Midterm Exam

• Midterm will be on Thursday, March 13th
• It will cover material up until Feb 27th

Propositional Logic

• We’re still emphasizing Propositional Logic

• Very important question with this method
  – Does knowledge base of propositional logic satisfy a particular proposition?
  • Can we generate some sequence of resolutions that prove a proposition is possible?

Backtracking

• Another way (representation) for searching
  – Problem to solve is in CNF
    • Is Marvin a Martian given \(- M = 1\) (true)?
      – Marvin is green \(- G = 1\)
      – Marvin is little \(- L = 1\)
      – (Little and green) implies Martian \(- (L \land G) \Rightarrow M\)
      \(- (L \land G) \lor M\)
      \(- L \lor \neg G \lor M = 1\)
  • Proof by contradiction… are there true/false values for G, L, and M that are consistent with knowledge base and Marvin not being a Martian?
    – \(G \land L \land (\neg L \lor \neg G \lor M) \land \neg M = 0\) ?

Searching for variable values

• Want to find values such that:
  \(G \land L \land (\neg L \lor \neg G \lor M) \land \neg M = 0\)
  – Randomly consider all true/false assignments to variables until we exhaust them all or find match
    • \((G, L, M) = (1, 0, 0)\)… no
    \((0, 1, 0)\)… no
    \((0, 0, 0)\)… no
    \((1, 1, 0)\)… no
  – Alternatively…
Searching for variable values

- Other ways to find (G, L, M) assignments for:
  \[ G \land L \land (\neg L \lor \neg G \lor M) \land \neg M = 0 \]

- **Simulated Annealing (WalkSAT)**
  - Start with initial guess (0, 1, 1)
  - Evaluation metric is the number of clauses that evaluate to true
  - Move "in direction" of guesses that cause more clauses to be true
  - Many local mins, use lots of randomness

WalkSAT termination

- How do you know when simulated annealing is done?
  - No way to know with certainty that an answer is not possible
  - Could have been bad luck
  - Could be there really is no answer
  - Establish a max number of iterations and go with best answer to that point

So how well do these work?

- Think about it this way
  \[
  (\neg D \lor \neg B \lor C) \land (B \lor \neg A \lor \neg C) \land (\neg C \lor \neg B \lor E) \\
  \land (E \lor \neg D \lor B) \land (B \lor E \lor \neg C). 
  \]
  - 16 or 32 possible assignments are models (are satisfiable) for this sentence
  - Therefore, 2 random guesses should find a solution
  - WalkSAT and DPLL should work quickly

What about more clauses?

- If # symbols (variables) stays the same, but number of clauses increases
  - More ways for an assignment to fail (on any one clause)
  - More searching through possible assignments is needed
  - Let’s create a ratio, m/n, to measure #clauses / # symbols
  - We expect large m/n causes slower solution

Combining it all

- **4x4 Wumpus World**
  - The "physics" of the game
    - \[ D_{x,y} \iff (P_{x+1,y} \lor P_{x-1,y} \lor P_{x,y+1} \lor P_{x,y-1}) \]
    - \[ S_{x,y} \iff (W_{x,y+1} \lor W_{x,y-1} \lor W_{x+1,y} \lor W_{x-1,y}) \]
  - At least one wumpus on board
  - At most one wumpus on board (for any two squares, one is free)
  - \( n(n-1)/2 \) rules like: \( \neg W_{1,1} \lor \neg W_{1,2} \)
  - Total of 155 sentences containing 64 distinct symbols
Wumpus World

• Inefficiencies as world becomes large
  – Knowledge base must expand if size of world expands

• Preferred to have sentences that apply to all squares
  – We brought this subject up last week
  – Next chapter addresses this issue