than exponential time
  - ❖ infinite domains:
    - ➢ integers, strings, etc.
    - ➢ e.g., job scheduling, variables are start/end days for each job
    - ➢ need a constraint language, e.g., StartJob1 + 5 ≤ StartJob3
- Continuous variables
  - ➢ e.g., start/end times for Hubble Space Telescope observations
  - ➢ linear constraints solvable in polynomial time by linear programming

TYPES OF CONSTRAINTS

- Unary constraints involve a single variable,
  - e.g., SA ≠ green
- Binary constraints involve pairs of variables,
  - e.g., SA ≠ WA
- Higher-order constraints involve 3 or more variables
  - e.g., cryptarithmetic column constraints

POPULAR PROBLEMS WITH CSP

- The following problems are some of the popular problems that can be solved using CSP:

  ✓ CryptArithmetic (Coding alphabets to numbers)

  ✓ n-Queen (In an n-queen problem, n queens should be placed in an nXn matrix such that no queen shares the same row, column or diagonal)

  ✓ Map Coloring (coloring different regions of map, ensuring no adjacent regions have the same color)

  ✓ Crossword (everyday puzzles appearing in newspapers)

  ✓ Sudoku (a number grid)

  ✓ Latin Square Problem

REAL WORLD CSPs

- Assignment problems
  - e.g., who teaches what class
- Timetabling problems
  - e.g., which class is offered when and where?
- Transportation scheduling
- Factory scheduling
- Hardware configuration
- Spreadsheets
- Floorplanning

MODULE III: ROADMAP

- Constraint Satisfaction Problems:
  - ✓ Defining Constraint Satisfaction Problems,
  - ✓ Constraint Propagation:
    - Inference in CSPs,
    - Backtracking Search for CSPs,
    - Local Search for CSPs,
    - The Structure of Problems.

- Local Agents:
  - ✓ Knowledge-based Agents,
  - ✓ The Wumpus World, Logic,
  - ✓ Propositional Logic:
    - A Very Simple Logic,
    - Propositional Theorem Proving.

CONSTRAINT PROPAGATION

- 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.
- 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
- 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.

LOCAL CONSISTENCY

- There are different types of local consistency:

  i. 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.

  ii. 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).

LOCAL CONSISTENCY

iii. 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.

LOCAL CONSISTENCY

iv. 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.

MODULE III: ROADMAP

- Constraint Satisfaction Problems:
  - ✓ Defining Constraint Satisfaction Problems,
  - ✓ Constraint Propagation:
    - Inference in CSPs,
    - Backtracking Search for CSPs,
    - Local Search for CSPs,
    - The Structure of Problems.

- Local Agents:
  - ✓ Knowledge-based Agents,
  - ✓ The Wumpus World, Logic,
  - ✓ Propositional Logic:
    - A Very Simple Logic,
    - Propositional Theorem Proving.

BACKTRACKING SEARCH FOR CSPS

- Backtracking is an algorithmic-technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time (by time, here, is referred to the time elapsed till reaching any level of the search tree).
- Only need to consider assignments to a single variable at each node →b = d and there are d^n leaves
- Backtracking search is used for 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 is the basic uninformed algorithm for CSPs

BACKTRACKING PROPERTIES

Variable assignments are commutative, i.e.,

[$WA = red$ then $NT = green$] same as [$NT = green$ then $WA = red$]

Only need to consider assignments to a single variable at each node

$\Rightarrow$ $b = d$ and there are d^n leaves

Depth-first search for CSPs with single-variable assignments is called backtracking search

Backtracking search is the basic uninformed algorithm for CSPs

BACKTRACKING EXAMPLE

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.
$\Rightarrow$ fail if no legal assignments (not fixable!)

◇ Goal test: the current assignment is complete

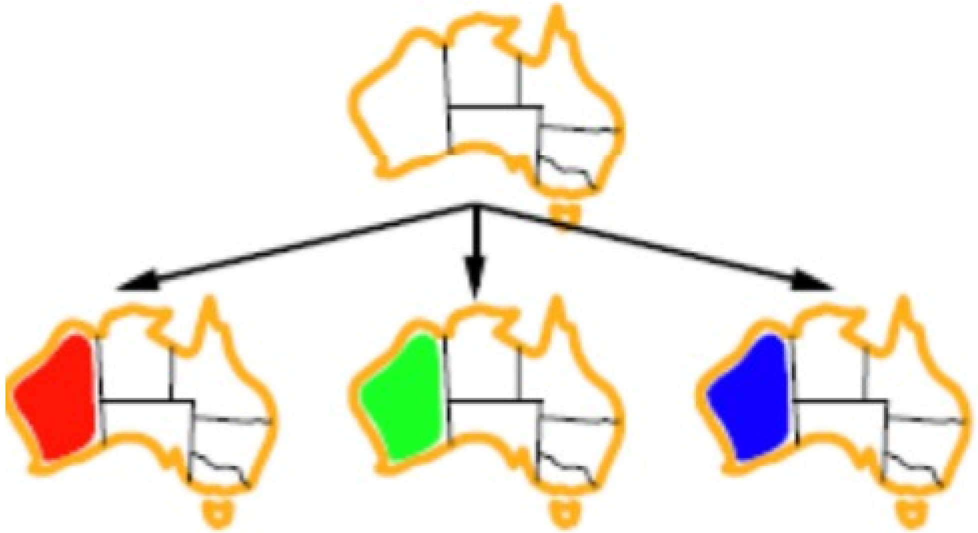
1) This is the same for all CSPs! 😆
2) Every solution appears at depth n with n variables

3) Path is irrelevant, so can also use complete-state formulation
4) $b = (n - \ell)d$ at depth ℓ, hence $n!d^n$ leaves!!!! ☹

BACKTRACKING EXAMPLE

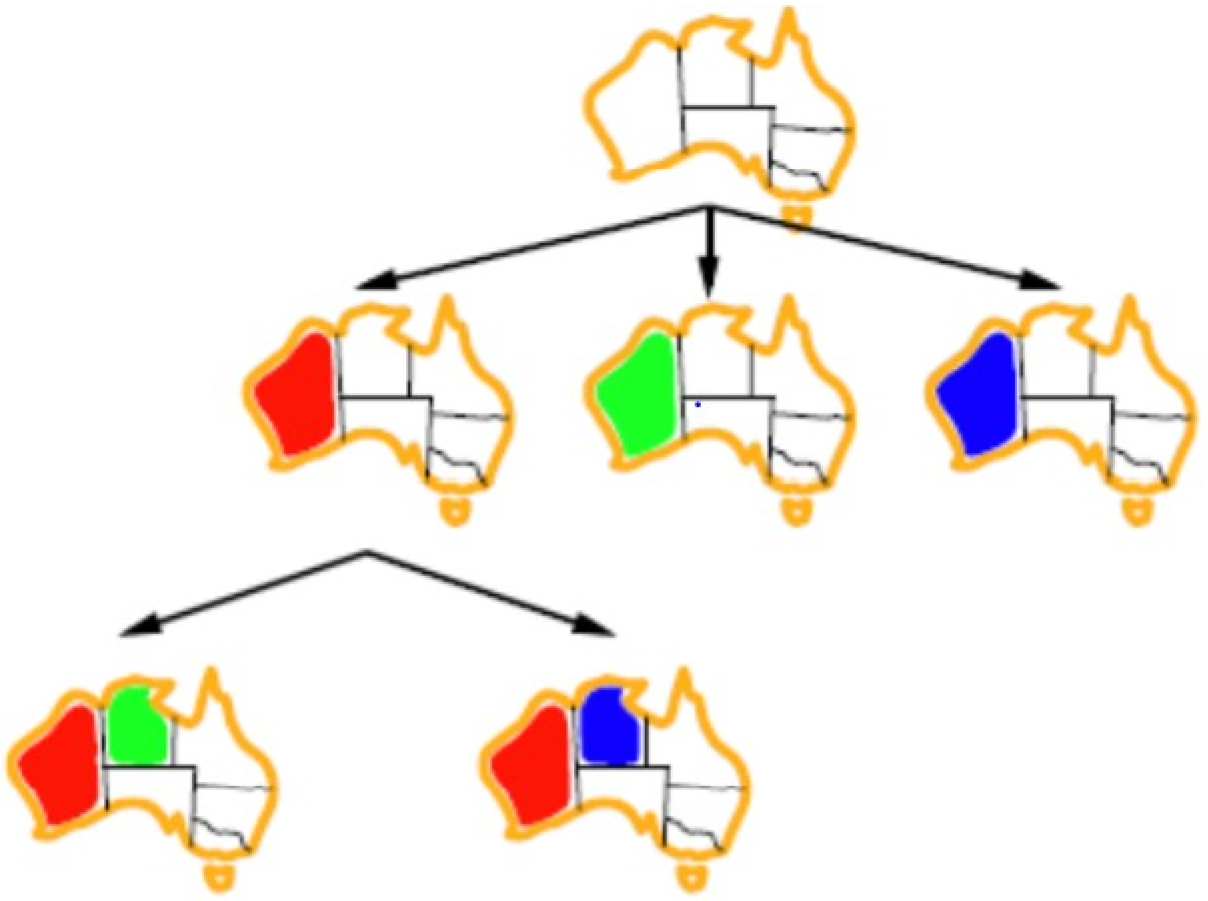
- Step 1:

- Step 2:

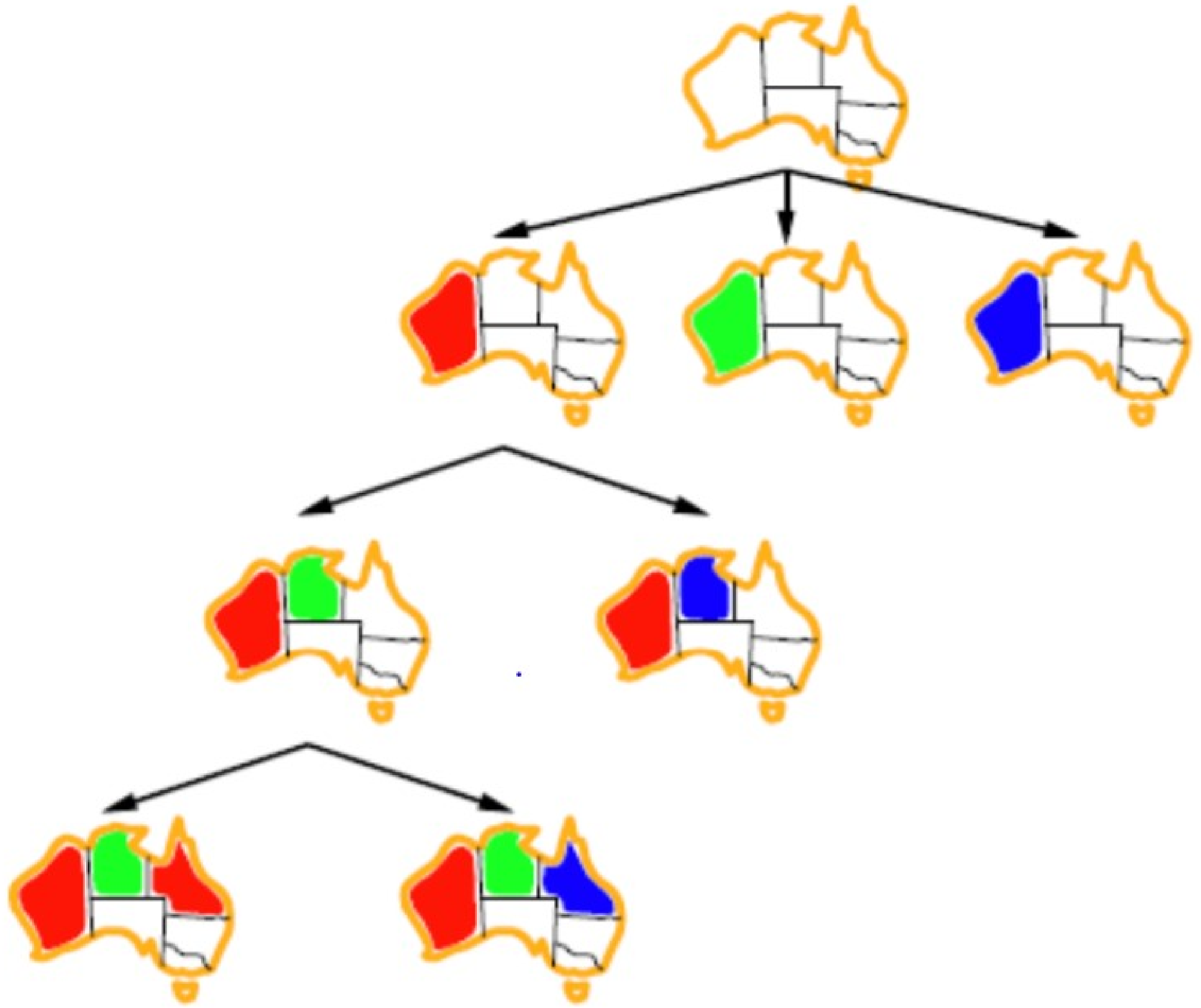


BACKTRACKING EXAMPLE

- Step 3:



BACKTRACKING EXAMPLE



- Step 4:

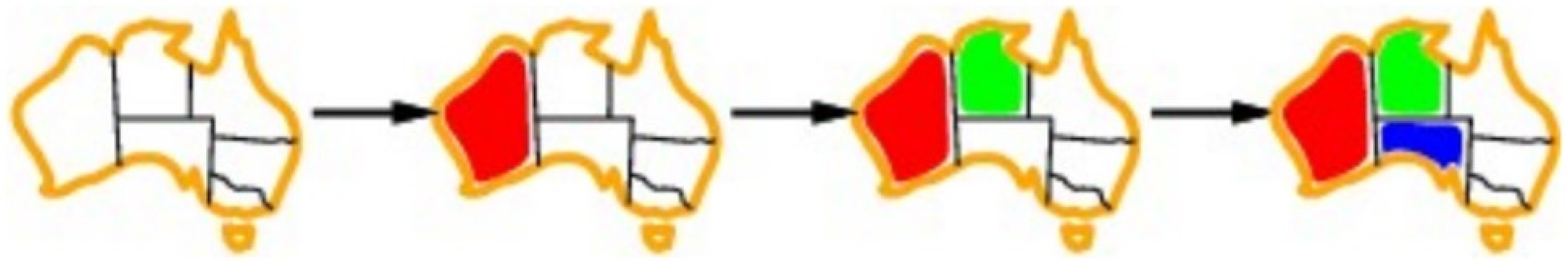
IMPROVING BACKTRACKING EFFICIENCY

- We can solve CSPs efficiently without domain-specific knowledge, addressing the following questions:

  - in what order should variables be assigned, values be tried?
  - what are the implications of the current variable assignments for the other unassigned variables?
  - when a path fails, can the search avoid repeating this failure?

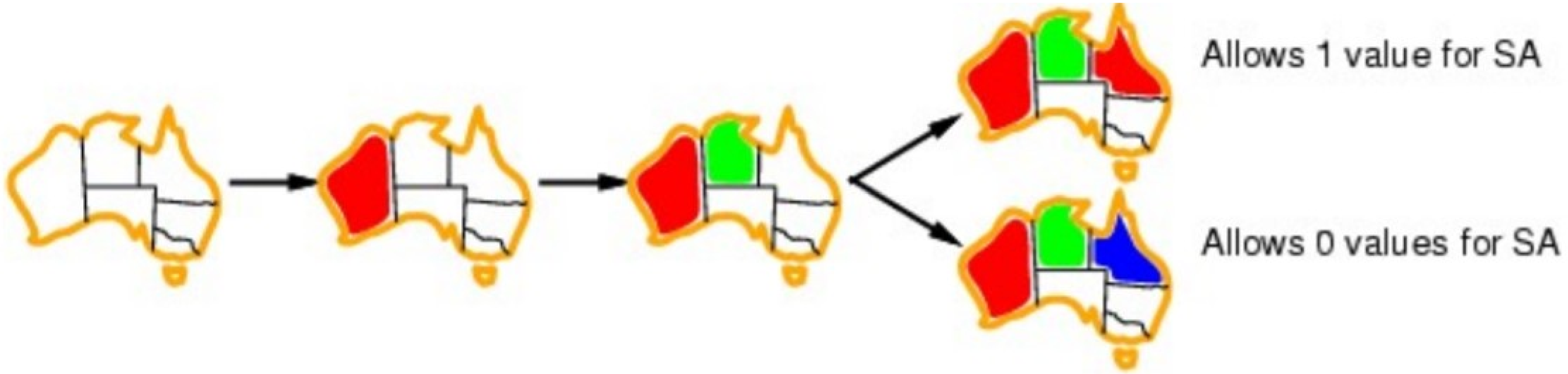
TECHNIQUES

- Minimum remaining values (MRV)
  - ➢ choose the variable with the fewest "legal" values
  - ➢ also called most constrained variable or fail-first heuristic
  - ➢ does it help in choosing the first variable?



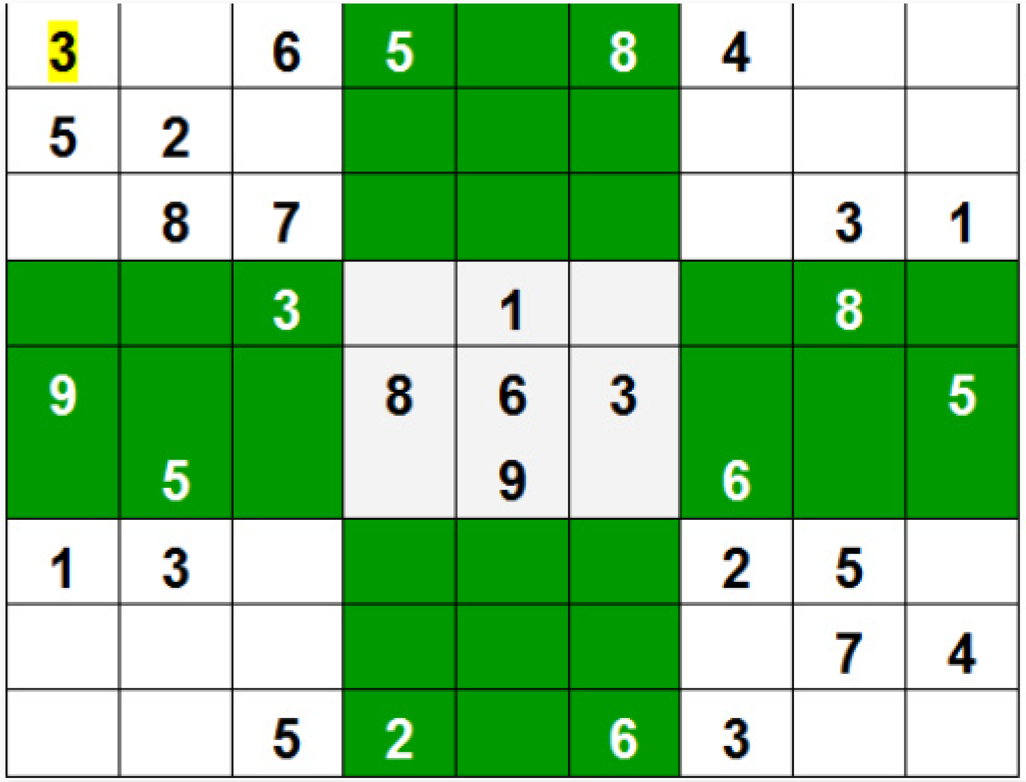
TECHNIQUES

- Most constraining variable:
  - ➢ selecting the variable that has the largest number of constraints on other unassigned variables
  - ➢ also called degree heuristics
  - ➢ the one that rules out the fewest values in the remaining variables



EXAMPLE

- For example, consider the SudoKu solving Problem, we try filling digits one by one.
- Whenever we find that current digit cannot lead to a solution, we remove it (backtrack) and try next digit.
- This is better than naive approach (generating all possible combinations of digits and then trying every combination one by one) as it drops a set of permutations whenever it backtracks.

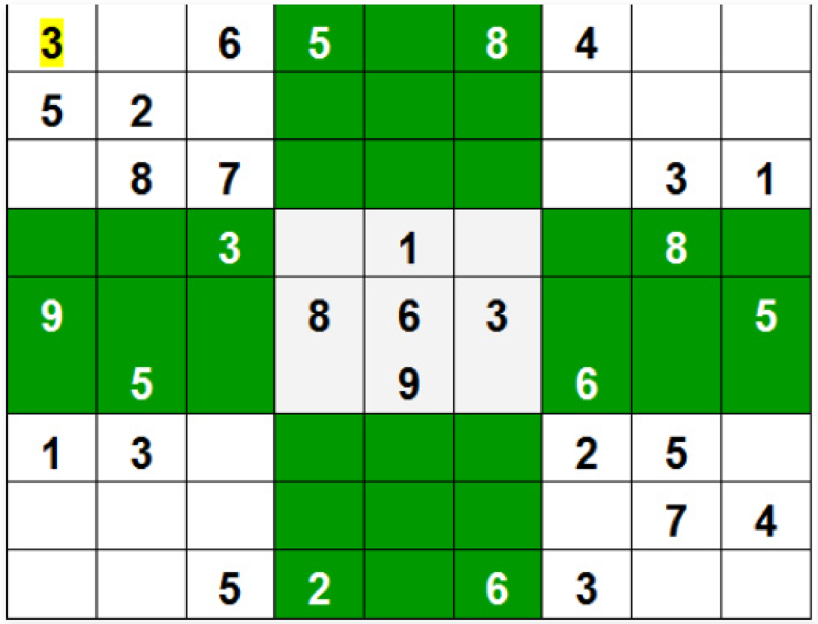


3		6	5		8	4		
5	2							
	8	7					3	1
		3		1			8	
9			8	6	3			5
	5			9		6		
1	3					2	5	
							7	4
		5	2		6	3		

SUDOKU| BACKTRAKING

- Given a partially filled 9×9 2D array ‘grid[9][9]’, the goal is to assign digits (from 1 to 9) to the empty cells so that every row, column, and subgrid of size 3×3 contains exactly one instance of the digits from 1 to 9.

```
Input:
grid = { {3, 0, 6, 5, 0, 8, 4, 0, 0},
         {5, 2, 0, 0, 0, 0, 0, 0, 0},
         {0, 8, 7, 0, 0, 0, 0, 3, 1},
         {0, 0, 3, 0, 1, 0, 0, 8, 0},
         {9, 0, 0, 8, 6, 3, 0, 0, 5},
         {0, 5, 0, 0, 9, 0, 6, 0, 0},
         {1, 3, 0, 0, 0, 0, 2, 5, 0},
         {0, 0, 0, 0, 0, 0, 0, 7, 4},
         {0, 0, 5, 2, 0, 6, 3, 0, 0} }
```



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

EXAMPLE

- Algorithm:

  1. Create a function that checks after assigning the current index the grid becomes unsafe or not. Keep Hashmap for a row, column and boxes. If any number has a frequency greater than 1 in the hashMap return false else return true; hashMap can be avoided by using loops.

  2. Create a recursive function that takes a grid.

  3. Check for any unassigned location. If present then assign a number from 1 to 9, check if assigning the number to current index makes the grid unsafe or not, if safe then recursively call the function for all safe cases from 0 to 9. if any recursive call returns true, end the loop and return true. If no recursive call returns true then return false.

  4. If there is no unassigned location then return true.

MODULE III: ROADMAP

- Constraint Satisfaction Problems:
  - ✓ Defining Constraint Satisfaction Problems,
  - ✓ Constraint Propagation:
    - Inference in CSPs,
    - Backtracking Search for CSPs,
    - Local Search for CSPs,
    - The Structure of Problems.

- Local Agents:
  - ✓ Knowledge-based Agents,
  - ✓ The Wumpus World, Logic,
  - ✓ Propositional Logic:
    - A Very Simple Logic,
    - Propositional Theorem Proving.

LOCAL SEARCH FOR CSPs

- Local search is an incomplete method for finding a solution to a problem.
- It is based on iteratively improving an assignment of the variables until all constraints are satisfied.
- In particular, local search algorithms typically modify the value of a variable in an assignment at each step. The new assignment is close to the previous one in the space of assignment, hence the name local search.
- All local search algorithms use a function that evaluates the quality of assignment, for example the number of constraints violated by the assignment. This amount is called the cost of the assignment.
- The aim of local search is that of finding an assignment of minimal cost, which is a solution if any exists.

TYPES OF LOCAL SEARCH FOR CSPs

- Greedy or non-randomized algorithms:

  ➢ These algorithms proceed by changing the current assignment by always trying to decrease (or at least, non-increase) its cost.

  ➢ The main problem of these algorithms is the possible presence of plateaus, which are regions of the space of assignments where no local move decreases cost.

- Randomized local search algorithms: Escape the plateaus by doing random moves.

GREEDY OR NON-RANDOMIZED ALGORITHMS

- Hill Climbing
- Constraint weighting or breakout method
- Tabu search

HILL CLIMBING

- The most basic form of local search is based on choosing the change that maximally decreases the cost of the solution. This means that it moves in a direction in which the cost function is optimized.
- It does not look at the previous states.
- This method, called hill climbing, proceeds as follows:

i. first, a random assignment is chosen;

ii. then, a value is changed so as to maximally improve the quality of the resulting assignment.

iii. If no solution has been found after a given number of changes, a new random assignment is selected.

iv. Hill climbing algorithms can only escape a plateau by doing changes that do not change the quality of the assignment.

v. As a result, they can be stuck in a plateau where the quality of assignment has a local maxima.

CONSTRAINT WEIGHTING OR BREAKOUT METHOD

- A method for escaping from a local minimum is that of using a weighted sum of violated constraints as a measure of cost, and changing some weights when no improving move is available.
- More precisely, if no change reduces the cost of the assignment, the algorithm increases the weight of constraints violated by the current assignment.
- This way, every move that would not otherwise change the cost of the solution decreases it.
- Moreover, the weight of constraints that remain violated for a large number of moves keeps increasing.
- Therefore, during a number of moves not satisfying a constraint, the cost of moves to assignments satisfying that constraint keeps increasing.

TABU SEARCH

- A drawback of hill climbing with moves that do not decrease cost is that it may cycle over assignments of the same cost.
- Tabu search overcomes this problem by maintaining a list of "forbidden" assignments, called the tabu list.
- In particular, the tabu list typically contains only the most recent changes. More precisely, it contains the last variable-value pair such that the variable has been recently assigned to the value.
- This list is updated every time the assignment is changed. If a variable is assigned to a value, the variable-value pair is added to the list, and the oldest pair is removed from it. This way, the list only contains the most recent assignments to a variable. If a variable-value pair is in the tabu list, then changing the current assignment by setting the variable to the value is forbidden.
- The algorithm can only choose the best move among the ones that are not forbidden. This way, it cannot cycle over the same solution unless the number of moves in this cycle is larger than the length of the tabu list.

RANDOMIZED LOCAL SEARCH ALGORITHMS

- A random walk algorithm sometimes moves like a greedy algorithm but sometimes moves randomly.
- It depends on a parameter p, which is a real number between 0 and 1.
- At every move, with probability p the algorithm proceeds like a greedy algorithm, trying to maximally decrease the cost of the assignment.
- With probability 1- p, however, the solution is changed in some other way, which involves some degree of randomness.
- There are two types of randomized local search algorithms:

  i. WalkSAT

  ii. Simulated annealing

WALKSAT

- The random move of WalkSAT is changing the value of a random variable of a random violated constraint.
- For propositional satisfiability of conjunctive normal form formulae, which is the original settings of this algorithm, every such a move changes the value of the variable from true to false or vice versa, and produce the satisfiability of the violated constraint.
- As for all random walk strategies, a random move is only done with a given probability, and a move maximally decreasing the cost is done otherwise.

SIMULATED ANNEALING

- The technique of simulated annealing is based on changing the probability of doing a random move over one that maximally decreasing the cost.
- In particular, the name originates from the strategy of decreasing the probability of doing random moves during the execution of the algorithm, thus virtually "freezing" the space of search.
- In particular, if the improvement of cost d of a move is negative (the move increases cost), this move is done with probability $e^{-d.T}$, where T is a real number. Since the probability of doing this move increases with T, this parameter is called the temperature.
- Simulated annealing decreases this temperature over time, thus allowing more random moves at the beginning and less after time.

MODULE III: ROADMAP

GITAM (DEEMED TO BE UNIVERSITY) VISAKHAPATNAM · HYDERABAD · BENGALURU

- Constraint Satisfaction Problems:
  - ✓ Defining Constraint Satisfaction Problems,
  - ✓ Constraint Propagation:
    - Inference in CSPs,
    - Backtracking Search for CSPs,
    - Local Search for CSPs,
    - The Structure of Problems.

- Local Agents:
  - ✓ Knowledge-based Agents,
  - ✓ The Wumpus World, Logic,
  - ✓ Propositional Logic:
    - A Very Simple Logic,
    - Propositional Theorem Proving.

THE STRUCTURE OF PROBLEMS

- 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.
  - ➢ It is represented by: Tree and Directed arc consistency (DAC)
- The structure in the values of variables
  - ➢ By introducing a symmetry-breaking constraint, we can break the value symmetry and reduce the search space by a factor of n!.
  - ➢ e.g. Consider the map-coloring problems with n colors, for every consistent solution, there is actually a set of n! solutions formed by permuting the color names.(value symmetry). On the Australia map, WA, NT and SA must all have different colors, so there are 3!=6 ways to assign. We can impose an arbitrary ordering constraint NT<SA<WA that requires the 3 values to be in alphabetical order. This constraint ensures that only one of the n! solution is possible: {NT=blue, SA=green, WA=red}.

References:

Text Book(s)

1. Stuart Russell, Peter Norvig, Artificial Intelligence: A Modern Approach, 3rd Edition, Pearson Publications, 2020

2. Dr. Nilakshi Jain, Artificial Intelligence : Making a System Intelligent, Wiley Publications,1st Edition,2019

Reference Book(s)

1. Elaine Rich, Kevin Knight and Shivashankar B. Nair, Artificial Intelligence, TMH Education Pvt. Ltd., 2008.

2. Dan W. Patterson, Introduction to Artificial Intelligence and Expert Systems, Pearson.

Internet:

https://www.javatpoint.com/artificial-intelligence-tutorial

https://www.geeksforgeeks.org/types-of-environments-in-ai/

https://www.tutorialspoint.com/artificial_intelligence/

etc