ANNOUNCEMENTS

- REMEMBER LECTURE ON TUESDAY!

- EXAM ON OCTOBER 18
  - OPEN BOOK
  - ALL MATERIAL COVERED IN LECTURES
  - REQUIRED READINGS

- WILL MOST PROBABLY NOT COVER MATERIAL ON PLANNING
Today’s Lecture

- Another Form of Local Search
  - Repair/Debugging in Constraint Satisfaction Problems
    - GSAT

- A Systematic Approach to Constraint Satisfaction Problems
  - Simple Backtracking Search
Constraint Satisfaction Problems (CSP)

- A set of **variables** $X_1...X_n$, and a set of **constraints** $C_1...C_m$. Each variable $X_i$ has a **domain** $D_i$ of possible **values**.
- A **solution** to a CSP: a complete assignment to all variables that satisfies all the constraints.
- Representation of constraints as predicates.
- Visualizing a CSP as a **constraint graph**.
Example: Map coloring

Constraint graph: nodes are variables, arcs show constraints.

Variables: $WA$, $NT$, $Q$, $NSW$, $V$, $SA$, $T$

Domains: $D_i = \{\text{red, green, blue}\}$

Constraints: adjacent regions must have different colors

- e.g., $WA \neq NT$ (if the language allows this), or
- $(WA, NT) \in \{(\text{red, green}), (\text{red, blue}), (\text{green, red}), (\text{green, blue}), \ldots\}$
Solutions are assignments satisfying all constraints, e.g.,
\[ \{ WA = \text{red}, NT = \text{green}, Q = \text{red}, NSW = \text{green}, V = \text{red}, SA = \text{blue}, T = \text{green} \} \]
Example 3: N queens

- What are the variables? domains? constraints?
8 queens

- 8 variables $X_i$, $i = 1$ to $8$; for each column
- Domain for each variable $\{1, 2, \ldots, 8\}$
- Constraints are:
  - $X_i \neq X_j$ for all $j = 1$ to $8$, $j \neq i$; not on same row
  - $|X_i - X_j| \neq |i - j|$ for all $j = 1$ to $8$, $j \neq i$; not on diagonal
  - Note that all constraints involve 2 variables
- Generate-and-test with no redundancies requires “only” $N^N$ combinations…
Task scheduling

- T1 must be done during T3
- T2 must be achieved before T1 starts
- T2 must overlap with T3
- T4 must start after T1 is complete

• What are the variables? domains? constraints?
Non-Binary Constraints

\[
\begin{align*}
\text{TWO} & \quad \text{+ TWO} \\
\text{FOUR} & \quad \\
\text{\quad O + O = R + 10 \cdot X_1} \\
\text{\quad X_1 + W + W = U + 10 \cdot X_2} \\
\text{\quad X_2 + T + T = O + 10 \cdot X_3} \\
\text{\quad X_3 = F} \\
\text{\quad \text{alldiff}(F, T, U, W, R, O)} \\
\text{\quad \text{Between0–9}(F, T, U, W, R, O)} \\
\text{\quad \text{Between0–1}(X_1, X_2, X_3)} \\
\end{align*}
\]

3 or more variables constraints
Constraint optimization

- Representing preferences versus absolute constraints.
  - Weighted by constraints violated/satisfied
- Constraint optimization is generally more complicated.
- Can also be solved using local search techniques.
- Hard to find optimal solutions.
Local search for CSPs: Heuristic Repair

- Start state is some assignment of values to variables that may violate some constraints.
  - Create a complete but inconsistent assignment
- Successor state: change value of one variable.
- Use **heuristic repair** methods to reduce the number of conflicts (iterative improvement).
  - The min-conflicts heuristic: choose a value for a variable that minimizes the number of remaining conflicts.
  - Hill climbing on the number of violated constraints
- Repair constraint violations until a consistent assignment is achieved.
- Can solve the *million*-queens problem in an average of 50 steps!
Heuristic Repair Algorithm

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

local variables: current, a complete assignment
                 var, a variable
                 value, a value for a variable


current ← an initial complete assignment for csp
for i = 1 to max-steps do
    var ← a randomly chosen, conflicted variable from VARIABLES[csp]
    value ← the value v for var that minimizes CONFLICTS(var, v, current, csp)
    set var=value in current
if current is a solution for csp then return current
end

return failure
N-Queens Heuristic Repair

- Pre-processing phase to generate initial assignment
  - Greedy algorithm that iterates through rows placing each queen on the column where it conflicts with the fewest previously placed queens

- Repair phase
  - Select (randomly) a queen in a specific row that is in conflict and moves it to the column (within the same row) where it conflicts with the fewest other queens
Example of min-conflicts: N-Queens Problem

A two-step solution of an 8-queens problem. The number of remaining conflicts for each new position of the selected queen is shown. Algorithm moves the queen to the min-conflict square, breaking ties randomly.
SAT - Satisfiability Problem

Given a propositional sentence, determine if it is satisfiable, and if it is, show which propositions have to be true to make the sentence true. 3SAT is the problem of finding a satisfying truth assignment for a sentence in a special format.

*Why are we interested in this representational framework?*
Definition of 3SAT

- A literal is a proposition symbol or its negation (e.g., $P$ or $\neg P$).

- A clause is a disjunction of literals; a 3-clause is a disjunction of exactly 3 literals (e.g., $P \lor Q \lor \neg R$).

- A sentence in CNF or conjunctive normal form is a conjunction of clauses; a 3-CNF sentence is a conjunction of 3-clauses.

- For example,

$$ (P \lor Q \lor \neg S) \land (\neg P \lor Q \lor R) \land (\neg P \lor \neg R \lor \neg S) \land (P \lor \neg S \lor T) $$

*Is a 3-CNF sentence with four clauses and five proposition symbols.*
Mapping 3-Queens into 3SAT

At least 1 has a Q  
not exactly 2 have Q's  
not all 3 have Q's  

\[(Q_{1,1} \lor Q_{1,2} \lor Q_{1,3}) \land (Q_{1,1} \lor \neg Q_{1,2} \lor \neg Q_{1,3}) \land (\neg Q_{1,1} \lor Q_{1,2} \lor \neg Q_{1,3}) \land (\neg Q_{1,1} \lor \neg Q_{1,2} \lor Q_{1,3}) \land (\neg Q_{1,1} \lor \neg Q_{1,2} \lor \neg Q_{1,3})\]

Do the same for each row, the same for each column, the same for each diagonal, and 'ing them all together.

\[(Q_{2,1} \lor Q_{2,2} \lor Q_{2,3}) \land (Q_{2,1} \lor \neg Q_{2,2} \lor \neg Q_{2,3}) \land (\neg Q_{2,1} \lor Q_{2,2} \lor \neg Q_{2,3}) \land (\neg Q_{2,1} \lor \neg Q_{2,2} \lor Q_{2,3}) \land (\neg Q_{2,1} \lor \neg Q_{2,2} \lor \neg Q_{2,3}) \land (Q_{1,1} \lor \neg Q_{2,2} \lor \neg Q_{3,3}) \land (\neg Q_{1,1} \lor Q_{2,2} \lor \neg Q_{3,3}) \land (\neg Q_{1,1} \lor \neg Q_{2,2} \lor \neg Q_{3,3}) \land (\neg Q_{1,1} \lor \neg Q_{2,2} \lor \neg Q_{3,3}) \land (\neg Q_{1,1} \lor \neg Q_{2,2} \lor \neg Q_{3,3}) \land \vdots \land etc.\]
Converting N-SAT into 3-SAT

\[ A \lor B \lor C \lor D \equiv (A \lor B \lor E) \land (\neg E \lor C \lor D) \]

\[
\begin{align*}
A &= T & A &= F & A &= F \\
B &= F & B &= T & B &= F \\
C &= F & C &= F & C &= T \\
D &= F & D &= F & D &= F \\
E &= F & E &= F & E &= T
\end{align*}
\]

2 - SAT polynomial time but can't map all problem into 2 - SAT

Add in dummy variable E, not interested in its truth value from problem perspective nor does its truth affect satisfiability of original proposition

……..
Davis-Putnam Algorithm
(Depth-First Search)

\[(A \lor C) \land (\neg A \lor C) \land (B \lor \neg C)\]
\[\land (A \lor \neg B)\]
GSAT Algorithm

**Problem:** Given a formula of the propositional calculus, find an interpretation of the variables under which the formula comes out true, or report that none exists.

**procedure GSAT**

**Input:** a set of clauses \( \propto \), MAX-FLIPS, and MAX-TRIES

**Output:** a satisfying truth assignments of \( \propto \), if found

begin

for \( i := 1 \) to MAX-TRIES ; *random restart mechanism*

\( T := \) a randomly generated truth assignment

for \( j := 1 \) to MAX-FLIPS

if \( T \) satisfies \( \propto \) then return \( T \)

\( p := \) a propositional variable such that a change in its truth assignment gives the largest increase in total number of clauses of \( \propto \) that are satisfied by \( T \).

\( T := T \) with the truth assignment of \( p \) reversed

end for

end for

return “no satisfying assignment found”

end

V. Lesser; CS683, F10
GSAT Performance

GSAT versus Davis-Putnam (a backtracking style algorithm)

Domain: hard random 3CNF formulas, all satisfiable (hard means chosen from a region in which about 50% of problems are unsolvable)

| Domain: n-queens |

<table>
<thead>
<tr>
<th>formulas</th>
<th>GSAT M-FLIPS</th>
<th>tries</th>
<th>time</th>
<th>choices</th>
<th>depth</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>vars</td>
<td>clauses</td>
<td>200</td>
<td>6.4</td>
<td>0.4s</td>
<td>77</td>
<td>11</td>
</tr>
<tr>
<td>200</td>
<td>425</td>
<td>500</td>
<td>11.4</td>
<td>0.8s</td>
<td>42</td>
<td>15</td>
</tr>
<tr>
<td>600</td>
<td>425</td>
<td>500</td>
<td>2.5s</td>
<td>0s</td>
<td>84 x 10^3</td>
<td>19</td>
</tr>
<tr>
<td>700</td>
<td>526</td>
<td>1000</td>
<td>14s</td>
<td>14s</td>
<td>0.5 x 10^5</td>
<td>22</td>
</tr>
<tr>
<td>1500</td>
<td>645</td>
<td>1500</td>
<td>45s</td>
<td>45s</td>
<td>2.2 x 10^6</td>
<td>27</td>
</tr>
<tr>
<td>2000</td>
<td>860</td>
<td>2000</td>
<td>2.8m</td>
<td>2.8m</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>2500</td>
<td>1262</td>
<td>2500</td>
<td>4.1m</td>
<td>4.1m</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>3000</td>
<td>1275</td>
<td>6000</td>
<td>12m</td>
<td>12m</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>4000</td>
<td>1700</td>
<td>8000</td>
<td>34m</td>
<td>34m</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>5000</td>
<td>2150</td>
<td>10000</td>
<td>1.6h</td>
<td>1.6h</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

V. Lesser; CS683, F10
GSAT Performance (cont’d)

- **GSAT Biased Random Walk**
  - With probability $p$, follow the standard GSAT scheme, *i.e.*, make the best possible flip.
  - With probability $1 - p$, pick a variable occurring in some unsatisfied clause and flip its truth assignment. (Note: a possible uphill move.)
- GSAT-Walk < Simulated-Annealing < GSAT-Noise < GSAT-Basic

### Comparing noise strategies on hard random 3CNF formulas. (Time in seconds on an SGI Challenge)

<table>
<thead>
<tr>
<th>formula</th>
<th>basic</th>
<th>GSAT walk</th>
<th>noise</th>
<th>Simul. Ann.</th>
</tr>
</thead>
<tbody>
<tr>
<td>vars</td>
<td>clauses</td>
<td>time</td>
<td>flips</td>
<td>time</td>
</tr>
<tr>
<td>100</td>
<td>430</td>
<td>.4</td>
<td>7554</td>
<td>.2</td>
</tr>
<tr>
<td>200</td>
<td>860</td>
<td>22</td>
<td>284693</td>
<td>4</td>
</tr>
<tr>
<td>400</td>
<td>1700</td>
<td>122</td>
<td>2.6 \times 10^6</td>
<td>7</td>
</tr>
<tr>
<td>600</td>
<td>2550</td>
<td>1471</td>
<td>30 \times 10^6</td>
<td>35</td>
</tr>
<tr>
<td>800</td>
<td>3400</td>
<td>*</td>
<td>*</td>
<td>286</td>
</tr>
<tr>
<td>1000</td>
<td>4250</td>
<td>*</td>
<td>*</td>
<td>1095</td>
</tr>
<tr>
<td>2000</td>
<td>8480</td>
<td>*</td>
<td>*</td>
<td>3255</td>
</tr>
</tbody>
</table>

V. Lesser; CS683, F10
3SAT Phase Transition

- Easy -- Satisfiable problems where many solutions
- **Hard -- Satisfiable problems where few solutions**
- Easy -- Few Satisfiable problems

- Assumes concurrent search in the satisfiable space and the non-satisfiable space (negation of proposition)
A Simplistic Approach to Solving CSPs using Systematic Search

- **Initial state**: the empty assignment
- **Successor function**: a value can be assigned to any variable as long as no constraint is violated.
- **Goal test**: the current assignment is complete.
- **Path cost**: a constant cost for every step. – not relevant
What more is needed?

- Not just a successor function and goal test
- But also a means to **propagate the constraints** imposed by variables already bound along the path on the potential fringe nodes of that path and an early **failure test**
- Thus, need explicit representation of constraints and constraint manipulation algorithms
Exploiting Commutativity

- Naïve application of search to CSPs:
  - If use breath first search
  - Branching factor is $n \cdot d$ at the top level, then $(n-1)d$, and so on for $n$ levels ($n$ variables, and $d$ values for each variable).
  - The tree has $n! \cdot d^n$ leaves, even though there are only $d^n$ possible complete assignments!

- Naïve formulation ignores **commutativity** of all CSPs: the order of any given set of actions has no effect on the outcome.
  - [WA=red, NT=green] same as [NT=green, WA=red]

- Solution: consider a single variable at each depth of the tree.
Part of the map-coloring search tree

Variable 1 - WA

Variable 2 -- NT

Variable 3 -- Q
Next Lecture

- Informed-Backtracking Using Min-Conflicts Heuristic
  - Arc Consistency for Pre-processing
  - Intelligent backtracking
  - Reducing the Search by structuring the CSP as a tree search

- Extending the model of simple heuristic search
  - Interacting subproblem perspective