Constraint Programming

Dr Rong Qu
School of Computer Science
Automated Scheduling, Optimization and Planning (ASAP) Group
University of Nottingham, Nottingham, NG8 1BB, UK
rxq@cs.nott.ac.uk

CP Techniques & Search Tree
CSP – Formal Definition

- A constraint satisfaction problem (CSP) consists of
  - a set of *variables* \( \{x_1, x_2, \ldots, x_i\} \);
  - a finite set of *domain* \( D \) (possible values) for each variable;
  - a set of *constraints* \( C \) restricting the values that the variables can simultaneously take

- CSP is to assign values to variable so that all constraints are satisfied
CSP – Approaches

- Constraint programming techniques
  - Systematic search on search trees + constraint based techniques
  - Based on basic search techniques on search tree
    - Part of content of “Introduction to AI” G51IAI

- Artificial intelligence search algorithms
  - “move” within search space to promising solutions
  - Content of “AI Methods” G52AIM
CSP – Approaches

Four criteria to evaluate how good search techniques are

- Completeness
- Time complexity
- Space complexity
- Optimality
CSP – AI Search algorithms

- Not complete; Not guarantee to find (optimal) solution
- Difficult to evaluate time and space complexity
- Work well on large scale problems
  - when good enough rather than optimal solutions are needed

- Based on *search space* of solutions by neighborhood “moves”
  - Partially explored by searching algorithms

1. Start (usually) from complete solutions
2. Modify the solutions by changing and accepting the new solution
3. Return the best/satisfactory solution
CSP – AI Search algorithms

- **Search space**
  - all possible solutions approachable by neighborhood moves

- **Neighborhood moves**
  - Operators that modify values of certain elements in complete solutions

![Graph showing search space and iterations](image)
CSP – AI Search algorithms

- Simulated Annealing
  - accept worse solutions by probability, which is higher at the beginning and lower at the end of the search

- Tabu Search
  - use memory to remember previous moves, so as not go back revisit visited solutions

- Genetic Algorithms
  - simulate the evolution process by crossover and mutation within populations of individuals (solutions)

- Ant Algorithms
  - use pheromone as guidance of search towards promising and better solutions
CSP – Constraint programming

- It is complete
- Guarantee to find the (optimal) solution(s) if one exists
- Computational time and space may be expensive for large scale problems

- Based on systematically organised search tree
  - All possible solutions systematically organised and explored
CSP – Approaches

- We’ll concentrate on CP in this course
  - Theory of CP (lectures)
  - Practical of CP (case study and demos in ILOG Solver using OPL)
  - Research on constraint based scheduling

- Slight touch on AI Search algorithms to make the context complete

- Recent research
  - hybridise AI search algorithms with constraint based techniques
Constraint Programming

Constraint programming techniques solve CSPs by using the combination of

- Systematic search
  - Generate and Test
  - Search tree
  - Backtracking

and

- Constraint satisfaction techniques
  - Constraint propagation
  - Search strategies
Systematic Search

- **Search space**
  - All possible states (solutions) which a search could arrive at

- Systematically search through a search space of all possible assignment of values to variables
  - It is complete
    - Either a solution is guaranteed
    - Or no solution exist
Systematic Search

Generate-and-test (GT)
- enumerate all possible combinations of values for variables one by one and see if they satisfy all constraints

REPEAT
- Select the next variable
- Assign a value to the variable
- If current assignment lead to a failure (dead-end: no values are consistent with previous values)
  - Backtrack (replace the value assigned for previous variable with a new value)

UNTIL a (no) complete solution is found
Systematic Search – Search Tree

- Search Tree represent the state of search
  - Node: partial solution
  - Branch: possible assignment of values to variables
  - Labelling: assign one value to a variable (taking one branch)
  - Dead-end: no further values can be assigned to the variable

We can now find the solution by searching on the search tree.
How is the size of CSP measured
- Number of variables
- Size of domains for the variables
- Number of constraints

No. of leaves in the search tree \( n!^d n \)
Systematic Search – Search Tree

Properties of CSP’s search tree

- The depth of the tree is fixed
  - Solution will always be at the $n^{th}$ level
- The size of domains
  - Branching factor – average number of branches
- Sub-trees are similar
  - This is useful for learning during the search
- The size of the search space for a problem is finite
  - Total number of leaves is fixed for a problem
  - However the internal nodes are different

Try the graph coloring problem search tree with different orderings of variables labelled.
Systematic Search – Search Tree

- Depth first search
Systematic Search – Backtracking

- Backtracking
Systematic Search – Backtracking

- Revising past labels
  - Label one variable at a time
  - If current value is incompatible
    - Take an alternative value
  - If all values are tried
    - Un-assign the last variable

- Until
  - All variable labelled
  - No more label to backtrack to

Systematic search with backtracking
Tsang. (1996) Foundations of Constraint Satisfaction
Systematic Search – Backtracking

- Pure backtracking
  - is very inefficient
  - complexity is exponential
  - may explore branches which likely lead to infeasible solutions (dead-ends)

- Thrashing: repeat the same failed assignment
- Redundant: conflict values of variables not remembered
- Detection of conflict at later stage: after exploring large number of branches
Constraint Satisfaction Techniques

- **Aim at**
  - Avoid as much as possible backtrackings
  - Speed up the search

- **Constraint propagation**
  - Consistency enforcing
    - Arc consistency, path consistency

- **Search strategies**
  - Back and forward checking
  - Variable and Value ordering
  - Branch & bound (B&B)
    - Constraint optimisation techniques