Analytical Questions

1.

a. There are $n^2$ variables, one variable for every square on the chessboard.

b. Each variable can have two possible values: 1. A knight is placed in this square or 2. The square is empty.

c. All the squares that can be attacked by a knight should be put under a constraint such that no other knight is placed on that square. For a single knight, there are a maximum of eight squares it can attack depending on its position. The constraint value on the entire set of squares should be 'k' as this is the maximum number of knights, we can put on the chessboard without them attacking each other.

2. The idea here is to try to find a solution as fast as possible while also trying to detect a failure when there is no solution present as fast as possible as well. The most constraining variable (MCV) is chosen because this is the variable that will detect an inevitable failure as soon as possible. The least constraining value (LCV) is used to ensure maximum flexibility so that we find a solution as fast as possible.

3.

a. Give each vertex a heuristic (h) value. Let the heuristic in this case be the Euclidean distance. Now since we need to go from G to S let G be the start state and S be the goal state. Calculate the Euclidean distance from S to each vertex (including G) and give these values to the respective vertices. Initialize G as the current state. Now we will go the actual algorithm:

1. Check if the current state is the goal state (h-value will be zero), if so then a path has been found. If not, look for all the states that are accessible from the current state.
2. Find the h-value that is the lowest (lower the h-value, the closer we are to the goal) among all of the neighbors of the current state and compare it with the h-value of the current state. If the h-value of the neighbor state is less than the h-value of the current state, then make the neighbor state the new current state. If not, then that means you are stuck in a "flat local maximum" and the hill-climbing search will continuously loop.
3. Repeat Steps 1 and 2.

b. The algorithm will stay almost the same as before but we modify the second step as follows:

2. Find the h-value that is the lowest (lower the h-value, the closer we are to the goal) among all of the neighbors of the current state and compare it with the h-value of the current state. If the h-value of the neighbor state is less than the h-value of the current state, then make the neighbor state the new current state. If not, then check if the h-value of the neighbor state is less than a certain value 'k'. If so, make the neighbor state as the new current state.

c. If we keep the ‘k’ at a certain maximum value then we can avoid a local minima. In this case, we can keep ‘k’ as the h-value of the start state G. That way the algorithm avoids going farther away from the goal state than it initially was when it began at the start state.

Programming Question

The program was submitted with a text file called “test.txt”. In order to run the program, the user must enter their inputs into the text file. The user should enter the environment they wish to run on the first line. Input 1 to run the environment shown in Fig. 3.31, input 2 to run the new environment. The user should enter the C-value they wish to test on the second line. The file “test.txt” has been submitted with values that cause the program to run Environment 1 with a C-value of 5000.

C-values of 500 and 1000 did not work for either Environment 1 or 2. C-values of 1500, 2000 and 5000 worked for both Environment 1 and 2.