N-Queens via Relaxation Labeling

Project Report

Ilana Koreh (307247262)
Luba Rashkovsky (308820695)
## Table of Context

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>N-Queens problem</td>
<td>3</td>
</tr>
<tr>
<td>Relaxation labeling algorithm</td>
<td>5</td>
</tr>
<tr>
<td>Relaxation labeling algorithm and N-Queens problem</td>
<td>7</td>
</tr>
<tr>
<td>Reduction of N-Queens to Relaxation labeling algorithm</td>
<td>7</td>
</tr>
<tr>
<td>Queen's assignment by probability</td>
<td>8</td>
</tr>
<tr>
<td>Results</td>
<td>9</td>
</tr>
<tr>
<td>Problems</td>
<td>10</td>
</tr>
<tr>
<td>Simulator</td>
<td>11</td>
</tr>
<tr>
<td>Conclusion</td>
<td>12</td>
</tr>
<tr>
<td>Reference</td>
<td>12</td>
</tr>
</tbody>
</table>
**N-Queens problem**

The $N$-queens problem originally introduced in 1850 by Carl Gauss. The goal of this problem is to place $N$ queens on a $N \times N$ chessboard, so that no queens can take each other. Queens can move horizontally, vertically, and diagonally, this means that there can be only one queen per row and one per column, and that no two queens can find themselves on the same diagonal.

It is a computationally expensive problem - NP-complete, what makes it very famous problem in computer science.

Naïve solution for this problem based on building a search tree, where each node represents a valid position of a queen on the chessboard. Nodes at the first level correspond to one queen on the $N \times N$ board. Nodes at the second level represent boards containing two queens in valid locations, and so on. When a tree of depth $N$ is found then we have a solution for positioning $N$ queens on the board.

*Partial search tree for 8-queen problem.*

As the search progresses down the tree, more queens are added to the board. In this example we assume that queens are added on successive rows of the board, so as the search progresses down, we cover more rows of the board.

When the search reaches a leaf at the $N^{th}$ level, a solution has been found.
The program stops when it reaches a terminal leaf (success), or when all the subtrees have been visited without ever reaching a terminal leaf (no solutions).

Some examples of solution for this problem for different n's:

**4 Queens**

```
  1 2 3 4
  Q . . .
  . . . Q
  . Q . .
  . . . Q
```

**5 Queens**

```
  1 2 3 4 5
  Q . . . .
  . Q . . .
  . . . Q .
  . . . . Q
  . . Q . .
```

**6 Queens**

```
  1 2 3 4 5 6
  Q . . . . .
  . Q . . . .
  . . Q . . .
  . . . Q . .
  . . . . Q .
  . . . . . Q
```

Empirical observations of smaller-size problems show that the number of solutions increases exponentially with increasing $N$. Alternatively, search-based algorithms have been developed. For example, a backtracking search will systematically generate all possible solution sets for a given $N \times N$ board. Several authors have proposed other efficient search techniques to overcome this problem. These methods include search heuristic methods, local search and conflict minimization techniques. Recently, advances in research in the area of neural networks have led to several papers proposing solutions to the $N$-queens problem via neural networks.
Relaxation labeling algorithm

Relaxation labeling is a general name for group of methods for assigning labels to set of objects using contextual constraints. It was originally developed for use in the field of computer vision.

The algorithm works with set of objects $B = \{b_1, ..., b_n\}$. Each object should get one of the possible labels $\lambda = \{1, 2, ..., m\}$.

Initialization of this algorithm sets value $p^0_i(\lambda)$ for each object labeled by each label. This value is the measured confidence that object $b_i$ should be labeled by label $\lambda$.

\[
p^0_i(\lambda) \geq 0
\]
\[
\sum_{\lambda=1}^{m} p^0_i(\lambda) = 1 \quad \forall i
\]

Formulas used by the algorithm:

- $r_{ij}(\lambda, \mu)$ $\leftrightarrow$ The strength of compatibility between the hypotheses "$b_i$ has label $\lambda$" and "$b_j$ has label $\mu$"

For each iteration the algorithm calculates the support for label $\lambda$ for object $b_i$ (according to the compatibility of this label with the surrounding):

\[
s_i(\lambda) = \sum_{j=1}^{n} \sum_{\mu=1}^{m} r_{ij}(\lambda, \mu) p_j(\mu)
\]
• The original "ad-hoc" **update rule** – calculates the probability for labeling object $b_i$ with label $\lambda$ by current probability (from current iteration) and support for this label from the surrounding.

$$p_{i}^{k+1}(\lambda) = \frac{p_{i}^{k}(\lambda)S_{i}^{k}(\lambda)}{\sum_{\mu=1}^{m} p_{i}^{k}(\mu)S_{i}^{k}(\mu)}$$

• Under certain conditions on $r_{ij}(\lambda, \mu)$ the relaxation labeling process will converge to a consistent labeling while maximizing the average local consistency.

The **Average Local Consistency** of the assignment:

$$A(\bar{p}) = \sum_{i} \sum_{\lambda} p_{i}(\lambda)S_{i}(\lambda)$$
Relaxation labeling algorithm and N-Queens problem

Relaxation labeling algorithm was invented for the vision world. N-Queens problem is from the computer science world.
In this project we tried to solve a problem from one world with algorithm from other world.

Reduction of N-Queens to Relaxation labeling algorithm

- Objects: The queens are the objects of the algorithm, what means the algorithm will work with n objects.

- Labels: Queens's assignments are the labels. The labels that will be choose by the algorithm, will determine does the problem has consistent solution.

- $p^0_i(\lambda)$ Decision of initial probability divided to two groups: the first group contains only the first queen, and the second group contains all other queens.
  The label for the first queen that will get the highest probability is chosen randomly, will get probability of $(N+1)/2N$. All other labels for this queen, gets same probability of $1/2N$.
  Each queen from the second group will get some label from all labels that consistent with the first queen. The label will be chose randomly from the collection of consistent labels.

- $r_{ij}(\lambda, \mu)$ The value of $r_{ij}(\lambda, \mu)$ between queen i and queen j, calculated accordingly to the given constraints for N-Queens problem. If those two assignments are consistent, the value is 1, otherwise it 0.
- \( A(p) \) The stop condition of the algorithm. We say the solution is converges when the difference between current AverageLocalConsistency (of current iteration) and MaxAverageLocalConsistency less than 0.01 (this value was choose by "try and check" method).

**Queen’s assignment by probability**

How do we decide what will be the assignment for the queens?

For each queen we choose the best label. This label is the assignment this queen will get. Best label is chosen by maximal probability for each queen. In case that there are more than one label that got same probability, best label will be chosen randomly.
Results

Running of the simulator for different Ns (from N = 4 to N = 15), 100 problems for each.

- As N increases, the number of problems the algorithm solves decreases. The main reason for this is different purpose of the two problems (will be explained in "Problems" part).
- As N increases, the time to converge (or solving) the problem increases also.
Problems

1. Different purposes of two algorithms
   The purpose of N-Queens problem is to find solution, which will be consistent with the constraints of the problem. The algorithm should stop only when it finds at least one consistent solution.
   The purpose of Relaxation Labeling algorithm is to find some assignments of labels to objects, in such way that the solution will converge to max average local consistency.
   This difference causes to the relaxation labeling algorithm to stop (when it converges) even if current assignment of labels to queens doesn't make consistent solution (in terms of N-Queens problem).

2. In N-Queens problem, each object is affected by all other objects. In that case, the support for assignment of label to queen can be very high if it consistent with most of the queens, even if it doesn't consistent with all the queens.
   When the support is high, there is no reason for the relaxation labeling algorithm to change the probability for next iteration, and the problem can converge, but in terms of N-Queens the solution doesn't consistent.

3. Many possibilities for equal values and relaxation labeling algorithm don't handle this problem. We solved this problem by using the random method, but maybe it's not the best method.

4. Solution of the problem doesn't mean convergence for the relaxation labeling, so the algorithm will continue run, even if the solution was found.

5. Random in C# - not really random.
**Explanation of buttons:**

1. **Number of Queens** – represent the size of the chessboard.

2. **Start** – builds the first assignment as a result of $p_i^0(\lambda)$.

3. **New** - initialize the variables of the problem, needed when changing the number of queens.

4. **Next step** – runs one iteration of the algorithm. Shows on the board the results accordingly to current $p_i^{k+1}(\lambda)$.

5. **Run all** – runs the problems for many iteration. Stops when the solution is converge. Shows on the board the results accordingly to last $p_i^{k+1}(\lambda)$. Writes on the top of the board is the solution in consistent by N-Queens problem constraints.

6. **Statistic buttons:**
   6.1. **Run 50 problems** – runs N-Queens problem with given N (as written in the text box) for 50 times. Writes the results in file.
   6.2. **Run to all Ns** – runs N-Queens problem for different Ns (from 4 to 10), for each one 50 problems. Writes the results in file.
Conclusion

- For make a decision if relaxation labeling algorithm is good for the N-Queens problem, many another experiments should be done. For example, check another method for the $p_i^0(\lambda)$ values, find another solution for equal values, etc.

- Probably Relaxation labeling algorithm doesn’t the best choice algorithm for solving N-Queens problem. Solving it with backtracking methods gave much better results. But for being sure, many additional experiments are needed.

Reference

- On the Foundations of Relaxation Labeling Processes, by Hummel and Zucker 1983
- The N-Queens Problem
- Perceptual Organization (III), CS 202-1-5251, BGU, Ohad Ben-Shahar
- Demo for N-Queens by backtracking
- Article explain that random generator in C# doesn’t good