CS472
Foundations of Artificial Intelligence

Prelim I
October 4, 2004

(1) State Space Search

Greedy Best-First Search goal1
    States expanded: start B goal1

Iterative Deepening goal1
    States expanded: start start B C start B C D goal1

Hill Climbing goal1
    States expanded: start B goal1

Uniform Cost Search goal2
    States expanded: start C A B B E F D goal2

Admissible search question No. h(C) and h(E) (and possibly some others) overestimate the distance to the closest goal.

Grading guide: For part 1, each goal was worth 1 point. Each correct state list was worth 4 points; 3, if there was a small mistake; 2, for larger mistakes; 1 point was awarded for some minimal display of understanding of the problem; 0 points if it didn't seem as if you knew the algorithm at all.

For part 2, the admissible heuristic was worth 2 points for correctly answering *no*; 3 points were awarded for the correct reason why. It was also possible to get 1 point on this if you answered *yes* but also gave the proper reason why heuristics aren't admissible.

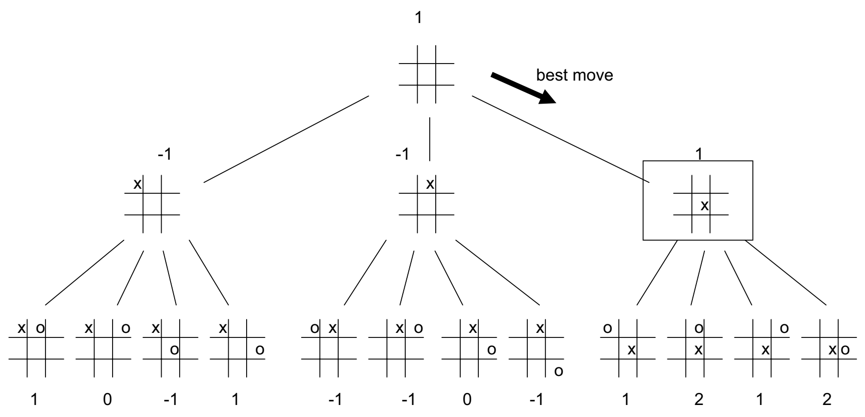
(2) Adversarial Search

For alpha-beta pruning, the best leaf for Max is one of the 1's in the rightmost branch. So the optimal ordering for alpha-beta pruning would be to evaluate this branch first (i.e. as if it were the left-most branch). None of the nodes in this branch can be pruned.

The ordering of the other branches doesn't matter. Within the 1-0-(-1)-1 branch, one of the 1's must be visited, but the remaining 3 leaves will be pruned by alpha-beta pruning. Within the (-1)-(-1)-0-(-1) branch, (any) one of the leaves must be visited, but then the rest will be pruned by alph-beta pruning.

Grading Guide: For part a, getting the evaluation function right with all correct leaves was worth 3 points. Each proper node in the tree above the leaves was worth 1 point and the proper best move was worth 3 points. If you got the leaves wrong, Eric was kind enough to grade the rest of the problem based on your leaves.

For part b, 10 points were awarded for a correct answer; 5 points, if you had at least 3/6 of the proper ones pruned; and 2 or 3 points if you had extras pruned.



Local Search and Genetic Algorithms

1. Short answer questions

   - hill-climbing
   - GA's do not maintain any path information. In addition, in each iteration of the search, a GA keeps track of a fixed number (although not just 1) of the best solutions and considers only "neighbors" of the search. (Other variations of answer are possible.)

2. GA components. Options include: representation for individuals, size of initial population, mutation operator, fitness function, selection function, stopping threshold.

   See R&N, section 4.3 for descriptions of each.

3. Genetic programming.

   - Terminals and primitive functions.

     The terminals represent any sensor input available to the agent. Some terminals include:

     CS current top of stack

     TB name of topmost correct block

     NN next needed block based on TB and goal

     Some primitive functions include:

     MS(x) move block x to stack from table

     MT(x) move top block on stack to table if x is anywhere in the stack

     DU(expr1, expr2) do until

NOT(expr) not

EQ(expr1, expr2) equal

- Crossover operates by replacing a randomly chosen subtree of parent1 by a randomly chosen subtree of parent2, and vice versa.
- An individual is evaluated by executing the program over a set of training cases, which consist of a set of correct input/output pairs. The fitness score for the individual might then be the percentage of these that the program gets correct.

Grading Guide: Part one (item one) was a right/wrong thing (with the possibility of getting only one point if you named some incorrect specialized variation of hill climbing).

For part one (item two), the answer had to have something about not maintaining a path or children being very close to their parents and/or pre-mutated states. Not explaining the answer well resulted in a 1 or 2 point deduction.

For question two, 3 points were given per feature listed — 1 point for naming it and 2 for describing it. A variety of answers were acceptable, although they could not have to do with crossover and the same answer used twice (often with conveniently different names :)) only received credit for one answer.

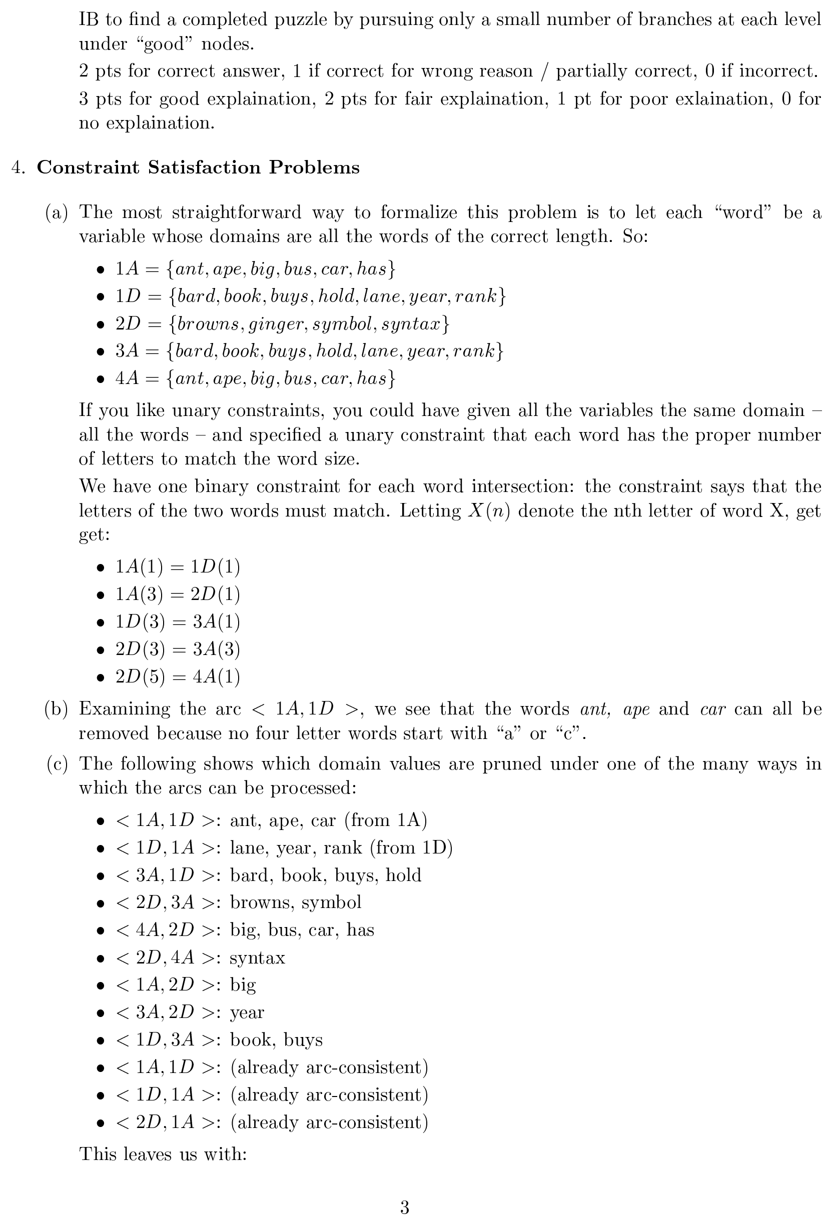
For question three (genetic programming), item 1: one point deducted for each missing or incorrect terminal; one point deducted for each missing or incorrect primitive function.

For the “crossover” part to question three, points were deducted for not making clear (a) that there are two parents/programs involved, (b) that a randomly chosen **subtree** is selected from each parent, and (c) that the two subtrees are swapped.

For the “quality” part to question three, 3 points were awarded for noting the need to execute the program over a set of correct I/O pairs; 2 points for correctly explaining how the fitness score would be computed.

Constraint Satisfaction Problems

1. See 4a in the figure.
2. Not shown.
3. Examples are provided in 4b and 4c in the figure.



IB to find a completed puzzle by pursuing only a small number of branches at each level under "good" nodes.

2 pts for correct answer, 1 if correct for wrong reason / partially correct, 0 if incorrect.

3 pts for good explaination, 2 pts for fair explaination, 1 pt for poor exlaination, 0 for no explaination.

4. **Constraint Satisfaction Problems**

(a) The most straightforward way to formalize this problem is to let each "word" be a variable whose domains are all the words of the correct length. So:

- $1A = \{ant, ape, big, bus, car, has\}$
- $1D = \{bard, book, buys, hold, lane, year, rank\}$
- $2D = \{browns, ginger, symbol, syntax\}$
- $3A = \{bard, book, buys, hold, lane, year, rank\}$
- $4A = \{ant, ape, big, bus, car, has\}$

If you like unary constraints, you could have given all the variables the same domain – all the words – and specified a unary constraint that each word has the proper number of letters to match the word size.

We have one binary constraint for each word intersection: the constraint says that the letters of the two words must match. Letting $X(n)$ denote the nth letter of word X, get get:

- $1A(1) = 1D(1)$
- $1A(3) = 2D(1)$
- $1D(3) = 3A(1)$
- $2D(3) = 3A(3)$
- $2D(5) = 4A(1)$

(b) Examining the arc $< 1A, 1D >$, we see that the words *ant, ape* and *car* can all be removed because no four letter words start with "a" or "c".

(c) The following shows which domain values are pruned under one of the many ways in which the arcs can be processed:

- $< 1A, 1D >$: ant, ape, car (from 1A)
- $< 1D, 1A >$: lane, year, rank (from 1D)
- $< 3A, 1D >$: bard, book, buys, hold
- $< 2D, 3A >$: browns, symbol
- $< 4A, 2D >$: big, bus, car, has
- $< 2D, 4A >$: syntax
- $< 1A, 2D >$: big
- $< 3A, 2D >$: year
- $< 1D, 3A >$: book, buys
- $< 1A, 1D >$: (already arc-consistent)
- $< 1D, 1A >$: (already arc-consistent)
- $< 2D, 1A >$: (already arc-consistent)

This leaves us with:

3

Grading Guide: For part 1:

- A handful of people gave search problem formulations instead of CSP formulations. You lost the full 10 points, unfortunately.
- If you didn't explicitly list the variables, but described them somehow, you lost 1 point.

- If you ignored the variables completely, you lost 3 points.
- If you forgot the domain, you lost 1 or 2 points, depending on the solution.
- If you ignored the puzzle word length constraint, you lost 2 points.

For part 2.

- If you forgot an arc or two, you lost 1 point.
- If you put in a node for each length constraint, you lost 1 point.
- If you put in a node for each intersection constraint, you lost 1 point.

For part 3.

- You got 2 points for giving a domain value that could be pruned, and 3 points for the explanation.
- If it wasn't clear at all which arc allowed the pruning, you lost 1 point.