15-381: AI
CSPs – Local Search - Structure

Fall 2009

Manuela Veloso

Chapter 5, Russell and Norvig

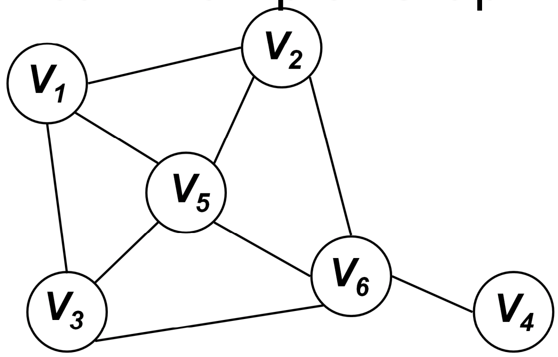
Thanks to all past 381 instructors, and
http://www.cs.cmu.edu/~awm/tutorials

Carnegie Mellon

Outline

- CSPs:
  - Definitions, DFS search, DFS with backtracking, Forward chaining, Constraint propagation, heuristics: variable and value ordering
- Local Search:
  - Hill climbing, stochastic hill climbing
- This lecture:
  - Local search for CSPs
  - Problem structure in CSPs

Canonical Example: Graph Coloring



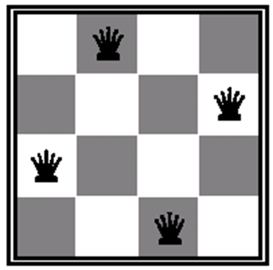
- Consider N nodes in a graph
- Assign values $V_1,..,V_N$ to each of the N nodes
- The values are taken in $\{R,G,B\}$
- Constraints: If there is an edge between i and j, then V_i must be different from V_j

Note that in CSPs, the path doesn't matter, only the solution.

Important things in a graph coloring problem: values (colors), variables (nodes), and the topology of the graph (constraints).

Local Search Techniques for CSP

N-Queens



Already seen: Hill climbing for N-Queens

Eval is # of pairs attacking queens.

Neighbors are states where one queen has been moved.

Hill climbing will try the move that leaves the fewest remaining conflicts. It is a greedy algorithm. Therefore hill climbing doesn't backtrack, not even for random search between ties. However, you may not want to keep track because a) it takes memory, b) if there are only global maxima/minima, or c) we're likely to revisit states.

Local Search for CSP

State = assignment of values to all the variables

V_1	V_2	V_3	V_4	V_5	V_6
a	*b*	*c*	*d*	*e*	*f*

Move = Change one variable

V_1	V_2	V_3	V_4	V_5	V_6
a	*b*	*c'*	*d*	*e*	*f*

Evaluation = number of conflicts (non-satisfied constraints) between variables

Generalize hill climbing algorithm that we used on N-Queens for all CSP's.

Other methods (DFS, forward search, constraint propagation) did not assign all variables up front.

Local Search for CSPs

- During search:
  - States have unsatisfied constraints
  - Successors mean to reassign **variable values**
- Variable selection
  - randomly select any conflicted variable
- Value selection
  - min-conflicts heuristic

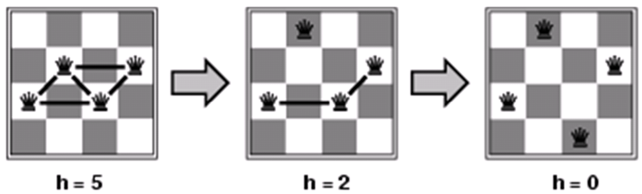
Min-conflicts heuristic - Select variable at random, and then give it the value that results in the fewest conflicts.

Min-Conflicts Algorithm

- Start with a **complete assignment** of variables
- Repeat until a solution is found or maximum number of iterations is reached:
  - Select a variable $\boldsymbol{V_i}$ *randomly* among the variables *in conflict*
  - Set $\boldsymbol{V_i}$ to the value that *minimizes* the number of constraints violated

Example: 4-Queens

- States: 4 queens in 4 columns (4^4 = 256 states)
- Actions: move queen in column
- Goal test: no attacks
- Evaluation: *h(n)* = number of attacks (pairs of queens)



- Given random initial state, can solve *n*-queens in almost constant time for arbitrary *n* with high probability (e.g., *n* = 10,000,000)

If you don't choose which queen to move randomly, then it is easy to get stuck in a local minimum

	USA (4 coloring)	N-Queens (1<N<=50)	Zebra
DFS Backtracking	$> 10^6$	$> 40\ 10^6$	$3.9\ 10^6$
+ MRV	$> 10^6$	$13.5\ 10^6$	1,000
Forward Checking	2,000	$> 40\ 10^6$	35,000
+ MRV	60	817,000	500
Min-Conflicts	64	4,000	2,000

(Data from Russell & Norvig)

(Zebra is a complicated murder mystery-like problem.)

This is why AI is awesome. Because simple ideas produce disturbingly large improvements.

MRV is Minimum Remaining Value

Discussion

- N-queens is easy for local search
  - Solutions are densely distributed
- Advantages of local search applied to CSPs
  - Online settings when the problem changes
    - Weekly's airline schedule; flights, personnel assignments
  - Local search from current is faster than backtracking
  - Backtracking could find a solution with many changes from the current schedule

Deterministic bias is bad…it is generally much better to choose randomly when choosing which queen to move.

Outline

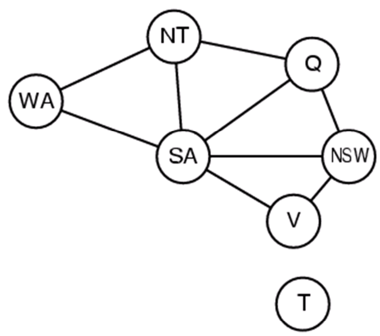
- Last Monday – CSPs:
  - Definitions, DFS search, DFS with backtracking, Forward chaining, Constraint propagation, heuristics: variable and value ordering
- Last lecture – Local Search:
  - Hill climbing, stochastic hill climbing
- This lecture:
  - Local search for CSPs
  - Problem structure in CSPs

With local search each state is a complete assignment.

How do you generate initial assignment? It is domain specific, though random is generally pretty good.

Constraint Graph

- Observations about *structure* of the graph
  - Components, independence, connectivity…..



T is not connected
If just SA were not there….

T can be anything

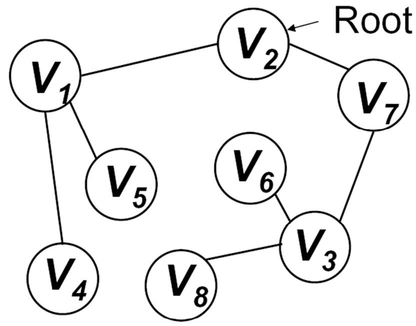
Everyone except T has to be different than SA

Independence

- Why important?
- Suppose each component has c variables, each takes d values, from a total of n variables; then n/c subproblems, with time d^c therefore O(d^c * n/c) linear in n, versus O(d^n).

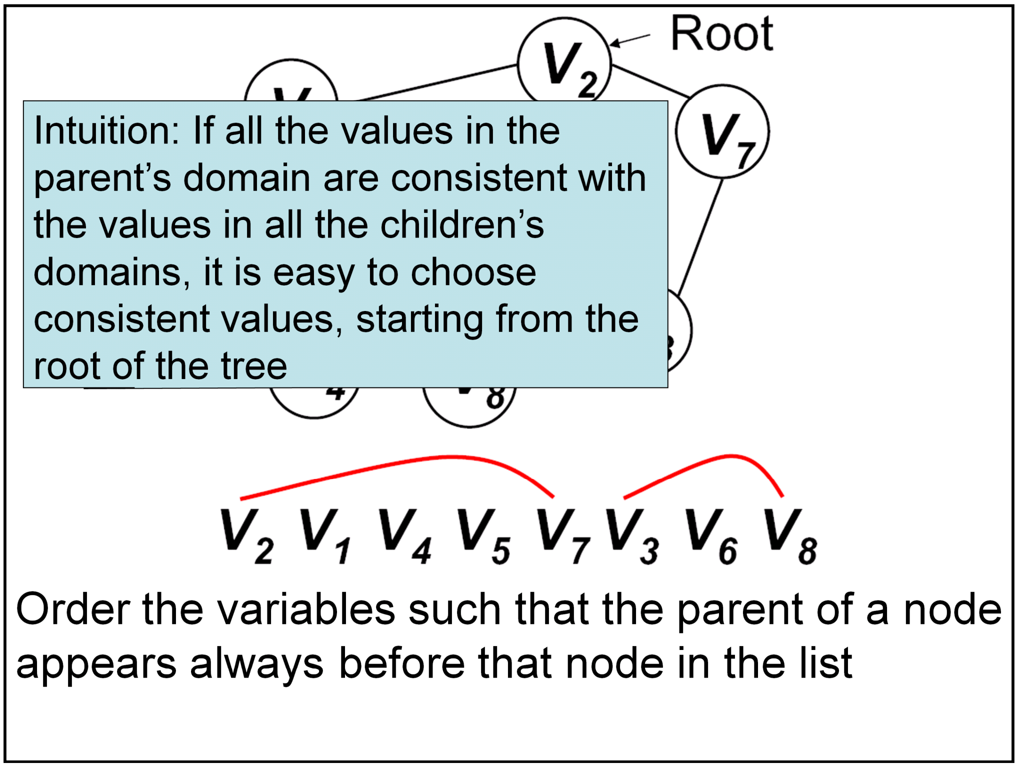
- n=80 boolean CSP, four with c=20, worst case from a lifetime down to less than 1s.

Simplest Case: Constraint Trees



- Any two variables are connected by at most one path

- Complexity of solving tree-structured CSPs?
  - Time **linear** in the number of variables.



You can really choose any node as the root of the tree.

Constraint Tree Algorithm

1. Up from leaves to

Visit each variable once: *N*

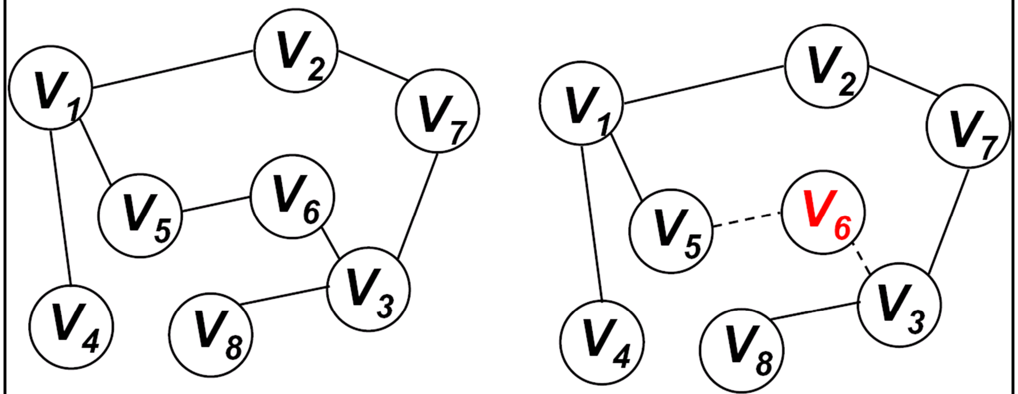
   – For every variable $\mathbf{V}_i$, starting at the leaves:
   – $\mathbf{V}_j = parent(\mathbf{V}_i)$
   – Remove all the values *x* in $D(\mathbf{V}_j)$ for which there is no consistent value in $D($

Worst case: Need to check all pais of values: d^2

2. Down from root
   – Assign a value to the root of the tree
   – For every variable $\mathbf{V}_i$:
      • Choose a value *x* in $D(\mathbf{V}_i)$ consi value assigned to $parent(\mathbf{V}_i)$

Total time: $O(N\,d^2)$

Almost Tree

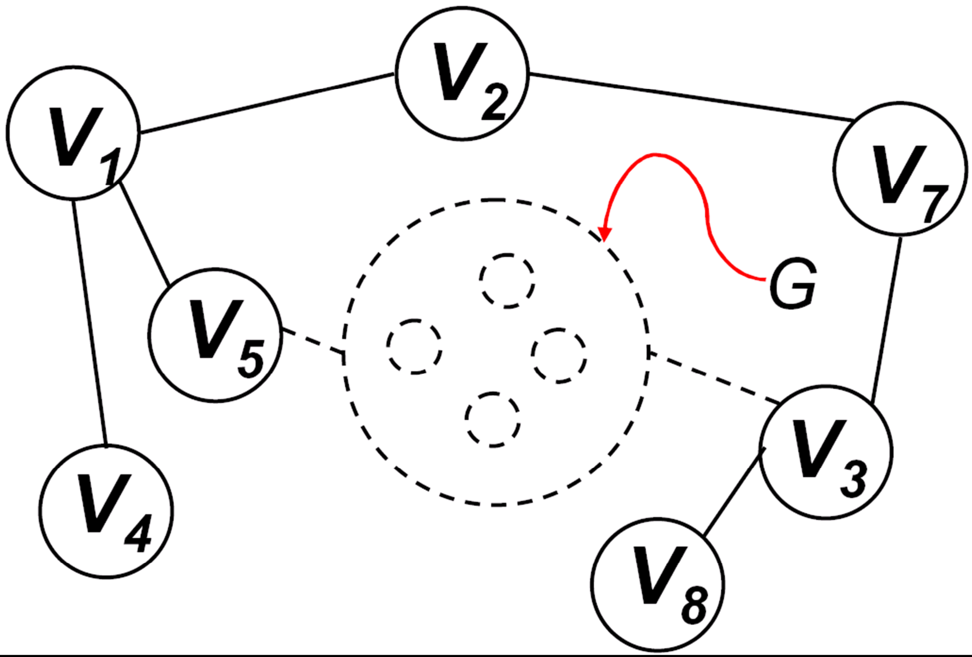


- The constraint graph becomes a tree once a value is chosen for V_6
- We don't know which value to choose → Try all possible values

After we give V6 a color, the graph can be treated as a tree, where V5 and V3 start off with one fewer acceptable colors.

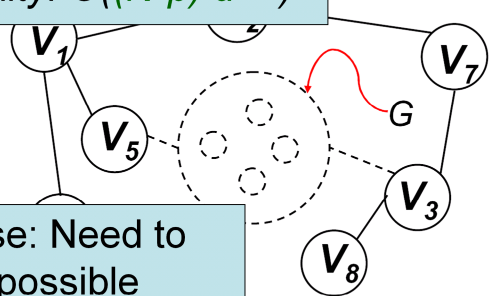
Here the complexity of the tree is (N-1 * d^2), done d times, so we have (N-1)*d^3

More General Case



For the HWs/midterm: you should think about these structural ideas, and how the algorithms are impacted by structure.

Complexity: O((N-p) d^{p+2})



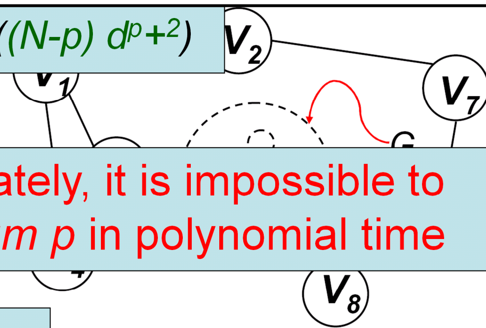
Worst case: Need to check all possible assignments in G → d^p

- … d group G of p variables transforms t… aph into a tree problem that can be solve… ficient…

Tree algorithm → $(N-p)\ d^2$

- We don't kn… how to set … ariables in G:
  - For every p…ssible con…ent assignment of values to variables in G:
    - Apply the tree algorithm to the rest of the variables

Complexity: $O((N-p)\, d^{p+2})$



Note: Unfortunately, it is impossible to find the *minimum p* in polynomial time

Worst case: Need to check all possible assignments in G → d^p

…nected group *G* of *p* …rms the graph into a tree problem … be solved efficiently.

- We don't … how to set the variables in *G*:
  - For every possible consistent assignment of values to variables in *G*:
    - Apply the tree algorithm to the rest of the variables

Tree algorithm → $(N-p)\, d^2$

CSPs – Summary

- Definitions
- Standard DFS search
- Improvements
  - Backtracking
  - Forward checking
  - Constraint propagation
- Heuristics:
  - Variable ordering
  - Value ordering
- Examples
- Local search for CSP problems
- Problem structure in CSPs

What you need to know