

CS 121, Summer 2009 Homework #3

Out: July 8, 2009
Due: July 20, 2009

How to complete this Homework: Your answers can be typed or carefully hand-written. Please begin each problem on a new page and make sure your name is written on each page of your assignment. Print out this cover sheet and fill in your name and email address, as well as any people you collaborated with. Staple all of your work together and turn in at the beginning of class, July 20, 2009. If you will not be in class you can turn it in prior to class under the door of Gates 132 (with the time of submission written on the homework), or email it to the course staff. Late homeworks will not be accepted.

Your Name:

Your email address:

Note on Honor Code: You must look at previously published solutions of any of these problems in preparing your answers. You may discuss these problems with other students in the class (in fact, you are encouraged to do so) and/or look into other documents (books, web sites), with the exception of published solutions, so long as your final submission is prepared on your own without referring to any notes taken during such collaboration. If you have discussed any of the problems with other students, indicate their name(s) here:

.....

Any intentional transgression of these rules will be considered an honor code violation.

General Information: Justify your answers, but keep explanations short and to the point. Excessive verbosity will be penalized. If you have any doubt on how to interpret a question, tell us in advance, so that we can help you understand the question, or tell us how you understand it in your returned solution.

Grading:

Problem #	Max. Grade	Your grade
1	15	
2	20	
3	20	
4	25	
5	20	
Total	100	

1. Genetic Algorithms [15 points]

In this question, you need to propose crossover and mutation procedures for genetic algorithms that will result in “valid” states. In the context of 8-queens, we define a valid state to be one in which there is only one queen on each row, and only one queen in each column.

The 8-queens problem, as discussed in class, asks you to place 8 queens on an 8×8 chessboard such that no two queens can attack each other (i.e. share the same row, column, or diagonal). The chromosome (string representing an individual) and crossover for the 8-queens problem discussed in class, as well as used in the book, Figure 4.15, generate chromosomes with exactly one queen per column, but often more than one queens (or none) per row. It is easy to see that an 8-queens state with more than one queen on any row cannot be a solution. Your task in this problem is to propose a chromosome representation, as well as crossover and mutation mechanisms that when applied will result in exactly one queen per column and one queen per row.

[Hint: Look at the chromosome in the book. What makes a chromosome valid (one queen per column and per row)? Think about how you can keep the chromosome valid. For crossover, you do not have to cut chromosome strings and concatenate them together. For mutation, you do not have to change one character in the chromosome strings.]

Specifically provide:

- (a) [3 points] The representation strings which represent valid states (or individuals) in your formulation
- (b) [7 points] A precise description of the crossover mechanism, an example of how it works, and detailed discussion of how it guarantees a resulting valid state.
- (c) [5 points] A precise description of the mutation mechanism, an example of how it works, and a discussion of how you can guarantee that it will produce a valid state.

2. Constraint Satisfaction: Design of a Crossword [20 points]

Consider the problem of constructing a 6×6 crossword puzzle. The goal is to place a valid 6-letter word in each of the six rows and each of the six columns. We are given a large dictionary that contains all the valid 6-letter words (there are 26 letters in total, A through Z). There are N words in the dictionary.

- (a) [8 points] Formulate this problem as a constraint satisfaction problem with 12 variables. What are these variables and their domains? How many constraints are there? What are they?
- (b) [8 points] Assume that the dictionary only allows the following type of query: Given any partial word in which $0 \leq k \leq 6$ letters and their locations are known, the dictionary returns the number (possibly 0) of valid words that contain these letters at these locations. If this number is not 0, then the dictionary also gives one such valid word picked at random. For instance, a partial word can be “A _ O _ _ T” (here, $k = 3$) and a valid word containing these letters is AMOUNT.

Let the crossword be constructed using a backtracking algorithm. After this algorithm has inserted the n -th word in the crossword, how should it determine if it needs to backtrack, and, if it does not backtrack, how does it select the variable that will be assigned a value next?

- (c) [4 points] Assume now that the dictionary returns all the valid words that contain the k letters at their given positions. How can this additional information be used by the backtracking algorithm? (Be concise.)

3. Constraint Satisfaction: Sudoku Game [20 points]

The game of Sudoku consists of filling a 9×9 grid with digits $1, 2, \dots, 9$. The 9×9 grid is divided into nine 3×3 grids shown alternately white and gray in Figure 1.

	1	2	3	4	5	6	7	8	9
1									
2									
3									
4									
5									
6									
7									
8									
9									

Figure 1: Grid of Sudoku game

Each row, each column, and each of the nine 3×3 grids should contain each of the nine digits exactly once. A specific instance of the Sudoku game is one in which certain digits have already been placed, like in Figure 2.

	1	2	3	4	5	6	7	8	9
1	5	6	9	4					
2	2					5	4		9
3			1		8	3		2	
4					7		9	1	
5				8		1			3
6		5	2		9				7
7		7		2	1		6		
8	4		5	7					8
9						9	7	3	2

Figure 2: A specific instance of a Sudoku game

The goal of the game is then to fill the remaining cells.

- (a) [12 points] Assume that we formulate the Sudoku game as a Constraint Satisfaction Problem with 81 variables $X_{ij}, i = 1, \dots, 9, j = 1, \dots, 9$, in which each variable X_{ij} corresponds to the cell of the grid at row i and column j . The initial domain of each variable is $\{1, 2, \dots, 9\}$. The constraints are those described above. A solution is a

complete assignment that satisfies all these constraints. Any specific Sudoku game like the one shown in Figure 2 can be input by directly assigning a value to some of the 81 variables.

- i. [4 points] Assume that forward checking is performed immediately after having input a specific game. Explain how it would work in general for a Sudoku game. In particular, what would be the remaining domain for the variable X_{23} in Figure 2?
 - ii. [2 points] Why is it not very beneficial to use AC3 to eliminate additional values from variable domains for this problem?
 - iii. [2 points] Consider variable X_{24} in Figure 2. Why can we assign the value 1 to it right away? Can forward checking and AC3 find that X_{24} must take the value 1?
 - iv. [4 points] Explain how the most-constrained variable and most-constraining variable heuristics would work for this problem. In particular, for the application of the most-constraining variable, explain precisely how you would count the number of variables related by a constraint to a variable X_{ij} .
- (b) [8 points] Let us now define another set of 3×81 variables. We know that each of the nine digits 1 through 9 must appear exactly once in every row, every column, and every 3×3 grid. So, let us define a set of 9 variables for each row, for each column and for each 3×3 grid. For row i , we define the 9 variables R_i^1, \dots, R_i^9 , such that the value of R_i^m designates the column number where the digit m appears in row i . For each column j , we define 9 variables C_j^1, \dots, C_j^9 , such that the value of C_j^m designates the row number where the digit m appears in column j . For each 3×3 grid k (see below), we define 9 variables G_k^1, \dots, G_k^9 , such that the value of G_k^m designates the cell number (see below) where the digit m appears in grid k . The 3×3 grids are numbered $k = 1, \dots, 9$ as follows:

	1	2	3	4	5	6	7	8	9
1									
2	1	2	3						
3									
4	4	5	6						
5									
6	7	8	9						
7									
8									
9									

and the cells in any 3×3 grid are numbered as follows:

1	2	3
4	5	6
7	8	9

So, $G_2^1 = 4$ implies that cell 4 of the 2nd 3×3 grid contains 1, i.e., that $X_{24} = 1$.

- i. [4 points] What are the constraints involving the variables R_i^m ? [Give three “types” of constraints that do not only differ by the values of i and m .]
- ii. [2 points] What is the domain of G_2^1 after applying forward checking on this set of variables in the game of Figure 1?
- iii. [2 points] Same question for G_6^2 .

4. Local search algorithms [25 points]

Another technique that might work well in solving the Sudoku game (See question 3), is local search. Please design a local search algorithm that is likely to solve Sudoku quickly. Please describe your algorithm as precisely as possible, we encourage you to use some readable pseudocode style. Specifically provide:

- (a) [5 points] The state representation you will use to represent a state
- (b) [5 points] How successors of a single state will be generated
- (c) [5 points] The objective function you will use to evaluate each state. Specify clearly how this will be computed (you might want to include an example to make it clear), and what the value of this function will be when you have reached the goal
- (d) [9 points] The algorithm you would use to try and solve Sudoku using local search and the components previously specified in this problem. This should be given in something as precise as pseudocode. You are free to use any of the local search algorithms from section 4.3.
- (e) [1 points] Briefly discuss how well you expect this approach to work, as well as why you chose the local search algorithm that you did.

5. Action planning and STRIPS [20 points]

In this problem we will describe a simple problem in STRIPS-style.

Suppose that you work for a local moving company. They have a single truck (*truck*), which can be either at the company office (A), at the customer’s old home (B), or at the customer’s new home (C). The customer has boxes (*boxes*), which can be loaded into the truck, if the truck and boxes are at the same location. The boxes can also be unloaded.

At first the truck is at the company office, but the company needs you to make a plan to use the truck to move the customer’s boxes from their old home to their new home, afterwards returning the truck to the office. If you are successful at this simple task they will allow you to design a more complex AI planning system for them, which will make you fabulously rich.

Your task for this problem is to write down the components of the problem in a STRIPS-style. Specifically:

- (a) [4 points] Write down the initial state description
- (b) [9 points] Write down the STRIPS-style definition of each of the 3 actions. (*drive*, *load*, *unload*)
For each action provide:
 - Action name and parameter list
 - Precondition
 - Effect, as an add list and a delete list

- (c) [4 points] Write down the goal state description
- (d) [3 points] Construct a plan which achieves the goal state starting from the initial state.