An Optimization Approach to the N-Queens Problem

Puning Zhao and Ruolan Zeng
Department of Electrical and Computer Engineering
University of California, Davis
One Shield Avenue, Davis, CA, 95616

***Abstract*— N-Queen problem is a traditional problem of placing n queens on a chessboard so that no two queens threaten each other. In this paper we propose an optimization-based method for finding the solutions of n-queens problem. Moreover, we proposed a statistical estimation method to determine the number of solutions m. The result of this estimation method is close to the true value Even if we found much less than m solutions. To the best of our knowledge, this is the first attempt for such an estimation.**

I. INTRODUCTION

Eight-Queen problem is the problem of placing eight chess queens on an 8×8 chessboard so that no two queens threaten each other. Thus, a solution requires that no two queens share the same row, column, or diagonal. Franz Nauck published the first solutions in 1850. [1]. This problem can be extended to the more general n-queens problem of placing n non-attacking queens on an $n \times n$ chessboard, for which solutions exist for all natural numbers n with the exception of n=2 and n=3. Since then, many mathematicians, including Carl Friedrich Gauss, have worked on both the eight queens puzzle and its generalized n-queens version.

When the n-queens problem were first proposed, it was recognized as only a 'mathematical recreation'. However, in recent years, this classical problem has more and more practical applications, including parallel memory storage schemes, VLSI testing, traffic control, deadlock prevention, etc [3]. Therefore, a in-depth investigation of this classical problem is becoming increasingly important.

The existing methods can be divided into three categories.

1) Backtracking algorithm: The idea of backtracking algorithm is to place one queen on one edge and then continue by placing the next queen on the first valid position (in the next row or column) and so on. When no more queens can be placed the algorithm has either found a solution (if all queens are placed) or it needs to remove the processed queen and move the previous one to the next valid position. When a queen has been placed on the last valid position in a row or column and needs to be replaced it must be removed and the previous one must be moved to the next valid position. The advantage of this algorithm is that this algorithm can find all the solutions. However, a clear disadvantage is that it can not be completed in polynomial time. Finding one solution requires exponential time complexity, while finding all the solutions requires $O(n!)$ time complexity. Researchers have proposed some new methods to reduce the running time of this algorithm. However, the time complexity can not be reduced.

2) Systematic search and greedy algorithm: The idea is to place the queens in the best possible spots on each occasion or processed line. That means that it may also place the queens on non-valid locations. Queens need to be initialized before moved and ensured that each line has just one. Adjusting the starting positions may be done randomly or by following a pattern (like placing all the queens in the same row or column). This algorithm will not find a solution on every try. In that case the queens need to be reinitialized in a different way and the whole process to be repeated. Comparing with the classical backtracking algorithm, this algorithm is much faster. The time complexity of finding one solution becomes polynomial. However, greedy search may not be able to find all the solutions. Some of the existing solutions can never be found. This drawback is serious if we want to conduct statistical estimation of the total number of solutions.

3) Random permutation algorithm: This algorithm starts from solving a easy problem, that is, n-roots problem, which means that placing n roots so that no two roots attack each other. It is easy to show that this method has $O(n!)$ solutions. We can then randomly choose a solution, which is actually a random permutation. Then there may still exists two queen that threat each other. We can find those diagonal lines that has more than one queens, and define a loss function as the total number of queens that is located on these diagonal lines. Among all of these diagonal lines we randomly choose one and then choose a queen on this diagonal line. This queen can be permuted with other queen that do not lie in this diagonal. Then, this permutation is done if it decreases loss function, and canceled if it increases loss function. Such a random permutation is faster then the systematic search and greedy algorithms, and all the existing solutions have chance of being found by this algorithm. However, the probability distribution is highly non uniform, and thus may not be a good choice for statistical estimation.

In this paper, we propose an optimization based approach to find the solutions to n-queens problem. The chessboard can be regarded as an $n \times n$ matrix, and the positions of queens can be regarded as an permutation matrix. We define a loss function over this matrix. The optimization over permutation matrices is non-polynomial, therefore we relax the permutation matrix to the doubly stochastic matrix, and use a modified version of random coordinate descent method to find the solutions. With this method, all of the solutions have nearly equal chance of being found. This important property enables us to conduct the statistical estimation of total number of solutions. We also propose our estimation method, and experiments show that the result of our estimation is close to the real value.

Organization. The rest of this paper is organized as follows: in the second section we introduce the matrix model of our problem and the statistical model of the solutions. In the third section we give our problem formulation and the experimental results, and the final section is about the conclusion and future work.

II. System Model

A. Matrix model of N-Queens problem

In this problem, the chessboard is modeled as an $n \times n$ matrix. Each element of the matrix represent one point on the chessboard. After the queens are placed on the chessboard, the value of matrix element is 1 if the corresponding location is occupied by a queen, and 0 if no queen occupies it. As a result, if the queens are correctly placed on the chessboard, the matrix should be a permutation matrix, which is defined as the matrix that there are one and only one element with value 1 in each row or each column. The values of all other elements are 0. All the solution matrices are permutation matrices, but only part of permutation matrices are correct solutions, because permutation matrix only ensure that each two queens are not placed on the same column or the row, without ensuring there are no two queens in the same diagonal line.

Besides, the solutions of N-Queens problem can be expressed as a data array, in which the number represents the location in each column. For example, the following solution of eight queen problem is '15863724'.

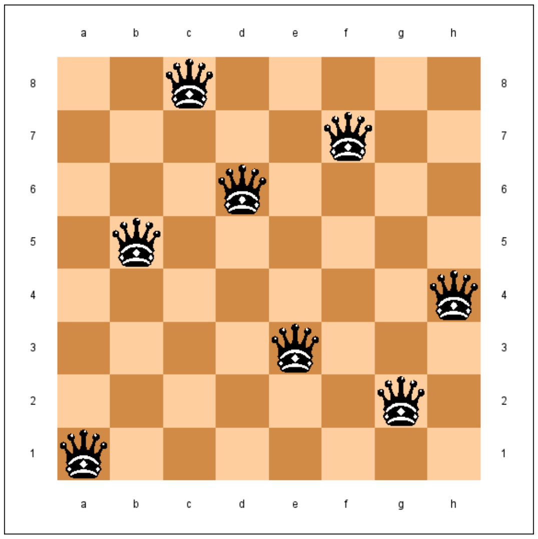

Fig. 1. One solution of Eight-Queens Problem

B. Statistical model of solutions

We can run our method for k times, at obtain k results. Each result is one of the solution for N-Queens problem. All of our results are identical and independently distributed among all the solutions. We use X to denote this random variable, and $x_i = j$ if the output of the $i-th$ run is the $j-th$ solution of the N-Queens problem. The total number of solutions for the problem with $n \times n$ chessboard is denoted as $m(n)$. The probability of finding each solution is denoted as $p_1, ..., p_m$, respectively. After we run k times, the number of occurrence of each solution follows a multinomial distribution. The *empirical distribution* is defined as the *histogram* of X:

$$k_j = \sum_{i=1}^{k} \mathbf{1}\{x_i = j\} \; j = 1, \ldots, m. \tag{1}$$

The *fingerprint* of the distribution can be regarded as the *histogram of histogram* of the random variable X, which is defined as

$$F_i = \sum_{j=1}^{m} \mathbf{1}\{k_j = i\} \; i = 1, \ldots, k. \tag{2}$$

III. Problem Formulation and Results

A. Problem statement and relaxation

Our problem can be formulated as the following maximization problem:

$$\begin{aligned} \underset{P}{\text{maximize}} \quad & L(P) \\ \text{subject to} \quad & P \in \mathscr{P}^* \end{aligned} \tag{3}$$

In which $\mathscr{P}^*$ denotes the set of permutation matrix. This set is not a convex set. It can take $O(n!)$ discrete values for a $n \times n$ matrix. Therefore, the optimization over this set is non polynomial. In order to reduce the computation cost, we relax the permutation matrix to doubly stochastic matrix, which is defines as:

Definition 1: A square matrix is *doubly stochastic* if all of its elements are non negative, and the sum of all the elements in each row and column equals to 1.

Clearly all the convex combinations of permutation matrices belong to doubly stochastic matrices. Therefore, the convex hull of the set of permutation matrix *belong to* the set of doubly stochastic matrix, which is denoted as $\mathscr{P}$, i.e. $\textbf{conv}(\mathscr{P}^*) \subset \mathscr{P}$. Moreover, we have the following theorem:

Theorem 1: (**Birkhoff Theorem**) Every doubly stochastic matrix is the convex combination of permutation matrices.

Combining these two arguments, we know that the convex hull of the permutation matrices is exactly the set of doubly stochastic matrices. Therefore, the set of doubly stochastic matrices is a reasonable relaxation to permutation matrices. The problem can then be formulated as:

$$\begin{aligned} \underset{P}{\text{maximize}} \quad & L(P) \\ \text{subject to} \quad & P \in \mathscr{P} \end{aligned} \tag{4}$$

B. Objective function

The next important problem is the definition of function L. Since our desired matrix is the permutation matrix in which no two elements with value 1 are at the same diagonal line, therefore we hope to design a function, so that given any doubly stochastic matrix, the function is able to measure the difference between this matrix to our desired matrix. Our tentative function is defined as:

$$L = \sum_{j=1}^{n} A_j + \sum_{i=1}^{n} B_i + \sum_{l=-(n-1)}^{n-1} C_l + \sum_{l=-(n-1)}^{n-1} D_l, \tag{5}$$

in which

$$\begin{aligned} A_j &= \sum_{j=1}^{n} P_{ij}^2 \\ B_i &= \sum_{i=1}^{n} P_{ji}^2 \end{aligned} \tag{6}$$

For $i = 1, \ldots, n$, and

$$\begin{aligned} C_l &= \begin{cases} 1 - \sum_{i=1}^{n-l} P_{i,i+l} & \text{if} \quad \sum_{j=1}^{n} P_{i,i+l} > 1 \\ 0 & \text{if} \quad \sum_{i=1}^{n} P_{i,i+l} \leq 1 \end{cases} \\ D_l &= \begin{cases} 1 - \sum_{i=l+1}^{n} P_{i,i-l} & \text{if} \quad \sum_{j=1}^{n} P_{i,i-l} > 1 \\ 0 & \text{if} \quad \sum_{i=1}^{n} P_{i,i-l} \leq 1 \end{cases} \end{aligned} \tag{7}$$

For $l = -(n-1), \ldots, n-1$.

In the above definition, A and B refers to the deviation in rows and columns, respectively, each A_j or B_i reach their maximum when there are only one element with value 1, and other elements has value 0. If we only have all the A and B in function L, without C and D, all the permutation matrices will have the same function value. As a result, the final output of our method can be any permutation matrix, and there may still exists two queens threatening on a diagonal line. Now we introduce C and D, which control the value of diagonal line. It is 0 if the sum of the values on this diagonal is less than or equal to 1. Otherwise C and D will be negative, therefore it penalizes the problem of diagonal threatening. It is trivial to prove that all the *solution matrices*, which is the permutation matrix that satisfy the condition of no two elements with value 1 are on the same diagonal line, have the same function value. And the function of all the other non-solution matrices is strictly less than the function value of solution matrices. Therefore, as long as the function reaches its optimal, it must be one of the solutions of N-Queens problem.

C. Optimization Algorithm

A property of doubly stochastic matrix is that for each (i_1, j_1) and (i_2, j_2), in which $i_1 \neq i_2$, $j_1 \neq j_2$, if we conduct the following transformation, the matrix is still a doubly stochastic matrix:

$$\begin{aligned} P(i_1, j_1) &\leftarrow P(i_1, j_1) + \delta \\ P(i_2, j_2) &\leftarrow P(i_2, j_2) + \delta \\ P(i_1, j_2) &\leftarrow P(i_1, j_1) - \delta \\ P(i_2, j_1) &\leftarrow P(i_2, j_2) - \delta \end{aligned} \tag{8}$$

As long as the all the function value after transformation still lie in interval $[0, 1]$. This important property enables us to optimize the *local objective function* for given (i_1, j_1) and (i_2, j_2).

In our algorithm, we firstly initialize an uniform matrix. The value of each element is $\frac{1}{n}$. The following table shows the initial matrix for $n = 4$. Then we select the coordinate

TABLE I
INITIATION(N=4)

0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25

(i_1, j_1) and (i_2, j_2) randomly, and then find δ to maximize the objective function. The method of finding optimal δ is to evaluate the increase of objective function. The loss function has, in total, $6n - 2$ terms, including n terms of A and B respectively, and $(2n - 1)$terms of C and D respectively, representing $(2n-1)$ diagonal lines. Among all these terms, only the rows, columns or diagonals on which elements (i_1, j_1) and (i_2, j_2) lie in, can contribute to the change of objective function, other term will remain the same after this transformation. We call these terms *local objective function*. With the optimization only for the local objective function, the time complexity of one operation of the transformation in (8) can be $O(1)$, which means that the time is irrelevant to n, because there are at most two rows, two columns and four diagonal lines that need to be updated. The detail of our algorithm can be written as follows: In the above algorithm U and V denotes the sum of diagonal lines. The local function value on the diagonal values C and D can be easily calculated. Since the objective function is non-convex, we can probably get a local optimal solution instead of a global optimal solution. A method to solve this problem is to find the maximum value in each row of elements. As long as all the maximum value is larger than 0.5, and all the sum of diagonal elements less than or equal to one, we can replace those maximum values with 1 and other elements with 0. Then, under these prerequisites, we can definitely get a solution matrix, because the value larger than 0.5 suggests that no other elements in the same row, column or diagonal is larger than 0.5, since the sum of values in the all the rows and columns are 1, according to the definition of doubly stochastic matrix, and the prerequisites that $U_i \leq 1$ and $V_i \leq 1$. As a result, if it is replaced by 1, then all other

```
Initialize: $P_{ij} = \frac{1}{n}$ for all $i$ and $j$;
$A_i = \frac{1}{n}, \forall i = 1, \ldots, n$;
$B_j = \frac{1}{n}, \forall j = 1, \ldots, n$;
$U_l = \sum_{i=1}^{n-l} P_{i,i+l}, l = -(n-1), \ldots, n-1$;
$V_l = \sum_{i=l+1}^{n} P_{i,i-l}, l = -(n-1), \ldots, n-1$;
While certain stopping criterion not satisfied
randomly pick $(i_1, j_1)$ and $(i_2, j_2)$, in which $i_1 \neq i_2$,
  $j_1 \neq j_2$;
find optimal $\delta$;
update matrix P;
end
Find the maximum value and the corresponding
  location of each column;
If all the maximum value are larger than 0.5, and $\forall l$,
  $U_l \leq 1$, $V_l \leq 1$
Replace those maximum values with 1, other
  elements with 0;
return this solution;
else
Repeat from beginning.
```

elements in same row, column, or diagonal are replaced by 0. This holds for all the column maximum values, thus no two elements with value 1 lie in same row, column or diagonal in the output matrix. The following table shows an example of the doubly stochastic matrix after one operation:

TABLE II
THE MATRIX AFTER FIRST OPTIMIZATION OPERATION, WITH δ(N=8)

0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
0.125	0.25	0.125	0.125	0.125	0	0.125	0.125
0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
0.125	0	0.125	0.125	0.125	0.25	0.125	0.125
0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125

D. Estimating the number of solutions

There is currently no known formula for the exact number of solutions, or even for its asymptotic behaviour. Table 1 and table 2 gives the number of solutions for placing n queens on an n×n board, for n=1−10, 26−27.

TABLE III
THE NUMBER OF SOLUTIONS FOR N-QUEENS PROBLEM(N=110)

n	1	2	3	4	5	6	7	8	9	10
solutions	1	0	0	2	10	4	40	92	352	724

Currently the 27×27 board is the highest-order board that has been completely enumerated.

TABLE IV
THE NUMBER OF SOLUTIONS FOR N-QUEENS PROBLEM(N=26-27)

n	26	27
solutions	22,317,699,616,364,044	234,907,967,154,122,528

In our work, instead of calculating the exact numbers of solutions, we use a statistical method to estimate it. It is based on statistical method in [2]. In 2013, Paul Valiant and Gregory Valiant showed that a class of distributional properties. Usually this estimator is used to estimate entropy, and the support size, i.e. the number of distinct elements. The estimator is efficient in the sense that given a sample consisting of independent draws from any distribution over at most n distinct elements, these properties can be estimated accurately using a sample of size $O(m/\log m)$. their estimator is theoretically optimal (to constant factors, over worst-case instances. This estimator can be used to estimate the number of solutions. Recall that the output of our optimization algorithm is the identical and independently distributed samples, with probability $p_i, i = 1, \ldots, m$ for finding the ith solution. After we get some outputs, the histograms and the corresponding fingerprints can be calculated, then the estimation method will calculate the approximate numbers of solutions using Poisson sampling, to find the distribution that is most likely to generate the obtained fingerprint.

Our algorithm can be very fast for finding the total number of solutions. Each solution can be represented as a integer number, for example, the solution matrix in VI can be represented by '2413', in which each digit represents the location of element 1 in each column. After we get k solutions, we can sort the associated number, which only take $O(k \log k)$ time, and then calculate the number of repetitions for each solution. Since our optimization algorithm has an advantage that all the solutions have chance of being obtained, and experiments show that the probability distribution is nearly uniform, this estimation method can generate a closely estimated number of total number of solutions.

TABLE V
EXPECTED OUTCOME(N=4)

0	0	1	0
1	0	0	0
0	0	0	1
0	1	0	0

E. Experiment Results

Experiment shows that our algorithm is much faster than traditional backtracking algorithm. For the classical Eight-Queen problem, we can find 100 solutions in about 3 seconds, with a PC whose CPU is 2.7GHz. Here we list part of our results: 31758246, 36418572, 62714853, 26174835, 64718253, 42857136, 42736851, 83162574, 42857136, 52468317, 26174835. Each digit of each number represents the location of queens in each column.

For the estimation of total number of solutions, we run our algorithm for $n = 8, 9, 10, 11$. When n increases, the ratio of the required number of obtained solutions become much less than the real number of solutions, without too much harmful effect on the accuracy of our prediction. Intuitively, the accuracy of the estimation method relies on the number of repetitions. If each solution only occured once, then it is very hard to estimate the number of solutions. However, when the number of trials increase, it becomes likely that we can get repetitions, which will make our prediction accurate. The result is listed in the following table, which shows that the result of our prediction is close to the real number of solutions.

TABLE VI
PREDICTIONS FOR NUMBER OF SOLUTIONS

n	8	9	10	11
trials	200	300	500	1000
predictions	99	346	872	2450
real number of solutions	92	353	724	2680
prediction / realities(%)	107.6	98.0	120.4	91.4

IV. CONCLUSION

We have two main contributions. For one thing, we designed an optimization based approach to n queens problem, which can be formulated as the maximization of a function over the set of permutation matrices. In order to reduce the time complexity, we relaxed the support set to doubly stochastic matrices. The outcomes of our algorithm are identical and independently distributed random samples of all the solutions. Experiments show that our algorithm is efficient.

For another, we proposed a method for statistical estimation of the total number m of solutions based on the results of our optimization approach. Since the outcomes are samples of all the solutions, we can use the fingerprint of the empirical distribution as a sufficient statistic, to estimate m. The sample complexity of such an estimation is $O(m/logm)$, which means that we can get a close estimate of the value even though the number of obtained solutions is much less than m. Our algorithm of the second operation is very time efficient, with only $O(klogk)$ time complexity, in which k is the number of obtained solutions. The error comparing to the true number is not too large.

There are several new interesting topic to study. Firstly, although our algorithm practically has polynomial time complexity, we need theoretical guarantees. Secondly, the algorithm can still be much improved. In our optimization algorithm, when the optimization process is about to finish, most of the matrix has been locally optimized, which means that they are composed mainly of 0 and 1, and the sum of the values on most of the diagonal lines is less than or equal to 1. As a result, we waste most of our running time on these locally optimized areas. Therefore, a natural idea of future improvement is to design a dynamic random coordinate selection rule, so that those diagonals with the sum of values larger than 1 have higher probability of being selected. This will significantly reduce the running time. Finally, the estimation of total number of solutions can also be improved, if we conduct a further analysis of the probability mass function of our outcome of the optimization algorithm. What Valiant has proposed is suitable for all the distributions, and we can design an estimation method that is suitable for our case.

V. DESCRIPTIONS OF THE TEAM MEMBER DUTIES

- System Model and algorithm by Puning Zhao and Ruolan Zeng.
- MATLAB program and results by Puning Zhao.
- Final report paper by Puning Zhao and Ruolan Zeng.

REFERENCES

[1] Ball, WW Rouse. ”The eight queens problem.” Mathematical Recreations and Essays, New York: Macmillan (1960).
[2] Valiant, Paul and Valiant, Gregory.,Estimating the Unseen: Improved Estimators for Entropy and other Properties.Curran Associates, Inc,Vol. XX (2013), pp. 21572165.
[3] Bell, Jordan, and Brett Stevens. ”A survey of known results and research areas for n-queens.” Discrete Mathematics 309.1 (2009): 1-31.
[4] Fogel, Fajwel, et al. ”Convex relaxations for permutation problems.” Advances in Neural Information Processing Systems. 2013.
[5] Sosic, Rok, and Jun Gu. ”A polynomial time algorithm for the n-queens problem.” ACM SIGART Bulletin 1.3 (1990): 7-11.