Homework 3 solutions

6.2

a. Solution A: There is a variable corresponding to each of the n^2 positions on the board.
Solution B: There is a variable corresponding to each knight.

b. Solution A: Each variable can take one of two values, $\{$occupied,vacant$\}$
Solution B: Each variable's domain is the set of squares.

c. Solution A: every pair of squares separated by a knight's move is constrained, such that both cannot be occupied. Furthermore, the entire set of squares is constrained, such that the total number of occupied squares should be k.
Solution B: every pair of knights is constrained, such that no two knights can be on the same square or on squares separated by a knight's move. Solution B may be preferable because there is no global constraint, although Solution A has the smaller state space when k is large.

d. Any solution must describe a *complete-state* formulation because we are using a local search algorithm. For simulated annealing, the successor function must completely connect the space; for random-restart, the goal state must be reachable by hillclimbing from some initial state. Two basic classes of solutions are:
Solution C: ensure no attacks at any time. Actions are to remove any knight, add a knight in any unattacked square, or move a knight to any unattacked square.
Solution D: allow attacks but try to get rid of them. Actions are to remove any knight, add a knight in any square, or move a knight to any square.

6.6 The problem statement sets out the solution fairly completely. To express the ternary constraint on A, B and C that $A + B = C$, we first introduce a new variable, AB. If the domain of A and B is the set of numbers N, then the domain of AB is the set of pairs of numbers from N, i.e. $N \times N$. Now there are three binary constraints, one between A and AB saying that the value of A must be equal to the first element of the pair-value of AB; one between B and AB saying that the value of B must equal the second element of the value of AB; and finally one that says that the sum of the pair of numbers that is the value of AB must equal the value of C. All other ternary constraints can be handled similarly.

Now that we can reduce a ternary constraint into binary constraints, we can reduce a 4-ary constraint on variables A, B, C, D by first reducing A, B, C to binary constraints as shown above, then adding back D in a ternary constraint with AB and C, and then reducing this ternary constraint to binary by introducing CD.

By induction, we can reduce any n-ary constraint to an $(n-1)$-ary constraint. We can stop at binary, because any unary constraint can be dropped, simply by moving the effects of the constraint into the domain of the variable.

6.12 On a tree-structured graph, no arc will be considered more than once, so the AC-3 algorithm is $O(ED)$, where E is the number of edges and D is the size of the largest domain.

6.14

We establish arc-consistency from the bottom up because we will then (after establishing consistency) solve the problem from the top down. It will always be possible to find a solution (if one exists at all) with no backtracking because of the definition of arc consistency: whatever choice we make for the value of the parent node, there will be a value for the child.

5.9 For **a**, there are at most 9! games. (This is the number of move sequences that fill up the board, but many wins and losses end before the board is full.) For **b–e**, Figure S5.4 shows the game tree, with the evaluation function values below the terminal nodes and the backed-up values to the right of the non-terminal nodes. The values imply that the best starting move for X is to take the center. The terminal nodes with a bold outline are the ones that do not need to be evaluated, assuming the optimal ordering.

5.16

a. See Figure S5.5.

b. Given nodes 1–6, we would need to look at 7 and 8: if they were both $+\infty$ then the values of the min node and chance node above would also be $+\infty$ and the best move would change. Given nodes 1–7, we do not need to look at 8. Even if it is $+\infty$, the min node cannot be worth more than -1, so the chance node above cannot be worth more than -0.5, so the best move won't change.

c. The worst case is if either of the third and fourth leaves is -2, in which case the chance node above is 0. The best case is where they are both 2, then the chance node has value 2. So it must lie between 0 and 2.

d. See figure.

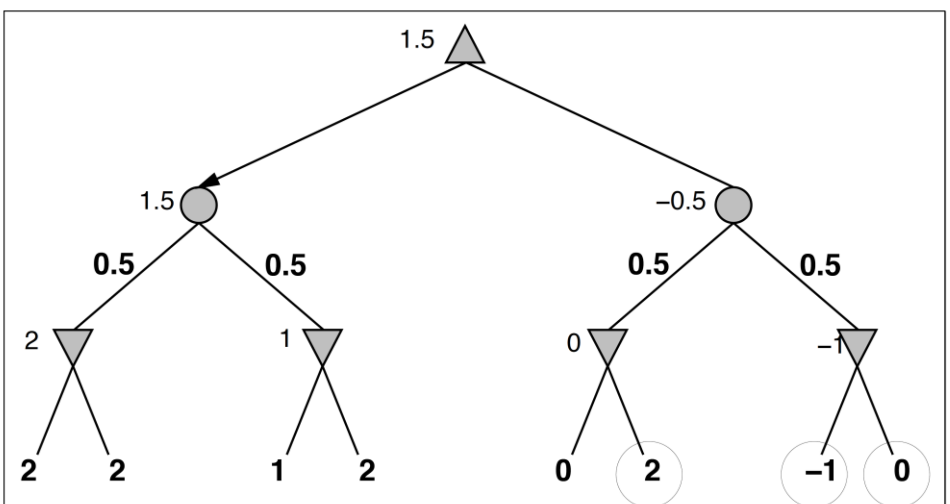

Figure S5.5 Pruning with chance nodes solution.