Introduction to AI

Lecture 6
Satisfying Constraints

Dr. Tamal Ghosh
Department of CSE
Adamas University

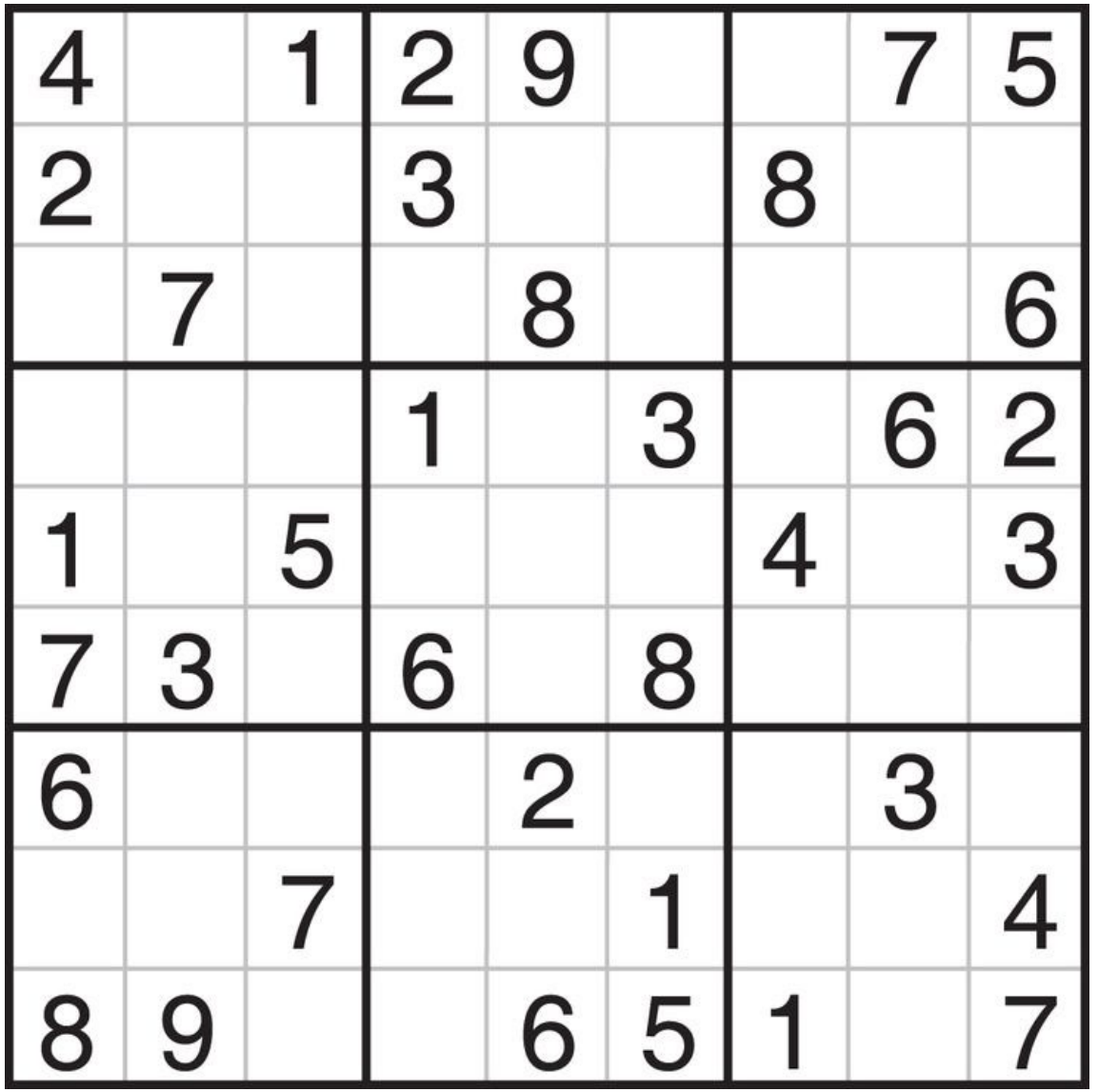
Constraints Satisfaction Problem (CSP)

Constraint Satisfaction problems are a class of problems where variables need to be assigned values while satisfying some conditions.

Constraints satisfaction problems have the following properties:

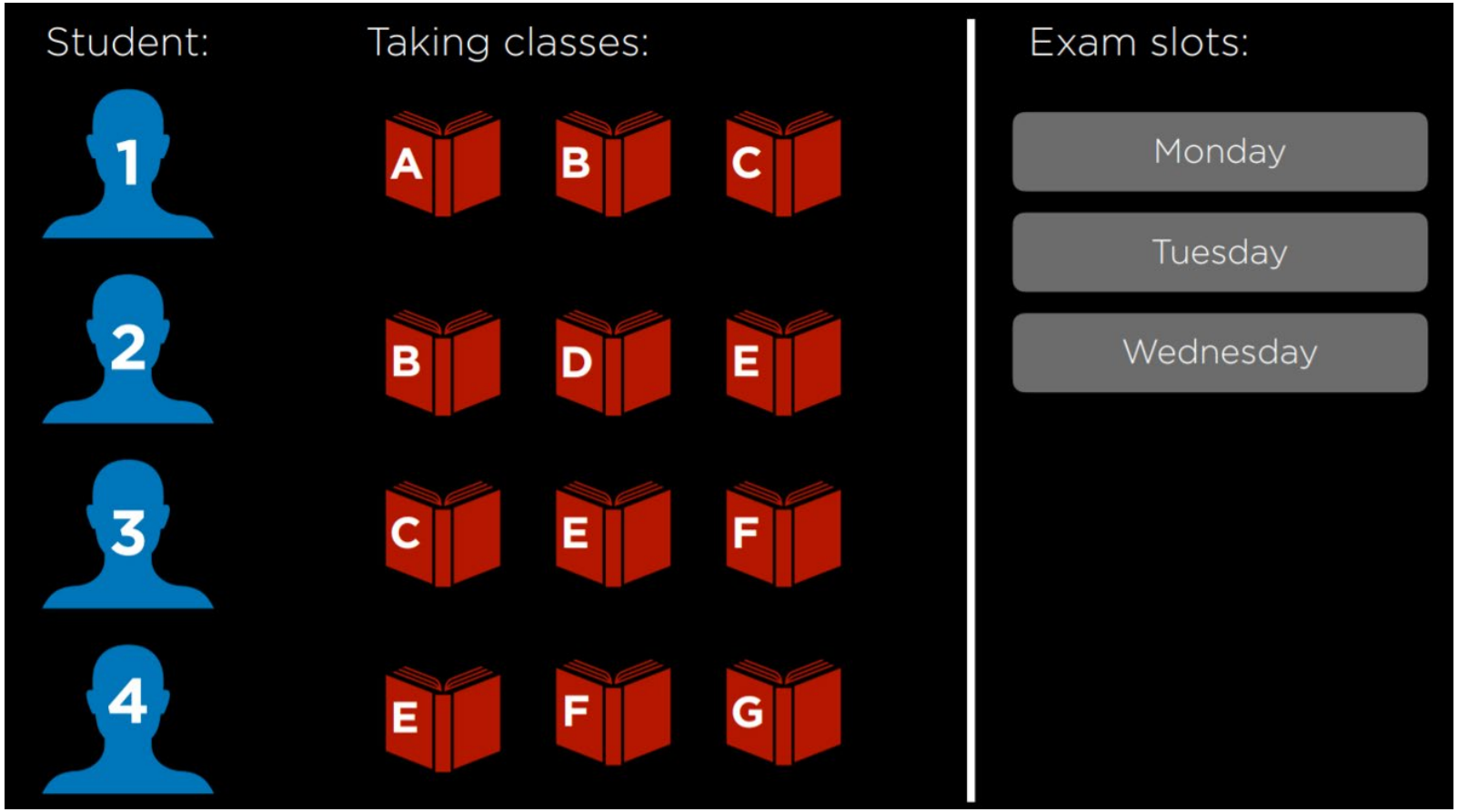
- Set of variables V (x_1, x_2, …, x_n)
- Set of domains for each variable {D_1, D_2, …, D_n}
- Set of constraints C
- A solution X = { a_i } i=1,…,n, for $a_i \in D_i$, is a complete assignment of variables in V such that every constraint in C is satisfied.
- $X \in D_1 \times \ldots \times D_n$ which is the Search Space for the CSP
- Sudoku can be represented as a constraint satisfaction problem, where each empty square is a variable, the domain is the numbers 1-9, and the constraints are the squares that can’t be equal to each other.

Sudoku



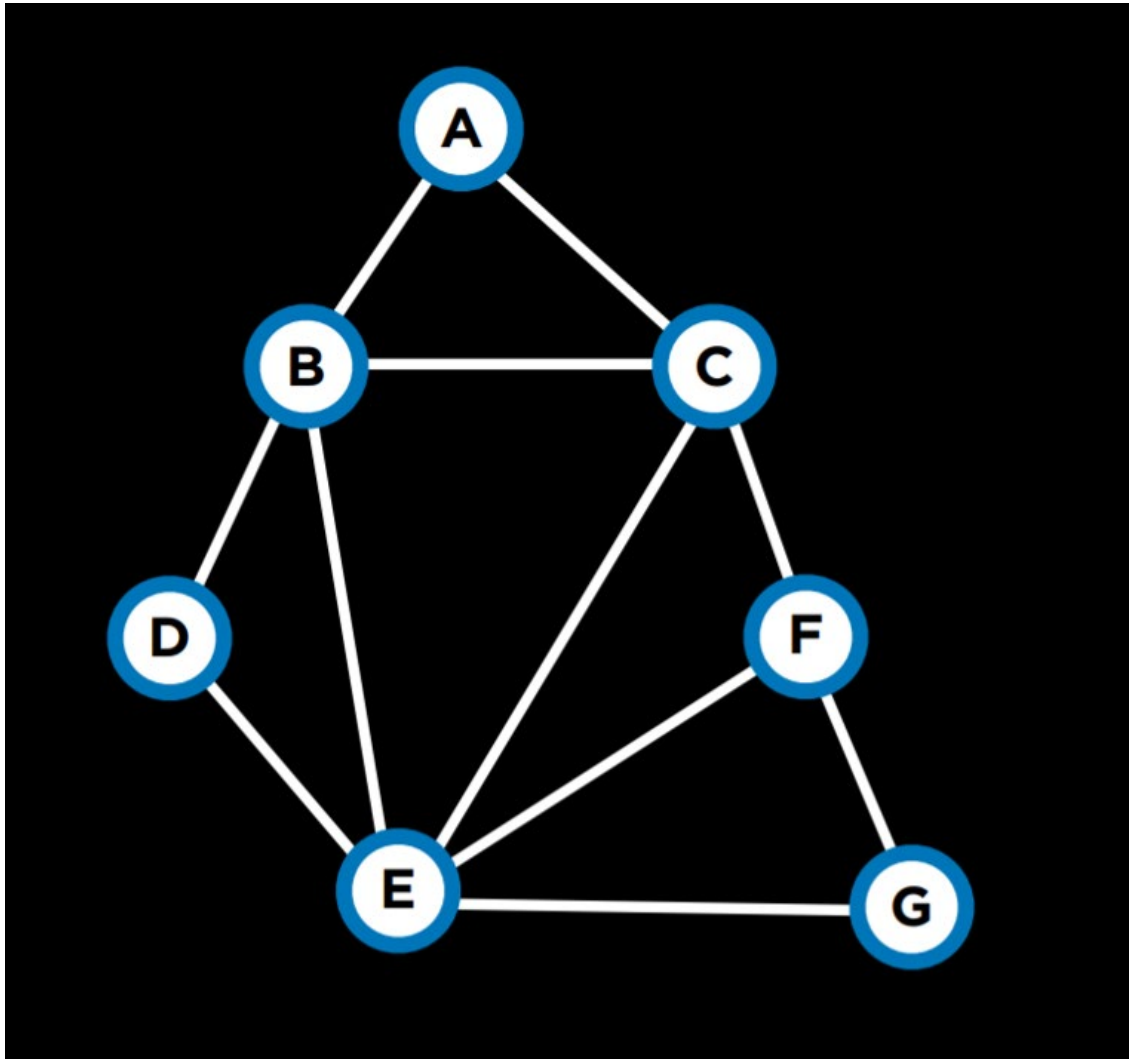
Example

- Consider another example. Each of students 1-4 is taking three courses from A, B, …, G. Each course needs to have an exam, and the possible days for exams are Monday, Tuesday, and Wednesday. However, the same student can't have two exams on the same day. In this case, the variables are the courses, the domain is the days, and the constraints are which courses can't be scheduled to have an exam on the same day because the same student is taking them. This can be visualized as follows:



CONTD…

- This problem can be solved using constraints that are represented as a graph. Each node on the graph is a course, and an edge is drawn between two courses if they can't be scheduled on the same day. In this case, the graph will look this:



Constraint Optimization Problem

The constraints of a CSP are usually represented by an expression involving the affected variables, e.g.

$$x_1 \neq x_2$$
$$2x_1 = 10x_2 + x_3$$
$$x_1 x_2 < x_3$$

- COP = CSP + Objective Function
- Objective function f assigns preferences among solutions to the CSP.
- Goal is to find the solution with the highest/lowest value of f.
- COP is harder than the CSP
- Requires completeness

Constraint Graph Representation

- Definition: a graph (V, E) is a set of vertices V and a set of edges E = $\{(v_i , v_j)\ldots\}$, $v_i , v_j \in V$ between vertices.
- Vertices are represented as nodes and edges are represented as lines between vertices.
- Graphs are suitable for representing only binary CSPs where the constraints are all of the form: $c_i (v_i)$ and $c_{ij} (v_i, v_j)$, for $v_i , v_j \in V$.

Formally, a constraint $C_{ijk...}$ between the variables $x_i, x_j, x_k, ...$ is any subset of the possible combinations of values of $x_i, x_j, x_k, ...$, i.e. $C_{ijk...} \subseteq D_i \times D_j \times D_k \times .....$ The subset is a set of tuples of values and specifies the combinations of values which the constraint allows. (Or you could equivalently define constraints by specifying the tuples which are *not* allowed.)

For example, if variable x has the domain $\{1, 2, 3\}$ and variable y has the domain $\{1, 2\}$ then any subset of $\{(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)\}$ is a valid constraint between x and y. The constraint $x = y$ would be represented by the subset $\{(1,1), (2,2)\}$.

Constraints types

If there are n variables in a problem, a constraint can affect any number of variables from 1 to n. The number of affected variables is the *arity* of the constraint. It is useful to distinguish two particular cases:

Unary constraints affect just one variable. The constraint can be used at the outset to remove any value which does not satisfy the constraint from the domain of the variable. For instance, if there is a constraint $x_1 \neq 1$, the value 1 can be removed from the domain of x_1, and the constraint will then be satisfied. Since unary constraints are dealt with by preprocessing the domains of the affected variable, they can be ignored thereafter.

Binary constraints affect two variables. Binary constraints play an important role in many of the algorithms we shall be using, so we shall meet them again, even though the constraints in real problems are often not binary.

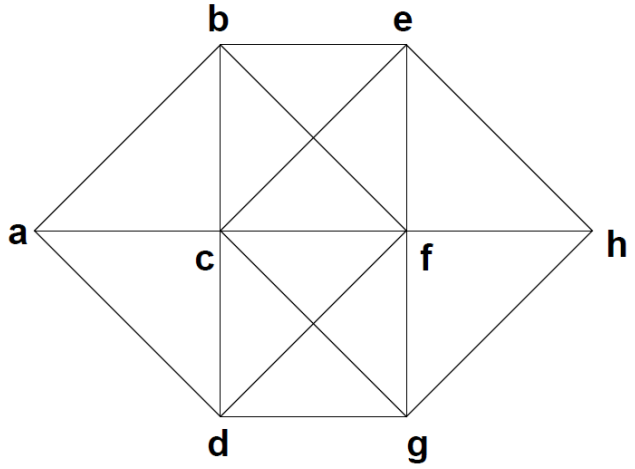
Is it hard to formulate CSP?

There are three basic questions to address in formulating a problem as a CSP:

- what entities in the problem should the variables represent?
- what are their possible values?
- what are the constraints?

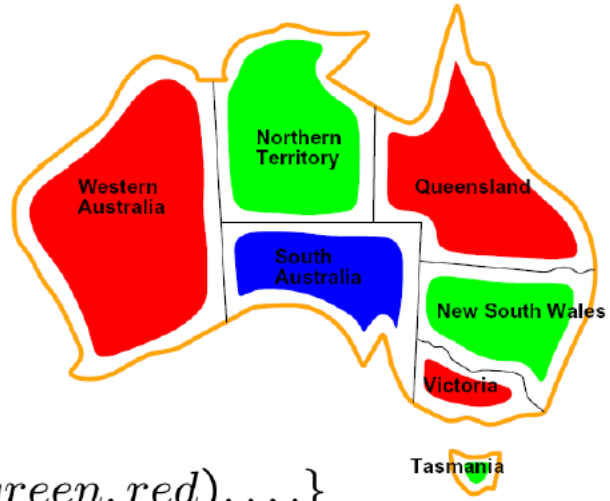
A number puzzle

A puzzle requires the numbers 1 to 8 to be placed at the positions marked `a` to `h` in the diagram, in such a way that wherever there is a line joining two of the positions, the corresponding numbers must differ by at least 2, and each number must be used exactly once.[2]



Example: Map coloring

- Variables: $WA,\ NT,\ Q,\ NSW,\ V,\ SA,\ T$
- Domain: $D = \{red, green, blue\}$
- Constraints: adjacent regions must have different colors



$$WA \neq NT$$

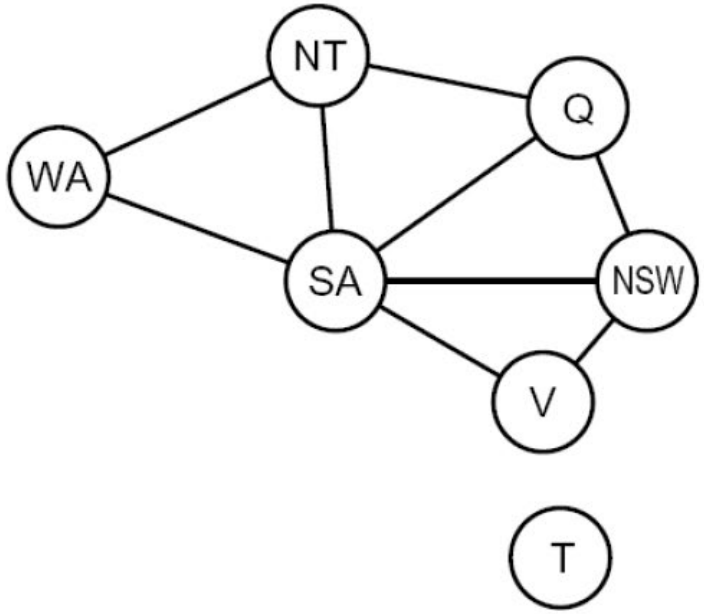
$$(WA, NT) \in \{(red, green), (red, blue), (green, red), \ldots\}$$

- Solutions are assignments satisfying all constraints, e.g.:

$$\{WA = red, NT = green, Q = red,$$
$$NSW = green, V = red, SA = blue, T = green\}$$

Constraints graph

- Binary CSP: each constraint relates (at most) two variables
- Binary constraint graph: nodes are variables, arcs show constraints
- General-purpose CSP algorithms use the graph structure to speed up search. E.g., Tasmania is an independent subproblem!



Real World Problems

- Assignment problems: e.g., who teaches what class
- Timetabling problems: e.g., which class is offered when and where?
- Hardware configuration
- Transportation scheduling
- Factory scheduling
- Floorplanning
- Fault diagnosis
- … lots more!

- Many real-world problems involve real-valued variables…

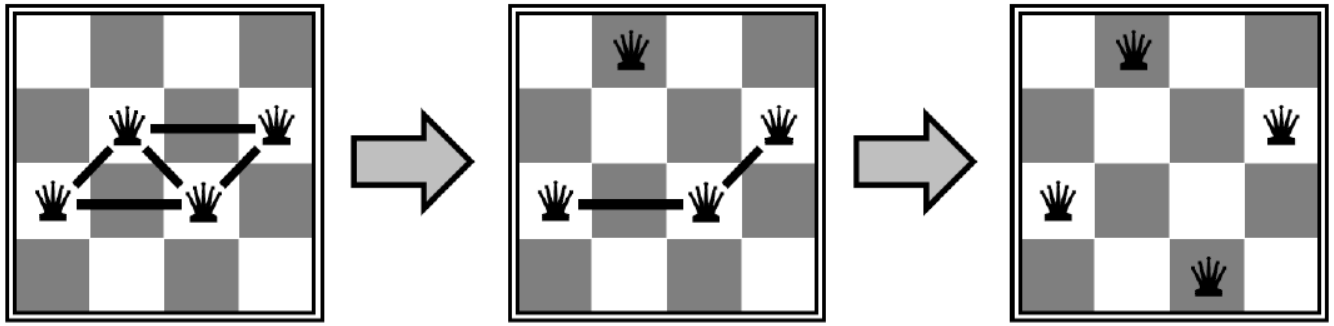
Standard Search Formulation

- Standard search formulation of CSPs (incremental)
- Let's start with the straightforward, dumb approach, then fix it
- States are defined by the values assigned so far
  - Initial state: the empty assignment, {}
  - Successor function: assign a value to an unassigned variable
  - Goal test: the current assignment is complete and satisfies all constraints
- Simplest CSP ever: two bits, constrained to be equal

Local Search Method

- Local search methods typically work with "complete" states, i.e., all variables assigned
- To apply to CSPs:
  - Start with some assignment with unsatisfied constraints
  - Operators *reassign* variable values
  - No fringe! Live on the edge.
- Variable selection: randomly select any conflicted variable
- Value selection by min-conflicts heuristic:
  - Choose value that violates the fewest constraints
  - I.e., hill climb with h(n) = total number of violated constraints

Example: n-Queens



- States: 4 queens in 4 columns ($4^4 = 256$ states)
- Operators: move queen in column
- Goal test: no attacks, i.e., no two queens on same row, same column or same diagonal
- Evaluation: c(n) = number of attacks