Midterm

CSC 242

20 March 2001

*Write your **NAME** legibly on the bluebook. Work all problems. Best strategy is not to spend more than the indicated time on any question (minutes = points). Open book, open notes.*

1. Heuristic Search: 15 Min. Suppose you want to use AI techniques to create $N \times N$ panmagic squares. That is, the elements are the integers between 1 and N^2 inclusive, and the sum along all rows and all columns and the main diagonals is the same, like this one for $N = 4$.

```
13  2 16  3
12  7  9  6
 1 14  4 15
 8 11  5 10
```

(There are many varieties of magic squares... see
`http://www.pse.che.tohoku.ac.jp/ msuzuki/MagicSquare.html`, for instance...but I digress...).

In as much technical detail as you have time for, say how you would set up this problem as a constrained search problem (5 min.), a heuristic search problem (5 min.), and a hill-climbing or simulated annealing search problem (5 min.).

Answer: It helps to figure out what the magic sum is... easy.

Most natural as CSP. The constraints come from the definition of panmagic. The variables in the problem are the square's elements. Thus all the row's elements must total 34, all elements must be different, etc. etc. Our operators are the assignment of an unassigned (due to constraints) integer value to a square position (variable). Simple DFS would be one way to get a solution, but backtracking search is more eficient... it checks all constraints whenever any variable is assigned. Even better is forward checking... this makes sure other variables can achieve legal values given the current assignment, and eliminates possibilities that cannot. This detects failure and narrows search both. Constraint propagation could work too, in which changes in the possible values of each variable recursively affect the values of other variables until the whole system settles down into a so-far consistent state.

A*: f = g + h, g is how many swaps, h is maybe how many rows, cols, diags are not equal to the magic sum (34 for 4 x 4).

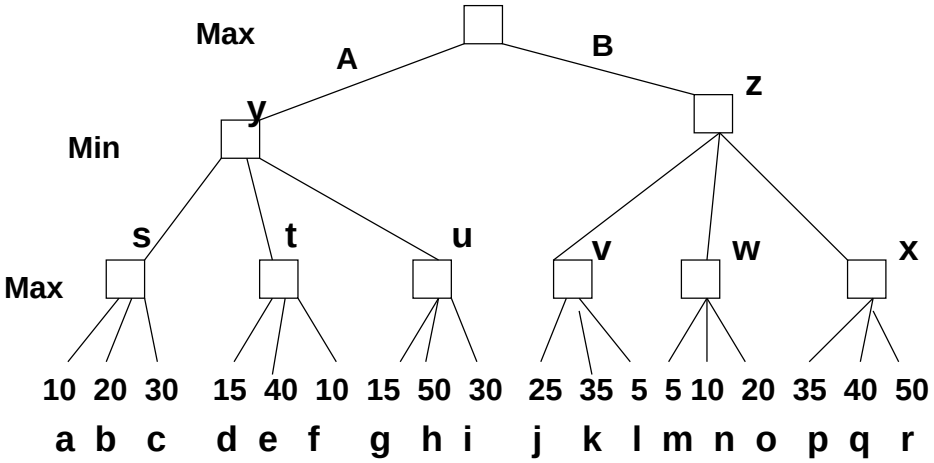
simulated annealing or hill-climbing: similar. swaps is the op, and goodness figure is maybe the h above, or something like the absolute value of the average difference of the sums from what they should be...

2. Games 5 Mins. The minimax algorithm returns the best move for MAX assuming MIN plays "optimally" (according to MAX's evaluation function at least). What happens when MIN plays suboptimally?

Answer:

Max's outcome can be no worse if MIN plays suboptimally. If there is a model of MIN's suboptimality (limited memory may limit his lookahead or he may be irrationally fond of his knights or ...) then that can be used to create his version of the evaluation function for use in game-tree analysis. However, as we have seen, minimax has no conception of "playing for complications" (i.e. fighting back in technically lost position to make it hard for the opponent to find the winning line), nor does it ever set traps. If the opponent is in fact limited, minimax is definitely not the best strategy.

3. Minimax and $\alpha - \beta$: 15 Mins. Consider the following game tree in which the indicated static scores are all from the first player (MAX)'s point of view. What move (A or B) should MAX choose and why (3 mins)?



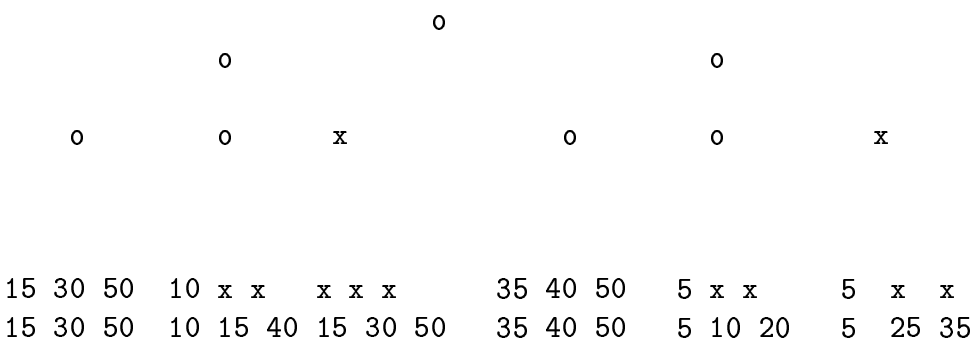
Which nodes (interior and leaf, a–z) would not be examined if alpha-beta pruning is used (5 mins)?

Suppose the move generator produces moves so that the static evaluation of positions is in the optimal order for $\alpha - \beta$ pruning? Draw the resulting tree and indicate which nodes would not be examined. (7 mins)

Answer: a) Max chooses A since its backed up value is 30 vs. 20 for B

b) f, i, x,p,q,r

c) x's not examined: positions generated in order of goodness.



4. Inference: 5 Min.

Generalized Modus Ponens (GMP) is a powerful rule that can collapse several resolution steps into one inference. Why might you not want to make GMP the only rule in your theorem proving system? Can you think of an example?

Answer:

Not complete. In particular, can't derive null clause from A and not-A.

5. Resolution Proof: 10 Min. Given these premises:

1. Anyone who prepares is smart.
2. Everybody has a friend who prepares.

Express them in FOPC, put them into clause form, and use resolution to answer the question: Who is smart? (That is, is anyone smart and if so, who?).

Answer:

$$\forall x(P(x) \Rightarrow S(x))$$

$$\forall x \exists y(F(x,y) \wedge P(y))$$

So we get clauses

$$\neg P(x) \vee S(x)(1)$$

$$F(x, f(x))(2)$$

$$P(f(x))(3)$$

Assert that no one is smart:

$$\neg S(x)(4)$$

Unify (3) and (1) , and the result with (4) (or the other order) and we find that if you give me any x, I'll give you good old $f(x)$, the individual generated by the Skolem function, in short it's x's geeky friend who prepares and who is smart. This is the brown-nosing individual who gives us the substitution that allows the derivation of the null clause.

6. Unification: 10 Mins.

You are given the clauses:

$$L(x, y, F(A, y), D)$$

$$L(z, C, F(w, u), v)$$

with variables u, v, w, x, y, z; constants A, C, and D; function F and predicate L.

a) (4 min) Find the most general unifying substitutions and show the result of unifying the clauses.

Answer:

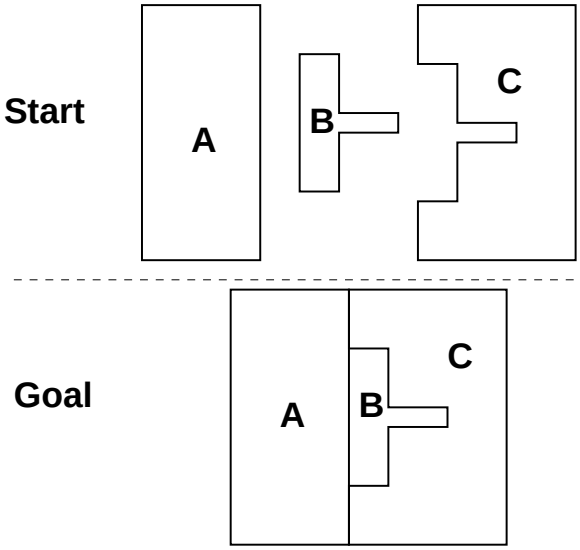
$w \mid A, y \mid C, u \mid C, v \mid Dx \mid z$ (or vice versa) to give (e.g.)
$L(x, C, F(A, C), D)$.
b) (3 min) Show a less general unifying substitutions and show the result of unifying the clauses.
Answer: use $x \mid E$ add $z \mid E$, instead of $x \mid z$, say.
c) (3 min) How would your answer to a) change if the clauses were

$$L(x, y, F(A, y), D)$$

$$L(y, C, F(x, u), z)?$$

Answer: It wouldn't. Common practice is to relabel vars so don't get mixed up, however.

7. Planning: 15 Mins.



You are going to use AI planning to tell you how to assemble the three pieces A, B, C into the structure shown. Assume that you have operators ATTACH(A,C) and ATTACH(B,C) that do the obvious right thing, and predicates ATTACHED(x,y) and NOT-ATTACHED(x,y). The more technical details the better.

1. How would STRIPS solve this problem? (7 min.)
2. How would UCPOP solve this problem? (8 min.)

Answer:

Some nice diagrams would be good.

Strips would axiomatize the situation and formalize operators with action descriptions, preconditions and postconditions (add and delete lists) for each operator. (Good if these are spelled out). It is a linear situation space planner, that does difference reduction to figure out what operators to apply; more specificially the operator and the differences fall out of the process of unifying the operators's axioms with the desired goals or subgoals. Strips would use a goal stack and might get lucky enough to attach B to C before A to C, however it might not. If it gets the operators in the wrong order it has to backtrack to sort things out.

UCPOP would develop a nonlinear plan by searching in plan space, expanding a trivial plan that starts with a step whose results are the initial condition and ends in a step whose preconditions are the goal. It adds steps one at a time, each one of which should achieve one of the preconditions of the goal or another step. These causal links are not necessarily in a defined temporal sequence.

However by noticing when an action clobbers a precondition in another step, ucpop removes conflicts by promoting or demoting steps (force to be before or after another). In this case UCpop would consider the two actions in parallel, notice a threat, and resolve it so that attach b,c happened before a,c. (Nice diagram here would be good). A UCPOP plan is complete when variable bindings and step-ordering constraints on the causal links are all consistent. (Beautiful fully-labelled diagram of final plan would be lovely here!!). This means that the resulting plan can be interpreted as one (or more) sequences of steps– the PO in POP means that the plan does not specify step order unless it is necessary.