Constraint Satisfaction Problems: n-queens

The purpose of this CP468 term project is to implement the Local Search algorithm MIN-CONFLICTS and apply it to the n-queens CSP. You can use the programming language of your choice.

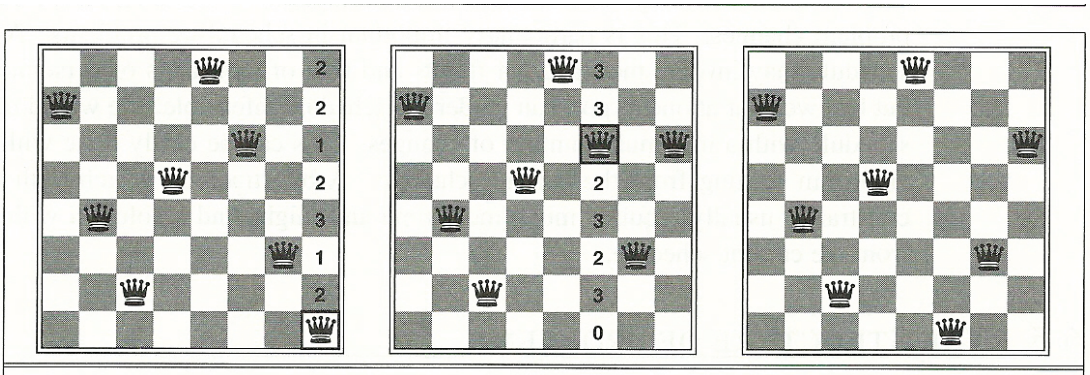

Figure 6.9 A two-step solution using min-conflicts for an 8-queens problem. At each stage, a queen is chosen for reassignment in its column. The number of conflicts (in this case, the number of attacking queens) is shown in each square. The algorithm moves the queen to the min-conflicts square, breaking ties randomly.

Project Resources

The MIN-CONFLICTS algorithm has been seen in class and is also described on pages $220 - 221$ of AIMA 3^{rd} ed. Write an additional function that takes as input a configuration of n queens and decides whether or not this is a solution to the n-queens problem.

Testing the implementation

You will have to decide on a reasonably readable format to represent n-queens configurations on the $n \times n$ chessboard/grid. The main focus of the project is on the algorithmic and programming aspects of the problem. Try your implementation with increasingly larger values of n. What is the maximum value of n for which you are able to solve the n-queens problem in a reasonable amount of time? Find solutions to the n-queens problem for the following values of n (representing orders of magnitude) $10, 100, 1000, 10000, 100000, 1000000$.

What to submit by the due date

Each group should submit **one** .zip file (code) and **one** .pdf file (design document) whose contents should be:

1. the cover sheet, (Names/Student IDs of group members, Course Number/Name, Date);
2. discussion of the design choices you made in your implementation (e.g data structures used and so forth);
3. clear instructions of how to install/compile/execute your code;
4. results of the test runs of your code on the range of n-values specified above.
5. a poster with graphical representations of a solution you found with your code, for a large value of n.

Note: Your code should be well-commented.

$CW(n, k^2)$

Problem description

The problem consists in searching for arrays called $CW(n, k^2)$ where n, k are positive integers.

A $CW(n, k^2)$ is an array $[a_1, \ldots, a_n]$ of length n with elements from $\{-1, 0, +1\}$, that has the following properties:

- There are exactly k^2 non-zero elements and exactly $n - k^2$ zero elements among $a_1, \ldots, a_n$
- $(a_1 + \ldots + a_n)^2 = k^2$ constant sum
- Set $m = n/2$ when n is even and $m = (n-1)/2$ when n is odd. Then the following m autocorrelation equations must be satisfied.

$$a_1 a_{1+s} + a_2 a_{2+s} + \cdots + a_{n-1} a_{(n-1+s)} + a_n a_{(n+s)} = 0 \quad s = 1, \ldots, m$$

where the indices $1 + s, 2 + s, \ldots, n - 1 + s, n + s$ are taken modulo n, when they exceed n.

Example: $CW(24, 9)$

$$[0, 0, 0, -1, -1, 0, 0, 0, 0, 0, 1, -1, 0, 0, 0, -1, 1, 0, 0, 1, 0, 0, -1, -1]$$

Example: $CW(28, 4)$

$$[-1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$

It is advisable to write code to test the correctness of the two examples above.
Note that there are several values of n, k for which $CW(n, k^2)$ do not exist.

Two search algorithms

Decide on two different appropriate search algorithms from the ones seen in class, to search for $CW(n, k^2)$. Implement the algorithms that you decided upon and test them with the following four cases:

$$CW(24, 9), \quad CW(28, 4), \quad CW(48, 36), \quad CW(142, 100)$$

Additional values of n, k will be provided during the project demonstration.

What to submit by the due date

Each group should submit **one** .zip file (code) and **one** .pdf file (design document) whose contents should be:

1. the cover sheet, (Names/Student IDs of group members, Course Number/Name, Date);
2. discussion of the design choices you made in your implementation (e.g data structures used and so forth);
3. clear instructions of how to install/compile/execute your code;
4. results of the test runs of your code on the 4 cases given above.

Note: Your code should be well-commented.

Path Planning

The purpose of this CP468 term project is to design and implement an $A^\star$-based algorithm to solve a path planning problem. You can use the programming language of your choice.

Project Description

Consider a Museum room that is patrolled by N robots at night. At a pre-determined time, the robots are supposed to rendezvous at a given point R in the room. The robots move inside the room, and the room contains obstacles, such as chairs and benches for the visitors, paintings, sculptures etc. The robots are supposed to know the locations of the obstacles in the room.

Implement an $A^\star$-based algorithm to compute the path of each robot, from its initial position to the given rendezvous point R.

Technical details

The location of the obstacles in the room will be given in the form of a text file. The text file will contain the dimensions of the room, the number of robots N, the initial position of each robot, the given rendezvous point R as well as the locations of the obstacles in the room. All coordinates in the text file are expressed as Cartesian coordinates in the plane.

Here is an example of an input text file: (0 means vacant point, 1 means obstacle point)

```
8 10            //  the room has dimensions 8 by 10
2               //  there are N = 2 robots
2 1             //  1st robot initial position: point (2,1)
8 2             //  2nd robot initial position: point (8,2)
4 7             //  the rendezvous point R has coordinates (4,7)
1000000001      // room points (0,7), (1,7), ... (9,7)
1100000011
0000000000
1000110001
1001111001
0001111000
0000110000      // room points (0,1), (1,1), ... (9,1)
1100000011      // room points (0,0), (1,0), ... (9,0)
```

What to submit by the due date

Each group should submit **one** .zip file (code) and **one** .pdf file (design document) whose contents should be:

1. the cover sheet, (Names/Student IDs of group members, Course Number/Name, Date);
2. discussion of the design choices you made in your implementation (e.g data structures used and so forth);
3. clear instructions of how to install/compile/execute your code;
4. results of the test runs of your code on 4 different input text files with $N = 1, 2, 3, 4$ robots and room dimensions between 20 and 30. Additional test files will be provided during the project demonstration.

Note: Your code should be well-commented.

Simple Genetic Algorithm and Applications

Introduction

The purpose of this CP468 term project is to implement a Simple Genetic Algorithm (SGA) and apply it to solve some specific problems. You can form groups of up to 4 members, to work collaboratively on this project. You can use the programming language of your choice.

Project Resources

Read the first two chapters of the book:
`Genetic Algorithms in Search, Optimization, and Machine Learning` by David E. Goldberg, Addison-Wesley, 1989. SGA is presented in detail in the first two chapters.

Test Objective Functions

Try your implementation with the three classical benchmark OFs:
(a) De Jong Sphere function (b) Rosenbrock's Valley (c) Himmelblau Function.
Additional OFs will be provided during the project demonstration.

What to submit by the due date

Each group should submit **one** .zip file (code) and **one** .pdf file (design document) whose contents should be:

1. the cover sheet, (Names/Student IDs of group members, Course Number/Name, Date);
2. discussion of the design choices you made in your implementation (e.g data structures used and so forth);
3. clear instructions of how to install/compile/execute your code;
4. results of the test runs of your code on the three classical benchmark OFs.

Note: Your code should be well-commented.