CS151 - Written Problem 3
Solutions

Based on similar problems found at:
http://www-nlp.stanford.edu/∼grenager/cs121/handouts/hw1.pdf

1. The book describes how you can formulate the Sudoku game as a CSP problem. Another way to formulate the Sudoku game is as a local search problem. Design a local search algorithm that is likely to solve Sudoku quickly, and write it in psuedocode. You may want to look at the `WalkSAT` algorithm (pg. 263) for inspiration. Do you think it will work better or worse than the best incremental search algorithm on easy problems? On hard problems? Why?

   ```
   def solveSudoku( SudokuBoard game, int numIter):
    foreach unbound var in game, assign randChoice([1...9])
    for i in range(numIter):
     if checkConstraints(game):
      return game
     choose a variable, v, that violates a constraint
     v = randChoice([1...9]) # ensuring v gets a new value)
   ```

   This algorithm will probably work better than incremental search on hard problems, but not easy problems because on easy problems, incremental search will direct itself quickly toward the correct solution, while the random search will needlessly explore fruitless assignments of values to variables. On the other hand, on hard problems, this approach might stumble on the right solution pretty quickly, while local search will probably take a very long time to find it.

2. We presented simulated annealing in lecture as a local search algorithm which uses randomness to avoid getting stuck in local maxima and plateaus.

(a) For what types of problems will greedy local search (hill climbing) work better than simulated annealing? In other words, when is the random part of simulated annealing no necessary?

For problems with no local maxima in the cost function, greedy search will work better.

(b) For what types of problems will randomly guessing the state work just as well as simulated annealing? In other words, when is the hill-climbing part of simulated annealing not necessary?

For problems where the cost function has no real structurei.e., the area around the global maximum looks like any other area of the function, so you would have no idea that you are even getting close to the solution, nor would the shape of the function necessarily push you in the right direction to find it.

(c) Reasoning from your answers to parts (b) and (c) above, for what types of problems is simulated annealing a useful technique? What assumptions about the shape of the value function are implicit in the design of simulated annealing?

Problems with a cost function with an overall global structure, but with some local maxima.

3. Exercise 6.2

a. A variable $x_1, x_2, ..., x_k$ represent the position of the k knights.

b. The *domain* of the variable is nxn, that is any board configuration, e.g. $(1, 2)$.

c. No knights may attach eachother. Let $x(1)$ denote the first coordinate of the position and $x(2)$ the second. For each pair x_i, x_j, introduce constraints: $x_i(1) \neq x_j(1) + a$ AND $x_i(2) \neq x_i(2) + b$ for $S = \{-2, -1, 1, 2\}$ for all pairs $(a \in S, b \in S)$ where $a \neq b$

d. The key challenge here is that we both need to be able to change the board configuration as well as add additional knights (unlike the local search version of the n-queens problem). One set of actions allows us to move any knight on the board to an unoccupied location. A second set of actions allows us to introduce a *new*

knight at an unoccupied location on the board. One reasonable heuristic would be the number of knights on the board, minus the number of conflicting knights (or the number of conflicts).

4. Exercise 6.9

   Both of these attempt to give the approach the best chance of succeeding. Picking a value for the variable that is most constrained makes tries as hard as possible to find a value for that variable since it can be problematic with many constraints. Similarly, picking the value that is least constrained, leaves the other values, which are already more constrained available to other variables.

   A secondary reason for picking the variable that is most constrained is that we it limits the search space higher up where is can be most expensive.