Discussion #5: Midterm prep

May 9, 2007
Matthew Tong
Logistics

• HW 2&3 graded
• HW 4 due NOW!
• Review section tonight 7-9, room 4140
• Midterm tomorrow
  – In class
  – Chapters 1-9 & lectures through last Thurs
  – Cheat sheet (one side, standard size, handwritten)
HW 2-1

• Finite search space !-> finite search tree
  – Big notion is cycles
  – Any finite DAG -> finite search tree
HW 2-2

• Problem with either double or double + 1 and search for partic number
  – Biggest error was forgetting path (particularly in reverse)
HW 2-3

• Negative cost weights
  – If arbitrarily large, must search whole tree since single large negative value can drive cost of path down arbitrarily far
  – If bounded, can help for FINITE trees, since max path length
  – If negative weight cycles, will loop
• Negative cost weights
  – Reason humans don’t loop for scenic routes is the notion of diminishing returns
    • First time you drive a route, big reward
    • Then becomes boring
    • Implementable by altering weights
  – Human looping examples
    • Sleeping
    • Eating
    • Going to work, working, coming home, …, paycheck, ….
HW 2-4

- Uniform with fixed cost (>0) => breadth first search
- Best first search with increasing cost per depth => breadth first
- Best first search with decreasing cost per depth (1/d or -d) => depth first
- Best first search with h(n) = g(n) => uniform cost
- A* with h(n) = 0 => uniform cost
  - Note: if not 0, then not admissible!
HW 2-5

• Overestimating h()
  – Ended up being hard problem
  – Since needed to show things, if not doing on computer helps to choose wisely
  – h(n) = Manhattan except for

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
HW 2-5

- So given puzzle:

<table>
<thead>
<tr>
<th>1</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

- Easily solvable in 2 moves
- But by giving high cost to most desirable, ends up going long way around
HW 2-5

- If don’t overestimate by more than $c$, won’t be suboptimal by more than $c$
  - Proof by contradiction
  - Essentially follows proof of A*
HW 2-6

• Local beam search with \( k = 1 \) is hill climbing
• Local beam search with \( k = 1 \) and no limit on # of states is breadth first search
• Simulated annealing with \( T=0 \) is first choice hill climbing
  – Slight subtlety in R&N’s algorithm where picks at random and always chooses if improves - which is first choice
• Genetic algorithm with population size \( N = 1 \) is random walk
  – No cross over, only mutation
  – No fitness tests ever
HW 3-1

• # map coloring solution
  – 18
  – Needed to show some kind of work!
• Most constrained
  – Seemed to be some confusion with degree heuristic
    • It’s *NOT* because it has a large impact on other variables
    • Recall other name: minimum remaining values
    • It has lots of connections with variables that have already been set
    • Degree heuristics has lots of connections with variables that have NOT been set
HW 3-2

• Most constrained
  – Reduces the branching factor
    • Most work done in subtrees, not this level
    • Least remaining values reduces # of subtrees
    • Gives variables with large domains a chance to be more tightly constrained before being tackled
  – Fail-fast
    • Small # of choices left means most have been eliminated already
    • If there’s been an error, can find it faster since less searching
HW 3-2

• Least constraining
  – Most got this, simply a matter of leaving as many doors open as possible
HW 3-3a

• Variables: Locations of small rectangles
  – Such as (x,y) of one corner, maybe direction
• Domain: Valid positions within big rectangle (ignoring overlap)
• Constraints: No rectangles can overlap
• In general, defining with math (or logic) is much more clear
  – And if you actually had to implement, makes trivial since that’s the exact form most languages require
HW 3-3a

- n: # of small rectangles
- W, L: Width and length of big rect
- W_i, L_i: Width of length of small rect i
- Variables: \{C_i|i = 1\ldots n\} //C is for corner
- Domain: (X_i, Y_i) where X_i is between 0 and W - W_i and Y_i is between 0 and W - W_i
- Constraints:
  - For all i, j if i != j then x_j - x_i >= W_i or x_i - x_j >= W_j
  - For all i, j if i != j then y_j - y_i >= L_i or y_i - y_j >= L_i
HW 3-3b

- Class scheduling
- Since each class must have an assigned prof, time, and room this is your focus and should be your variable
- Domain: $C_i = (P, R, T)$ where $P$ in \{professors\}, $R$ in \{rooms\}, $T$ in \{time slots\}
HW 3-3b

• Constraints:
  – Forall $C_i = (P, R, T)$, $C_i$ must be on the list of classes $P$ can teach
  – Forall $C_i = (P_i, R_i, T_i)$, $C_j = (P_j, R_j, T_j)$ with $i \neq j$
    • $R_i = R_j \Rightarrow T_i \neq T_j$
    • $P_i = P_j \Rightarrow T_i \neq T_j$
HW 3-4b

• A) $9!$ Possible game trees played until board complete.
  – Many games end before this (as early as move 5)
  – Many symmetries reduce effective # of games
• B) Show tree, with symmetries
• C) Evaluations
• D) Minimax values, best move
HW 3-4c

• E) alpha-beta pruning
  – Lot of points lost here
  – Order matters!
    • Here specified best for alpha beta pruing
    • Meant that 8 of 12 leaves could be pruned!
    • Could also be something like left->right in tree
HW 3-5

• Game description
  – State descriptor
    • FULL state info
    • May want to include what’s hidden and what’s available
HW 3-5

– Move generator
  • Generates ALL possible moves
  • Set your domain right
    – General problem in AI
    – Too broad and have huge branching factor
    – Too small and could rule out correct solution
      » Also could blur subdivision, which makes debugging hellish - Generation and evaluation are separate for a reason!!
HW 3-5

- **Terminal test**
  - When is game done?
- **Utility function**
  - Value of game WHEN COMPLETE
HW 3-5

• Evaluation function
  – Expected value of a game state
  • NOT a game move
    – CLOSE to synonymous for 1-ply search
    – Would eliminate multi-ply search
  – Can be approximator of expected utility
  – Should be quick to calculate
HW 4-1

- Smoke => Smoke
  - Valid since ¬S ∨ S

- Smoke => Fire
  - Satisfiable

- (Smoke => Fire) => (¬Smoke => ¬Fire)
  - Satisfiable (would be Smoke <=> Fire)

- Smoke ∨ Fire ∨ ¬Fire
  - Valid since Fire ∨ ¬Fire is true
• \(((S \land H) \Rightarrow F) \iff ((S \Rightarrow F) \lor (H \Rightarrow F))\)
  – Valid
  – \((\neg (S \land H)) \lor F\)
    • \(\neg S \lor \neg H \lor F\)
  – \((\neg S \lor F) \lor (\neg H \lor F)\)
    • \(\neg S \lor \neg H \lor F\)
• \((S\Rightarrow F) \Rightarrow ((S \land H) \Rightarrow F)\)  
  - Valid

• \(B \lor D \lor (B \Rightarrow D)\)  
  - Valid

• \((B \land D) \lor \neg D\)  
  - Satisfiable
HW 4-2

• Not going to type up for today
• Good practice, may be something of this form
  – But not nearly as long
HW 4-3

• A) $P(A, B, B), P(x, y, z)$
  – Be sure to say which unification convention you’re using on test
  – $\{x/A, y/B, z/B\}$

• B) $Q(y, G(A, B)), Q(G(x, x), y)$
  – Fails since $y$ can’t unify with both $G(A, B)$ and $G(x, x)$ since $x$ can’t unify with both $A$ and $B$
HW 4-3

• C) Older(Father(y), y), Older(Father(x), John)
  – {x/John, y/John}

• D) Knows(Father(y), y)), Knows(x, x)
  – Fails since x cannot unify with both Father(y) and y
HW 4-4

• Horses, cows, and pigs are mammals
  – Horse(x) => Mammal(x)
  – Cow(x) => Mammal(x)
  – Pig(x) => Mammal(x)

• An offspring of a horse is a horse
  – Horse(x) ^ Offspring(x, y) => Horse(y)
HW 4-4

• BlueBeard is a horse
  – Horse(BlueBeard)
• Bluebeard is Charlie’s parent
  – Parent(Bluebeard, Charlie)
• Offspring and parent are inverse relationships
  – Offspring(x,y) => Parent(y, x)
  – Parent(x,y) => Offspring(y, x)
HW 4-4

• Every mammal has a parent
  – Mammal(x) => Parent(G(x), x)
HW 4-5

- On board