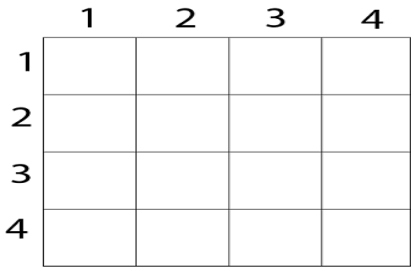
Beyond Classical Search (Chapter 4)

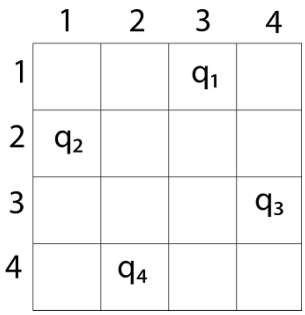
4 Queen Problem:

Consider a 4*4 chessboard. Let there are 4 queens. The objective is place there 4 queens on 4*4 chessboard in such a way that no two queens should be placed in the same row, same column or diagonal position.



4x4 chessboard

- Since, we have to place 4 queens such as q_1 q_2 q_3 and q_4 on the chessboard, such that no two queens attack each other. In such a conditional each queen must be placed on a different row.
- Now, we place queen q_1 in the very first acceptable position (1, 1). Next, we put queen q_2 so that both these queens do not attack each other. We find that if we place q_2 in column 1 and 2, then the dead end is encountered.
- Thus the first acceptable position for q_2 in column 3, i.e. (2, 3) but then no position is left for placing queen 'q_3' safely. So we backtrack one step and place the queen 'q_2' in (2, 4), the next best possible solution.
- Then we obtain the position for placing 'q_3' which is (3, 2). But later this position also leads to a dead end, and no place is found where 'q_4' can be placed safely. Then we have to backtrack till 'q_1' and place it to (1, 2) and then all other queens are placed safely by moving q_2 to (2, 4), q_3 to (3, 1) and q_4 to (4, 3).
- That is, we get the solution (2, 4, 1, 3). This is one possible solution for the 4-queens problem. For another possible solution, the whole method is repeated for all partial solutions. The other solutions for 4 - queens problems is (3, 1, 4, 2) i.e.



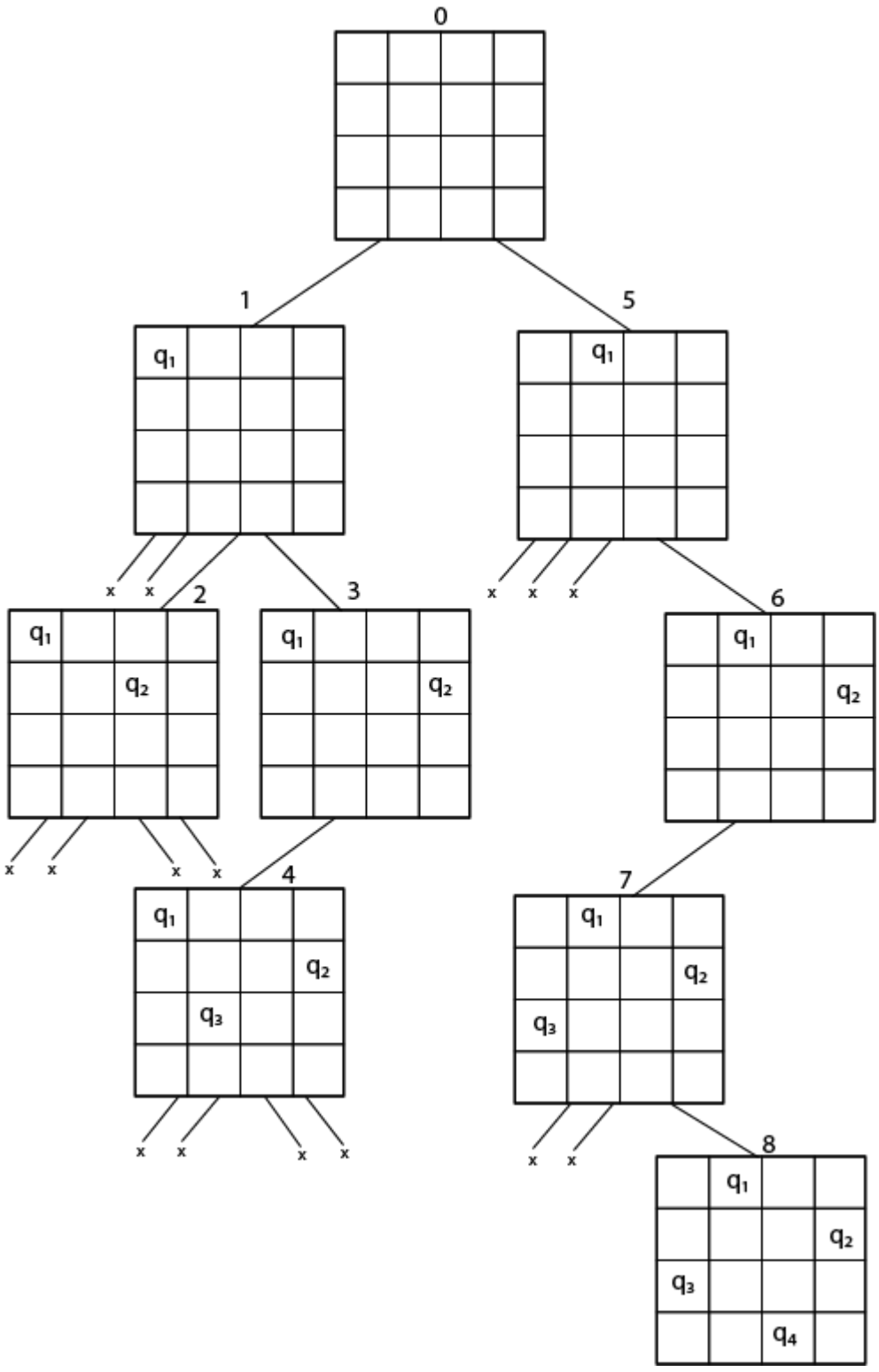
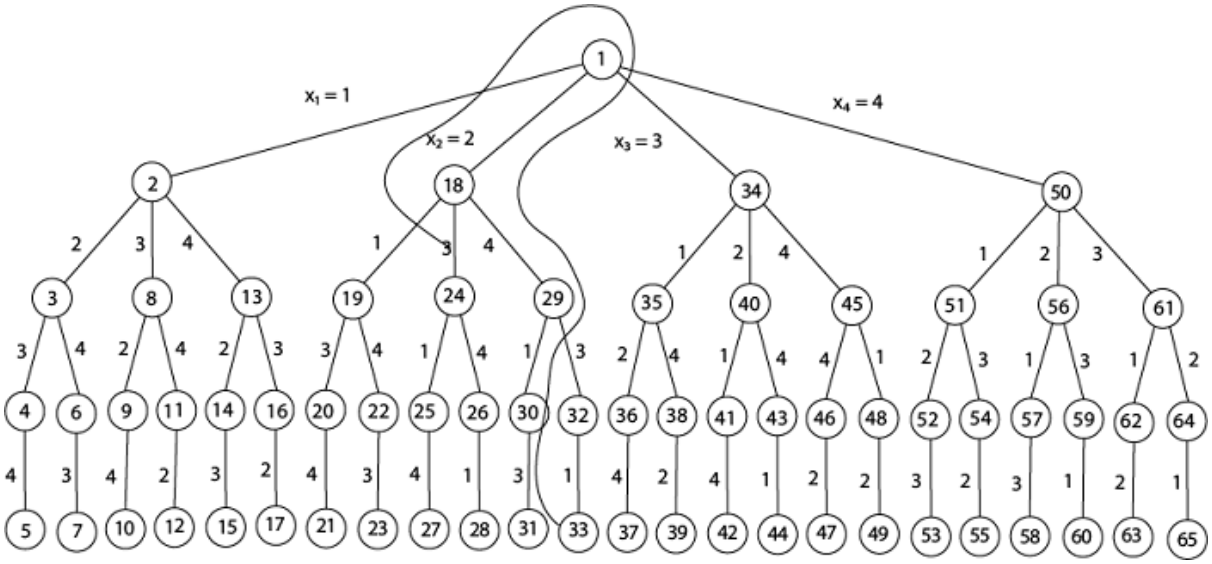
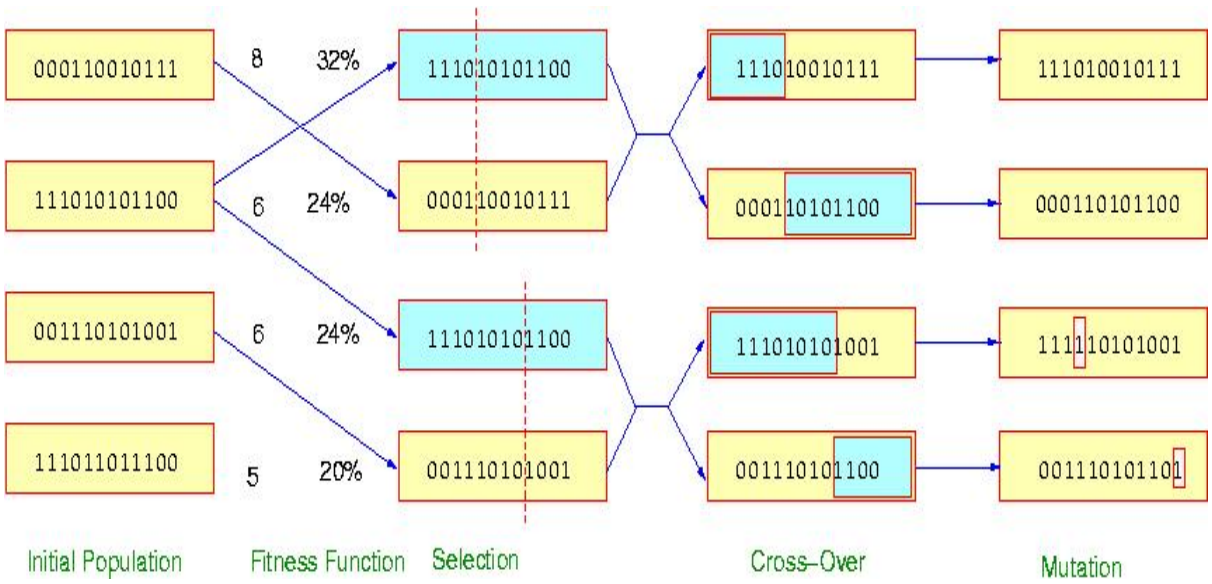


Fig shows the complete state space for 4 - queens problem. But we can use backtracking method to generate the necessary node and stop if the next node violates the rule, i.e., if two queens are attacking.



4 - Queens solution space with nodes numbered in DFS

Genetic Algorithm



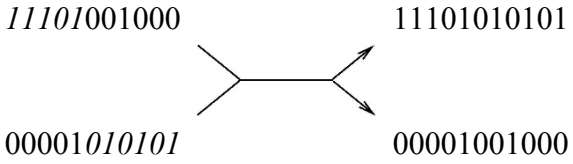
Create new generation P_s

1. Probabilistically select (1-r)p members of P to add to P_s. Probability of an element to be selected in one step is proportional to its fitness, i.e. ratio of its fitness divided by the sum of all fitness values.

2. Probabilistically select [r* p]/[2] pairs of P for crossover. Probability of an element to be selected in one step is proportional to its fitness. For each pair generate two offsprings. Add all offsprings to P_s.
3. Randomly flip each bit in P_s in the representation with (small) probability m.
4. Update P as P_s

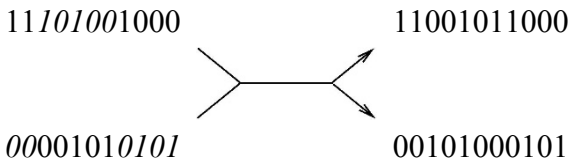
Genetic Operators

Single-point crossover



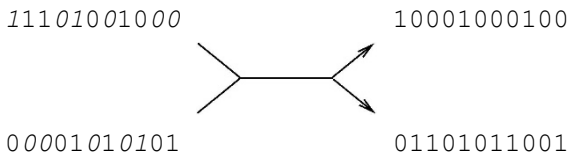
The procedure takes two bit strings of length n, randomly selects a number i between 1 and n and breaks up the strings at position i and puts together the first part of the first string with the second part of the second as well as the first part of the second string with the second part of the first string.

Two-point crossover



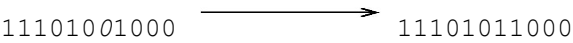
The procedure takes two bit strings of length n, randomly selects two numbers i and j between 1 and n and breaks up the strings at positions at i and j and puts together the first and last part of the first string with the middle part of the second as well as the first and last part of the second string with the middle part of the first string.

Uniform Crossover



The procedure takes two bitstrings of length n, randomly decides for each position of whether the bit of the first or the second should go in the first or in the second child.

Point Mutation



Local beam Search