CUCS-129-84

Construction Through Decomposition: A Linear Time Algorithm for the N-queens Problem

Bruce Abramson and Mordechai M. Yung

Department of Computer Science
Columbia University
New York, N.Y. 10027

Abstract

Configuring N mutually non-attacking queens on an N-by-N chessboard is a contemporary problem that was first posed over a century ago. Over the past few decades, this problem has become important to computer scientists by serving as the standard example of backtracking search methods. A related problem, placing the N queens on a toroidal board, has been discussed in detail by Polya and Chandra. Their work focused on characterizing the solvable cases and finding regular solutions, board setups that solve the problem while arranging the queens in a regular pattern. This paper describes a new linear time divide-and-conquer algorithm that solves both problems. When applied to the toroidal problem, the algorithm yields a family of non-regular solutions based on number factorization. These solutions, in turn, can be modified to solve the standard N-queens problem for board sizes which are unsolvable on a torus.

> The problem of the eight queens is a well known example of the use of trial-and-error methods and of backtracking algorithms. It was investigated by C.F. Gauss in 1850, but he did not completely solve it. This should not surprise anyone. After all, the characteristic property of these problems is that they defy analytic solution. Instead, they require large amounts of exacting labor, patience and accuracy. Such algorithms have therefore gained relevance almost exclusively through the automatic computer, which possesses these properties to a much higher degree than people, even geniuses, do.
>
> *-Nicklaus Wirth, Algorithms + Data Structures = Programs, p.143.*

1. Introduction

The preceding quote typifies the prevailing view of the N-queens problem. First posed by Max Bezzel in 1848, the problem of placing eight queens on an eight-by-eight chessboard in mutually nonattacking positions was solved in 1850 by Franz Nauck, who used trial-and-error methods to find 12 solutions. In 1874, Dr. S. Gunther proved Nauck's list exhaustive and Dr. J.W.L. Glaisher generalized the problem to placing N queens on an N-by-N board. Gunther and Glaisher proposed the following solution: Represent the board as a symbolic N-by-N matrix. Certain easily recognizable terms in this matrix' determinant indicate solutions to the N-queens problem. This approach, although helpful in the 8-by-8 case, is actually a brute force search on the factorially many terms in the determinant, and thus takes O(N!) time. For details of the problem's early history see [2] [6] [14].

The inherent difficulty of the N-queens problem is generally accepted as fact; work done on it in this century has taken one of two approaches. The first approach accepts the problem's intractability and uses it as an example of trial-and-error methods like backtracking. The second modifies the problem to one that does not defy analytic solution. Examples of the former approach abound. Various fields, including algorithm design [10] [20] [5], program development [4] [19], and artificial intelligence [15] (chapters 1,2) [8] [9] have relied on the N-queens problem to illustrate issues related to constraint satisfaction and systematic heuristic search. As for the latter approach, analytically solvable variations of the N-queens problem can be grouped into two broad categories: those that reduce constraints (e.g. see [18]) and those that increase them (e.g. see [3]).

This paper presents a family of solutions to the N-queens problem. The method used in constructing this family involves points of interest to both

approaches. In terms of heuristic search methods, it illustrates a way in which the board can be decomposed. Decomposition is always at a premium when search is involved because decomposing a search space saves both time and space [17] (chapter 7). The basis of the solution, however, lies in solving a related analytic problem, placing N-queens on a toroidal board. Previous work on this variation focused on ***regular solutions***, setups that are obtained by placing queens on the board in a regular pattern [3]. In this paper, a new, non-regular family of solutions to this toroidal variation is developed. This non-regular family is then modified to yield a general analytic solution to the N-queens problem. The N-queens solutions presented here are not the first. Yaglom and Yaglom [21] gave a solution, which, although thirty years old, is relatively unknown. This older solution has two major shortcomings. One, it is problem-specific, that is, it fails to link the original problem to any analytically understood relatives. Two, it is not based on a general algorithmic technique (e.g. recursive decomposition), and therefore has no implications to algorithmic or heuristic techniques for solving constraint satisfaction problems. A sketch of their solution can be found in appendix 3.

The rest of this paper proceeds as follows: section 2 discusses the N-queens problem as an example of backtracking techniques and constraint satisfaction problems, (shown to be general case NP-complete in appendix 1), while section 3 introduces and outlines previous work done on the analytically solvable toroidal problem. Section 4 then introduces a non-regular family of solutions to the toroidal problem, which leads to the general solution to the planar problem outlined in section 5. Section 6 offers a complexity analysis of the algorithm presented and section 7 mentions some conclusions and directions for possible future work. Along the way, some light is shed on the enumeration problem: a lower bound and a partial ordering on the number of toroidal solutions are shown.

2. Solutions Based on Search Techniques

The standard technique used to solve the N-queens problem is backtracking search [10]. Backtracking, which is frequently applied to ***constraint satisfaction problems***, is a worst case exponential method, and is therefore usually supported by heuristics [8] [15] (chapters 1,2) [12]. Any problem that involves assigning values to variables subject to a set of binary constraints is an example of constraint satisfaction. The constraints can be thought of as sets of mutually inconsistent

<variable,value> pairs. An assignment is a solution if every variable is assigned a value and no two mutually inconsistent <variable,value> pairs appear in the assignment. The popularity of the N-queens and other chessboard related problems as examples of constraint satisfaction can be attributed to two factors: the long history of the chessboard and the relative simplicity with which problems related to it can be stated. In the N-queens problem, rows can be regarded as variables and columns as values. Given the board size, N, as input, the <variable,value> pairs can be expressed as a permutation P of 0 to N-1, where P(i) is the column of the queen in the i^{th} row. This representation alone is enough to guarantee that no two queens will be in the same row or column, leaving only the diagonal constraints to be verified. Since the diagonals going from top left to bottom right can be characterized by (i-P(i))=constant, and those from top right to bottom left by (i+P(i))=constant, P is a solution if and only if for $i \neq j$, $(i-P(i)) \neq (j-P(j))$, and $(i+P(i)) \neq (j+P(j))$.

The general constraint satisfaction problem involves finding an appropriate assignment given sets of variables, possible values, and binary constraints. This problem is provably NP-complete, and ergo likely to be difficult. For a formal problem definition and the intractability proof, see appendix 1. As is usually the case, most problems of interest are not random instances of general constraint satisfaction, but special, structured cases of it, in which the constraints follow specific patterns. The N-queens problem, for example, can be viewed as a special case of independent set, a known NP-complete problem [7]. To see this, view the board as a graph with a vertex for each square and an edge between any two vertices representing squares in the same row, column or diagonal. A solution to the N-queens problem is a set of N squares sharing neither row, column, nor diagonal, or a set of N independent vertices.

The interesting, structured, constraint satisfaction family includes relatives of many NP-complete problems [7], as well as various hard problems of unknown complexity in computer vision and artificial intelligence [9] [8]. There are no known efficient algorithms to most of these problems. One reasonable source of this difficulty is the global propagation and interrelation of constraints. Because of this "globality factor", each <variable,value> assignment radically alters the set of possible values that can be assigned to all other variables. In general, a great deal of bookkeeping is required to keep track of all the constraints. This difficulty is

reflected in the fact that all common solutions are forms of heuristic search. The heuristics trade certainty for computational feasibility by reducing the amount of bookkeeping done. Standard backtracking does no complex bookkeeping; values are systematically assigned to variables until either a dead end or a solution is reached [10]. There are, however, several variations of backtracking that do some rather involved bookkeeping. The general aim of these variations is to minimize redundant checking of the validity of newly assigned <variable,value> pairs, speeding up the search at the cost of a modest amount of additional storage [8]. Forward checking and other look-ahead algorithms also rely on partial bookkeeping. No variable is assigned a value until the procedure first checks to make sure that it leaves at least one possible value assignment for each remaining variable [9].

3. The Toroidal Problem and Regular Solutions

Several interesting variations to the N-queens problem have been proposed. One that has attracted a good deal of attention is the N-superqueens problem. A ***superqueen***, introduced in [16], is a queen which upon reaching an edge of the board can wrap around to the opposite edge, in effect treating the board as a torus. The superqueen places additional constraints on the board by connecting previously separate diagonals. The resulting (toroidal) board has N rows, N columns, and two sets of N diagonals (characterized by (row-column)=constant mod N and (row+column)=constant mod N), each containing N squares. This symmetry makes the N-superqueens problem easier to analyze combinatorially than the N-queens. Polya [16] showed that an N-superqueens solution exists if and only if N is not divisible by 2 or 3. Since the N-superqueens is an overconstrained variation, any N-superqueens solution solves the N-queens as well.

Chandra [3] developed the theory of ***independent permutations*** which he used to characterize a family of solutions to the N-superqueens problem, the ***regular solutions***. A regular solution is one in which the permutation P can be characterized by P(i)=Ai+B (mod N). The permutation P(i)=2i, for example, starts with a superqueen in the upper left hand corner and proceeds around the board placing queens one row down and two columns across. This solution is called the ***knight-walk***. Chandra also built on the result of [16] by showing that for any N, if the N-superqueens problem is solvable and M is the smallest factor of N (M > 1), a family of regular solutions exists, and is characterized by P(i)=Ai,

A=2,4,...2^k, where k= [$\log_2$M] -1. Since P(0)=0, each member of this family has a superqueen in the upper left hand corner. Only regular solutions exist for N < 13. For N=13, however, the non-regular permutation P=(0 3 8 11 5 1 10 4 7 12 2 9 6) is a solution as well. Some regular extensions of the toroidal problem are discussed in [3] [11].

As an immediate extension of their existence for any solvable N-superqueens problem, the family of regular solutions described above solves the N-queens problem for 2/3 of the odd numbers. Furthermore, removing the top row and leftmost column from the N-by-N board deletes only the queen in the top left hand corner square. This leaves an (N-1)-by-(N-1) board with (N-1) mutually nonattacking queens, solving the (N-1)-queens problem, thereby constructing solutions for 2/3 of the even numbers. No other simple board modifications are possible, since removal of any other row and column would either delete two queens, shift the diagonals, or both. Regular solutions and upper left hand corner-removal, then, constitute a solution for 2/3 of all N.

4. The Decomposition Solution to the Superqueens Problem

This section shows how to construct a family of non-regular solutions to the N-superqueens problem, the decomposition solutions. Decomposition uses the factorization of N to apply a divide-and-conquer approach to the problem. Basically, if N can be factored as AB where both A and B are solvable, the N-superqueens problem can be reduced to solving A appropriately chosen copies of the B-superqueens problem.

Definition: *Let N=AB, where there are solutions to the A-superqueens and B-superqueens problems. A decomposition solution breaks the N-by-N board into an A-by-A grid of B-by-B tiles. Tiles corresponding to an A-superqueens solution are filled with a B-superqueens solution - the same B-superqueens solution is used throughout.*

Before formally proving that a decomposition solution as defined above solves the problem, look at the example in figure 1. In order to simplify further discussions, the following definitions will be used:

Definition: *Let LR(N) (RL(N)) refer to the set of diagonals running from top*

left to bottom right (top right to bottom left) on an N-by-N board. Let OLR(N) (ORL(N)) refer to those members of LR(N) (RL(N)) which are occupied by a queen. Similarly, let WLR(N) (WRL(N)) refer to the set of N wrapped diagonals on a planar representation of an N-by-N toroidal board running from top left to bottom right (top right to bottom left). Let OWLR(N) (OWRL(N)) refer to those members of WLR(N) (WRL(N)) which are occupied by a superqueen.

In figure 1, N=35, A=7, and B=5. The knight-walk is used as both the 5 and 7-superqueens solutions. A board set up according to the definition of decomposition clearly contains 35 superqueens, with no two in any one row or column. That no diagonal contains two superqueens, however, is a bit less obvious. The trick here is to realize that the diagonals that result from tiling a plane with 5-by-5 boards are equivalent to those resulting from placing a 5-by-5 board on a torus. As shown in the diagram, a diagonal in WLR(35) passes alternately through members of LR(5) of lengths 2 and 3. These are exactly the LR(5) diagonals which combine to form a single element of WLR(5). The use of a 5-superqueens solution guarantees that only one of these LR(5) diagonals contains a superqueen. This reduces the realm of possible conflicts to superqueens in the same position on different boards. Examination of the figure reveals that if a member of WLR(35) passes through corresponding LR(5) diagonals on two 5-by-5 tiles, the tiles lie along the same LR(7) diagonal. The use of a 7-superqueens solution precludes the possibility of two such boards being chosen, and thus no two superqueens can be in the same member of WLR(35). Thanks to the symmetry of the board, an identical argument can be used for WRL(35).

Theorem 1: A decomposition solution solves the N-superqueens problem.

As a prerequisite for proving this theorem, some machinery must be developed for referring to individual squares on a decomposed board. Each square's location can be specified in two coordinate systems: the N-by-N system and the B-by-B system within the A-by-A grid. In converting between the systems, let $<i_N,j_N>=$ tile $<i_A,j_A>$ square $<i_B,j_B>$, where $i_N,j_N=0$ to N-1, $i_A,j_A=0$ to A-1, $i_B,j_B=0$ to B-1. Since superqueens solutions are used throughout, all boards must be treated as tori, and therefore, all arithmetic is done modulo the subscripted number. The relationship between the systems is simple. Each tile contains B rows and B

columns. Thus, $i_N=Bi_A+i_B$ and $j_N=Bj_A+j_B$. In other words, to determine the location of a square on the N-by-N board, find out which tile it's in, multiply each tile index by B to determine how many rows or columns preceded it, and add the number of rows or columns preceding it in its tile. Furthermore, since rows and columns within a tile are counted modulo B, the row following the $(B-1)^{st}$ in any tile is the 0^{th} row of the following tile. The same is true of columns. In the example of figure 1, where N=35, A=7, and B=5, if $<i_N,j_N>=<14,23>$, the square in question is tile $<2,4>$ square $<4,3>$. Moving one square to the right, $<15,23>$ is tile $<3,4>$ square $<0,3>$. Two squares *correspond* if they have the same B coordinates. In the example of figure 1, $<7,4>$= tile $<2,1>$ square $<2,4>$ and $<12,14>$= tile $<3,3>$ square $<2,4>$ correspond with each other.

Proof (Theorem 1):

A board configuration is an N-superqueens solution if and only if there are N superqueens on the board, one in each row, column, WLR and WRL diagonal.

Claim 1: A board set up as specified by the decomposition solution contains exactly N superqueens, one in each row (column).

Proof (Claim 1): By the definition of decomposition, A copies of the B-superqueens solution are used, totalling AB=N superqueens. These copies are placed in tiles that correspond to an A-superqueens solution, partitioning the rows into blocks of size B. The use of a B-superqueens solution implies an even distribution throughout each block, one superqueen per row. Thus, all N rows are covered. By a simple pigeonhole principle argument there must be only one superqueen in each row. Symmetrically, the same holds for columns.

Claim 2: Each member of OWLR(N) (OWRL(N)) contains one superqueen.

Proof (Claim 2): There can't be a conflict between two superqueens on the same tile because any occupied tile contains a B-superqueen solution. The only possible conflicts in OWLR(N), then, are between superqueens on different tiles. In order to prove that conflicts of this nature are impossible as well, it is important to consider the LR(B) diagonals that combine to form an OWLR(N) diagonal.

Consider the OLR(B) diagonal of tile $<i_A,j_A>$ with a superqueen in square $<i_B,j_B>$. As mentioned before, this diagonal can be represented by the invariant (i_B-j_B). If $i_B < j_B$, the diagonal runs from the tile's top (square $<0,j_B-i_B>$) to its

right side (square $<i_B\text{-}j_B\text{+}B\text{-}1,B\text{-}1>$). The OWLR(B) diagonal of tile $<i_A,j_A>$ containing the OLR(B) diagonal in question is characterized by ($i_B\text{-}j_B$ (mod B)). This OWLR(B) diagonal starts with the OLR(B) diagonal, and then wraps around to $<i_B\text{-}j_B,0>$, down to $<B\text{-}1,B\text{-}1\text{+}j_B\text{-}i_B>$, and back up to $<0,j_B\text{-}i_B>$. The OWLR(N) diagonal containing this OLR(B) diagonal takes a course through corresponding squares. Starting with the OLR(B) diagonal, it continues rightward from tile $<i_A,j_A>$ square $<i_B\text{-}j_B\text{+}B\text{-}1,B\text{-}1>$ to tile $<i_A,j_A\text{+}1>$ square $<i_B\text{-}j_B,0>$. This square corresponds to the leftmost square of the wrapped portion of the OWLR(B) diagonal. Thus, the use of a single B-superqueens solution throughout guarantees the absence of superqueens from the portion of the OWLR(N) diagonal in this tile. The OWLR(N) diagonal then continues through tile $<i_A,j_A\text{+}1>$ square $<B\text{-}1\text{+}j_B\text{-}i_B,B\text{-}1>$ down to tile $<i_A\text{+}1,j_A\text{+}1>$ square $<0,j_B\text{-}i_B>$. This square corresponds to the top square of the original OWLR(B) diagonal. Tile $<i_A\text{+}1,j_A\text{+}1>$ however, lies along the same member of OWLR(A) as $<i_A,j_A>$, insuring that it was not used in the decomposition.

It can easily be shown by induction that this pattern continues. The wrapped diagonal of OWLR(N) alternately passes through tiles disqualified by OWLR(A) and diagonals disqualified by OWLR(B). By symmetry, the same argument holds for $i_B > j_B$. If $i_B = j_B$, the tile diagonal in question is the main diagonal, which only passes through tiles disqualified by the A-superqueens solution. Thus, no member of OWLR(N) contains more than one superqueen. By the pigeonhole principle, OWLR(N) = WLR(N), and symmetrically, OWRL(N) = WRL(N). Thus, decomposition constitutes an N-superqueens solution. QED.

Every ordered split of N into its (not necessarily prime) factors corresponds to at least one decomposition solution. Given some such split, $N=f_1f_2...f_k$, the first factor plays the role of A, breaking the N-by-N board into a grid of (N/f_1)-by-(N/f_1) tiles. Any solution to the f_1-superqueens problem can be used to choose tiles. The rest of the factors are used recursively to fill the tiles specified by the chosen f_1-superqueens solution. Ergo, if M is a proper divisor of N, any M-superqueens solution gives rise to at least one N-superqueens solution. Since there is at least one N-superqueen solution that is not built up from any M-superqueen solution, namely the knight-walk, there are fewer M-superqueens solutions than N-superqueens solutions. This induces a partial order on the number of N-superqueens

solutions.

Corollary 1: If there is a solution to the N-superqueens problem, and M is a proper divisor of N, there are fewer M-superqueens solutions than N-superqueens solutions.

5. The General Solution to the Queens Problem

This section develops a general solution scheme to the N-queens problem by showing how toroidal decomposition can be modified to solve the planar boards for which no solution has been exhibited. The key to deriving this modification lies in recognizing the major difference between the knight-walk and decomposition. The knight-walk uses individual superqueens as basic blocks. The interplay between these blocks severely limits the ways in which the solution can be modified. Thus, the knight-walk is of limited use in the construction of a general solution. Decomposition, on the other hand, relies on pre-solved boards as blocks. The size of these blocks offers a great deal of flexibility in terms of modifications to the solution, allowing decomposition to serve as the infrastructure of a general solution.

Consider a specific type of decomposition, a D-solution, in which the A-superqueens solution contains a superqueen in the upper left hand corner and the B-knight-walk is used. The scheme for constructing a general N-queens solution consists of appropriately modifying a D-solution. This section can be divided into two parts. The first part shows that replacing the top leftmost tile of a D-solution with a smaller tile does not violate any of the problem's constraints. The second part proves that such modifications provide decomposable N-queens solutions for nearly all remaining cases.

Lemma 1: If there is a solution to N-superqueens and N=AB, there is a solution to the B(A-1)-queens problem.

Proof:

Consider a D-solution. Removal of the top row and leftmost column from the A-superqueens board corresponds to removing the top B rows and B leftmost columns from the N-superqueens board. Just as a solution exists for (A-1)-queens, one exists for (N-B)=B(A-1)-queens. QED.

This modification alone is not enough. In order to proceed with the discussion, however, the following definition is needed:

Definition: ***For all $P \leq N$, call the bottom P rows of the P rightmost columns of an N-by-N board the lower right sub-board of order P.***

The notion of a lower right sub-board is important in showing that replacing the top leftmost tile of a D-solution with a smaller tile does not violate any of the problem's constraints. Consider the example of figure 2, which shows a 7-by-7 board and its lower right sub-board of order 5. The numbers drawn represent the 5 and 7 knight-walks, respectively. Note that the first three superqueens placed on the 5-by-5 sub-board fall in members of OLR(7). The remainder of the 5-superqueens fall in squares containing 7-superqueens. Ergo, OLR(5) is a subset of OLR(7).

Lemma 2: Given the knight-walk solutions to the B-queens and C-queens problems (B-knight-walk and C-knight-walk, respectively), $C \leq B$, replacing B's lower right sub-board of order C with the C-queens solution does not add new diagonals to OLR(B).

Proof:

An N-knight-walk solution wraps around the board exactly once, between rows (N-1)/2 and (N+1)/2. First consider the queens in rows 0 to (C-1)/2 of C's coordinates (rows (B-C) to ((C-1)/2+(B-C)) of B's coords). Replacing B's lower right sub-board of order C with the C-knight-walk puts the C-knight-walk's upper left hand corner on B's main diagonal, in square <(B-C),(B-C)>, thereby aligning the top leftmost queens of the two solutions. Following the knight-walks from these queens, the i^{th} queen, (i=0 to (C-1)/2), is placed on the diagonal characterized by column-row=i (using either coordinate system). This correspondence insures that each of C's queens placed before the wrap is placed in a member of OLR(B). Next, look at the remaining queens. The first of the C-knight-walk's queens placed after the wrap is in square <(C+1)/2,1>, which, in B's coordinates is <((C+1)/2+(B-C)),1+(B-C)>. Similarly, B's wrap lands in square <(B+1)/2,1>. The pointwise difference between these coordinates is ((B-C)/2,(B-C)), separating the squares by (B-C)/2 knight steps. The size of this separation guarantees that one of

B's queens will fall on the square occupied by C's queen. From then on, the two knight-walks will follow the same path. Thus, no diagonals are added to OLR(B). QED.

Lemma 3: If there are solutions to N-superqueens and C-superqueens, N=AB, and $C \leq B$, then there is a solution to the B(A-1)+C-queens problem.

Proof:

Once again, consider a D-solution. Rather than removing the top B rows and leftmost B columns (as was done in Lemma 1), remove only the top (B-C) rows and leftmost (B-C) columns, and replace the remainder of the top left corner tile with the C-knight-walk. By Lemma 2, no diagonal constraints are violated, leaving a solution to the B(A-1)+C queens problem. This is a ***general solution***. QED.

Lemma 3 enlarges the class of board sizes solved by the methods discussed here to all N=B(A-1)+C for some A,B,C not divisible by 2 or 3, and $C \leq B$. Clearly, knight-walk and decomposition are subsumed by this scheme. For a decomposition set C=B, and for a knight-walk set C=B=1.

Definition: *Let N=(A-1)B+C, where there are solutions to the A, B, and C-superqueens problems, and $C \leq B$. A general solution starts with a D-solution to the AB-superqueens problem. The top B rows and B leftmost columns of the decomposition are replaced with C rows and C columns. The C-knight-walk is then placed in the newly created C-by-C tile in the upper left hand corner of the board.*

The question remains, for which of the remaining board sizes does this general solution work? The following two lemmas provide the answer: (almost) all N. In order to find a general solution there must first be a D-solution that can be modified appropriately. The need for a D-solution, in turn, points to the importance of determining the conditions under which A and B exist such that $B(A-1) < N < BA$, and then finding an appropriate A and B.

Lemma 4: Let N be an odd number divisible by 3. If there exist odd numbers A and B, $A=2 \bmod 3$ and $B \neq 0 \bmod 3$, such that $B(A-1) < N < BA$,

then there exists a solution to the N-queens problem.

Proof:

Let $N=B(A-1)+C$. Then:

- $B(A-1)<N<BA$ implies $0<C<B$.
- $A=B=1 \bmod 2$ and $N=1 \bmod 2$ imply $B(A-1)=0 \bmod 2$, and thus $C=1 \bmod 2$.
- $A=2 \bmod 3$ and $B \neq 0 \bmod 3$ imply $B(A-1) \neq 0 \bmod 3$.
- $B(A-1) \neq 0 \bmod 3$ and $N=0 \bmod 3$ imply $C \neq 0 \bmod 3$.

In other words, if an appropriate D-solution can be found, a C that meets the requirements of lemma 3 can be found as well: odd, not divisible by 3, and no larger than B. QED.

Lemma 5: For odd N divisible by 3, $N \neq 3,9,15,27,39$, setting $A=5$ guarantees the existence of a B that meets the specifications of lemma 4.

Proof:

The specifications of lemma 4 are that B be an odd number not divisible by 3, or a member of one of two equivalence classes, (i) $B=1 \bmod 6$ and (ii) $B=5 \bmod 6$. Let N be an odd number divisible by 3. Let B_1 be the largest B such that $4B_1 < N$. Since B_1 is well defined only for $N > 3$, the case of $N=3$ is ruled out of consideration. (In fact, there is no solution to the 3-queens problem). If $N < 5B_1$ then B_1 is as required by lemma 4 (set $C=N-4B_1$, and A,B, and C are all defined as explained above), so assume $N > 5B_1$. Let B_2 be the smallest B larger than B_1, or the smallest B such that $4B_2 > N$. If B_1 is in equivalence class (i), $B_2=B_1+4$. Otherwise, B_1 is in equivalence class (ii), and $B_2=B_1+2$. In the first case, $5B_1 < N < 4B_2=4(B_1+4)$, which implies $B_1 \leq 14$. The only numbers which satisfy these conditions for B_1 are 1, 7, and 13. In the second case, $5B_1 < N < 4B_2=4(B_1+2)$, which implies $B_1 \leq 6$. The only number which satisfy these conditions for B_1 is 5. The intervals of candidates for N, then, are $5B_1 < N < 4B_2$, where B_1 is one of the aforementioned four numbers. In other words, either $5 < N < 20$ or $25 < N < 28$ or $35 < N < 44$ or $65 < N < 68$. The only odd numbers divisible by 3 in these intervals are 9, 15, 27, and 39. QED.

Although a family of solutions is given by any triple (A-1,B,C) meeting the specifications of lemma 3, assigning A a value other than 5 does not solve any of the remaining cases. Since lemma 4 required that A=2 mod 3, the next value that A could be assigned is 11. The minimal non-trivial value for B, however, is 5, (recall B=1 yields the knight-walk), so the smallest board size that A=11 could solve is 50. This is larger than the largest unsolved board. As for the hitherto unsolved even numbers, the construction described in the general solution guarantees the existence of a queen in the upper left hand corner. Removing the top row and leftmost column gives solutions to all even numbers but 2,8,14,26 and 38. As far as these few cases are concerned, the N-queens problem is unsolvable if N=2 or 3, and solutions have been provided for the rest (see figure 3). Lemmas 1 through 5 can be summarized as:

Theorem 2: The general solution scheme solves the N-queens problem for all N, $N \neq 2,3,8,9,14,15,26,27,38,39$.

The following algorithm outlines the general solution scheme (for a detailed program, see appendix 2):

```
Program Nqueens;
  Begin
  Input(N);          {The board size}
  Case N of:
      2,3 : Output (No Solution);
      8,9,14,15,26,27,38,39 : Output (Special-solution(N));
                                        {Table look up}

      Otherwise:  {The general solution scheme is applicable}
       Begin
         Even <- false;  {Solve for an odd number. If N is even,
                          add one, solve for the resulting board,
                          and drop the upper left hand corner}
         If (N is even) then Begin
                               N <- N+1;
                               Even <- true;
                             End;
         If (N mod 3 ≠  0) then Board <-Regular-solution(N)
                           Or Board <-Decomposition-solution(N)
         Else Begin
                Find B,C such that 4B+C=N;
                                   {Set A=5, find B and C}
                Board <- general-solution(N,5,B,C);
              End;
         If (Even) Then Board <- Board-minus-corner-queen;
            {Remove the upper left corner that was added above}
         Output(Board);
       End;
  End.
```

6. Complexity Analysis

The N-queens problem has been defined as: given the board size, N, return a description of an N-by-N board containing N mutually non-attacking queens. Algorithm NQUEENS (shown above) clearly does just that; a single parameter, N, is received as input, and, in only a constant number of operations, the appropriate method for setting up the board is chosen. Two RAM models [1] (chapter 2) are used here to analyze the algorithm's complexity. The first, which assigns unit cost to arithmetic and read/write operations is realistic when N can be represented in one computer register. The second considers bit read/write and logical bit operations as units, and is realistic when N is very large compared to the register's length.

The first output representation which bears consideration is the one mentioned in section 2, the explicit permutation P of 0,...,N-1. Since the number of arithmetic operations is constant, under the specifications of the first model the output cost dominates, making NQUEENS an O(N) algorithm. In the bit operations model the input costs n, where $n=\log_2 N$. The arithmetic calculations cost n as well, since all operations are performed between a number with no more than n bits and a

constant. The dominant factor remains the writing of the output, which in this case is of length O(Nn) bits. Although this constitutes a trivial lower bound for the explicit permutation, other output representations yield better results. For example, if the board is described by an explanation of the general solution's setup, and a list of the parameters N,A,B, and C, the bit operation model could output the result in O(n) time, yielding a linear time algorithm. No possible board description could improve on this, because it takes $\Omega(n)$ bits just to specify the board size for infinitely many N's.

The one possible remaining fly in the ointment is that the decomposition solution relies on number factorization. The best known factorization algorithm takes subexponential time in the bit operation model for certain input parameters [13]. It is important to note, however, that there is no possible input which *must* be fully factored into its primes; any regular solution is applicable wherever decomposition is, and when the general solution is called, finding B and C given that A=5 is done by dividing and adding, not factoring.

The solution family introduced in this paper also reveals a lower bound on the number of solutions for a given N. As discussed in corollary 1, each ordered split of N corresponds to at least one decomposition solution. Consider the sequence $N=\{5^i\}_{i \geq 1}$. The length of the input, 5^i, is $n=(i \log_2 5)$. There are 2^{i-1} different ordered splits of 5^i, corresponding to the number of ways to distribute i indistinguishable objects (the 5's) into j distinguishable cells (for j=1 to i), with at least one object in each cell. These ordered splits indicate that there are at least 2^{i-1} different solutions. In other words, this shows a lower bound of $O(N^{\log_5 2})$, which is subexponetial in n (in the bit operation model). Further exploitation of properties of integer sequences and techniques of combinatorial enumeration may be helpful in finding a better lower bound for this problem.

7. Conclusions

This paper investigated and presented new solutions to two related problems, the N-queens and the N-superqueens. Both problems are examples of constraint satisfaction, and both have been previously solved. Nevertheless, the exact nature of the relationship between them has never been quite understood, and the belief that N-queens is in fact an intractable problem, best solved by backtracking, persists. This paper provides possible resolutions to each of these difficulties. A

family of non-regular solutions to the N-superqueens problem is shown. These superqueen-decomposition solutions are then combined to form queens solutions. This represents a good example of lifting a result found for a special case (toroidal board) to a general case (planar board), and raises the question of whether other well understood algebraic or combinatorial problems can be used to solve their less well understood relatives.

The approach presented herein is somewhat of an oddity: a divide-and-conquer algorithm for a constraint satisfaction problem. Conventional wisdom (and perhaps even intuition) says that a problem with globally interrelated constraints must be considered at the global level. Considering such a problem at a local level runs the risk of violating long distance constraints. To counter this difficulty, local heuristics are frequently applied. In fact, several heuristically modified backtracking approaches to the N-queens problem have been tried [9]. Nevertheless, most algorithms for constraint satisfaction problems are worst case exponential. However, hope springs eternal in the heart of man, and simply exhibiting a divide-and-conquer solution to the N-queens problem indicates that there may be comparable solutions to other problems. Independent of the applicability of the methodology used, this paper took an unusual approach to a classic hard problem and succeeded in finding a new constructive solution.

8. Acknowledgements

We would like to thank all the people who've expressed interest in this paper, and engaged us in thought provoking converstations about the problem. Unfortunately, there are so many people who fall into this category that it would be difficult to list them all. We would, however, like to single out Ashok Chandra and Martin Gardner for providing references that helped put our work in its proper context, Pneena Sageev and Marylin Weitzer for helpful comments about the paper's style, and our advisors, Zvi Galil and Richard Korf for being helpful and supportive throughout.

Appendix 1: The General Constraint Satisfaction Problem

The general constraint satisfaction problem has been discussed (under a variety of names) in several places [12] [8]. The following definition is taken from the formal treatment given in [8].

Definition: *The general constraint satisfaction problem consists of a tuple* $\{N,R_1,R_2,...,R_N,P_{12},P_{13},...,P_{1N},...,P_{N-1,N}\}$ *such that:*

a) N *is a positive integer denoting the number of variables* $(x_1,...,x_N)$.

b) R_i *is a finite set of possible values that can be assigned to variable* x_i.

c) P_{ij} *is a relation. It is a subrelation of* R_i X R_j, *and it represents the problem's constraints.* P_{ij} *is symmetric, (that is,* $P_{ij}=P_{ji}$).

A solution to the problem is an assignment $A = (a_1,...,a_N)$, *where* a_i *is the value assigned to variable* x_i, *such that*

a) for all i, $a_i \in R_i$.

b) for all i,j, $1 \leq i < j \leq N$, $(a_i,a_j) \in P_{ij}$.

If no such assignment exists, an indication that there is no solution should be returned.

The decision problem associated with general constraint satisfaction is: "is a given tuple solvable?" The problem is at least as hard as its related decision problem, because the transformation between them is straightforward: run the problem and return 'yes' if a solution is found and 'no' otherwise. At first glance, the definition of P_{ij} as a binary relation might seem to make them both tractable problems, just as restricting the NP-complete satisfiabilty problem to two literals in a clause renders the problem solvable in polynomial time [7]. Theorem 3 shows that this is not the case. The problem as stated is, in fact, intractable.

Theorem 3: The general constraint satisfaction problem is NP-complete.

Proof: An NP-complete problem must be both in NP and NP-hard. General constraint satisfaction and its related decision problem both have simple nondeterministic polynomial time algorithms: guess an assignment and verify that it is a solution. The NP-complete problems are defined as predicates, or language recognition problems. Thus, only decision problems are eligible. To see that the decision problem in question is NP-hard, reduce it to a known NP-complete

problem, k-colorabilty (chromatic number) [1] (chapter 10).

An instance of k-colorability (KCOLOR) consists of a graph G=(V,E) and a number k. The problem is to decide whether the vertices can be colored with k or fewer colors such that if $(v_i,v_j) \in E$, $c(v_i) \neq c(v_j)$, where $c(v_i)$ is the color of v_i. Any instance of KCOLOR can be transformed into a decision constraint satisfaction (CONSAT) problem as follows:

Let the variables $x_1,...,x_N$ correspond to the vertices $v_1,...,v_N$.

Let $R_i=\{1,2,...,N\}$ for all i.

Let $P_{ij}= \{(x_i \neq x_j)$ AND $(x_i+x_j < 2k))\}$ if $(v_i,v_j) \in E$,
otherwise, $P_{ij}= ((x_i+x_j) \leq 2k)$.

Claim 1: If the KCOLOR problem in question answers 'yes', the CONSAT problem described above will answer 'yes' as well.

Proof (Claim 1): If KCOLOR answered yes, there is a legal coloring using colors c(1),...,c(m), for some $m \leq k$. In order to find a solution to CONSAT, generate an arbitrary one-to-one mapping from the colors to the integers 1,..,m. If $c(v_i)$ was mapped to r, set $a_i=r$. Recall that a_i represents the value assigned to x_i.

- $\forall x_i,x_j$: if $(v_i,v_j) \in E$, then when colored $c(v_i) \neq c(v_j)$. Thus, $a_i \neq a_j$.

- $\forall x_i,x_j$: $a_i,a_j \leq m \leq k$. Therefore, $(a_i+a_j \leq 2k)$.

- $\forall x_i,x_j$: $\{(a_i \neq a_j)$ AND $((a_i+a_j) \leq 2k)$ AND $(a_i,a_j \leq k)\}$
  $\Rightarrow$ $(a_i+a_j < 2k)$.

Thus, P_{ij} is true for all (a_i,a_j) assigned by the transformation from KCOLOR. Ergo, the assignment is legal and CONSAT answers 'yes' as well.

Claim 2: If the CONSAT problem described above answers 'yes', the KCOLOR problem in question will answer 'yes' as well.

Proof (Claim 2): Assume that CONSAT answers 'yes'. Then let $A=(a_1,...,a_N)$ be a solution. A uses m different values, $b_1,...,b_m$, where $b_1<b_2<...<b_m$. Generate a new assignment by mapping b_i to i and assigning i to all variables which were assigned b_i. Call the new assignment $A'=(a'_1,...,a'_N)$. A' solves CONSAT because if $a_i \neq a_j$, then $a'_i \neq a'_j$ and, $\forall i$, $a'_i \leq a_i$. Thus, the inequalities comprising P_{ij} still hold.

Next, generate a one-to-one mapping from A' to a set of m colors,

$c(1),\ldots,c(m)$. If the value assigned to x_i is $a'_i=r$, color vertex v_i with $c(r)$. If $(v_i,v_j) \in E$, then P_{ij} indicates that $a'_i \neq a'_j$. Therefore, $c(v_i) \neq c(v_j)$. If $m \leq k$, this constitutes a legal k-coloring, and KCOLOR answers 'yes' as well. So assume, to the contrary, that $m > k$. Then there exists a number $a'_i=r$, $r \geq k+1$. There also exists a number, $a'_j=r-1$, $r-1 \geq k$. Thus, $(a'_i+a'_j)=(r+r-1) \geq 2k+1 > 2k$. This indicates that in the original assignment, $a_i+a_j \geq a'_i+a'_j > 2k$ and P_{ij} didn't hold. Thus, in contradiction to the assumption, A was not a solution. Ergo, $m \leq k$.

Since the transformation of KCOLOR to CONSAT is in polynomial time, the general constraint satisfaction problem is NP-complete. QED.

Appendix 2: Program NQUEENS

The following PASCAL program implements the general solution algorithm. It solves the problem for odd N by using the knight-walk when applicable and the general solution with A=5 otherwise. For even N's, the (N+1)-queens problem is solved first, and the lower right sub-board of order N is returned.

```
PROGRAM Nqueens(outfile);

CONST
    max = 100;

TYPE
    boardtype = ARRAY[1..max] OF integer;


VAR
    N,A,B,C,number : integer;
    board : boardtype;
    even : boolean;
    outfile : text;

{This procedure sets up a board in a knight-walk configuration.}
PROCEDURE Kwalk (N : integer; VAR board : boardtype);

    VAR i : integer;

    BEGIN
    board[1] := 1;
    FOR i:= 2 TO n DO BEGIN
        board[i] := (board[i-1] + 2) MOD n;
        IF board[i] = 0 THEN board[i] := n;
        END;
    END;

{This procedure sets A to 5, and finds B and C.}
PROCEDURE Breakdown(N : integer; VAR A,B,C : integer);

    VAR M : integer;

    BEGIN
    A := 5;
    M := N;
    WHILE (M MOD 4 <> 0) DO
        M := M-1;
    C := N-M;
    IF (C MOD 3 = 0) THEN BEGIN
        M := M-4;
        C := C+ 4;
        END;
    B := M DIV 4;
    END;

{This procedure sets up the board as necessary.}
PROCEDURE Setup (N,A,B,C : integer; VAR board : boardtype);

    VAR i,j,bigboard : integer;
```

```
    BEGIN
    IF (C = 1) THEN board[1] := 1
    ELSE Kwalk(C,board);

    {Set up a CxC kwalk in the upper left hand corner tile}
    bigboard := 1;  {A record of where we stand in AxA grid.}

    FOR i := (C+1) TO N DO BEGIN
        j := i-C+B; {Where we'd be in a standard decomposition.}

        IF (j MOD B = 1) THEN BEGIN  {Move to next board}
            bigboard := (bigboard + 2) MOD A;

            IF bigboard =0 THEN bigboard := 5;
            board[i] := B * (bigboard - 2) + 1 + C;

              {Upper left hand corner square
             of the appropriate board.}
            END
        ELSE BEGIN
            board[i] := (((board[i-1] + 2)-C) MOD B) +
                                          (B * (bigboard-2)) + c;

            IF (board[i] = ((B * (bigboard-2)) + C))
               THEN board[i] := board[i-1] + 2;
            END;
        END;
    END;

{This procedure shifts the solution to an even size board.}

PROCEDURE Shift (VAR N: integer;VAR board:boardtype);
    VAR i : integer;

    BEGIN
    N := N-1;
    FOR i:= 1 TO N DO
        board[i] := board[i+1]-1;
    END;

{This procedure prints the board}

PROCEDURE Output(board : boardtype);
    VAR i: integer;

    BEGIN
    FOR I := 1 TO N DO
        Write(outfile,board[i] : 4);
    Writeln(outfile);
    END;

BEGIN {Main Program}
Rewrite(outfile);
FOR number := 1 TO max DO BEGIN
    N := number;
    Write(outfile,N:4,' : ');
    IF (N MOD 2 = 0) THEN BEGIN
        N := N+1;  {Make sure that N is odd.}
        even := true
        END
    ELSE even := false;
    IF ((N=3) OR (N=9) OR (N=15) OR (N=27) OR (N=39)) THEN
        Writeln(outfile,'No solution of the designated form.
```

```
                        Look at a table file.')

    ELSE BEGIN
        IF (N MOD 3 <> 0) THEN Kwalk(N,board)
        ELSE BEGIN
            Breakdown(N,A,B,C);
            Setup(N,A,B,C,board);
            END;
        IF even THEN Shift(N,board);
        Output(board);
        END;
    END;
END.
```

Appendix 3: An Older Solution

Yaglom and Yaglom described a solution to the N-queens problem for all N in [21]. They concentrated on solving even board sizes without placing queens in the main diagonal. That vacancy allowed the addition of a row, column, and queen to solve odd sized boards. For even N of the form N=6m or N=6m+4, the setup they describe is shown in figure 4a. It is basically the knight-walk of this paper (more accurately, the knight-walk minus the queen in the upper left hand corner). For board sizes of the form N=6m+2, however, a totally different setup is needed. They exhibited a pattern which works for these boards. Proceeding rightward from the leftmost column, placing successive queens in the diagonals specified by the following pattern (using the diagonal numbering scheme shown in figure 4b), solves the problem.

2n-4,n+1,n+2,n+3,...,3n/2-3,n/2+2,3n/2-1,n/2+1,3n/2-2,n/2+3,n/2+4,n/2+5,...,n-1,4

The example of N=14 is shown in figure 4b.

References

1. Aho, A.V., Hopcroft, J.E., and Ullman, J.D.. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, Massachusetts, 1974.

2. Ball, W.W.R. *Mathematical Recreations and Essays*. Macmillan, New York, 1973.

3. Chandra, A. K. "Independent Permutations, as Related to a Problem of Moser and a Theorem of Polya." *Journal of Combinatorial Theory 16*, 1 (January 1974), 111-120.

4. Dijkstra, E.W. Notes on Structured Programming. In *Structured Programming*, Dahl, O.J., Dijkstra, E.W., Hoare, C.A.R., Ed.,Academic Press, New York, 1972, pp. 1-82.

5. Floyd, R.W. "Nondeterministic Algorithms." *JACM 14*, 14 (1967), 636-644.

6. Gardner, M.. *The Unexpected Hanging and Other Mathematical Diversions*. Simon and Schuster, New York, 1969.

7. Garey, M.R. and Johnson, D.S.. *Computers and Intractability*. W.H. Freeman, New York, 1979.

8. Gaschnig, J. *Performance Measurement and Analysis of Certain Search Algorithms*. Ph.D. Th., Carnegie-Mellon University, May 1979.

9. Haralick, Robert M. and Elliot, Gordon L. "Increasing Tree Search Efficiency for Constraint Satisfaction Problems." *Artificial Intelligence 14*, 3 (October 1980), 263-313.

10. Horowitz, E. and Sahni, S.. *Fundamentals of Computer Algorithms*. Computer Science Press, Rockville, Md., 1978.

11. Hwang, F.K. and Lih, Ko-Wei. "Latin Squares and Superqueens." *Journal of Combinatorial Theory, Series A 34*, 1 (January 1983), 110-114.

12. Knuth, D.E. "Estimating the Efficiency of Backtrack Programs." *Mathematics of Computation 29*, 129 (January 1975), 121-136.

13. Knuth, D.E. *The Art of Computer Programming*. Volume 2: *Seminumerical Algorithms*. Addison-Wesley, Reading, Massachusetts, 1981.

14. Kraitchik, M.. *Mathematical Recreation*. W.W. Norton, New York, 1942.

15. Pearl, J.. *Heuristics*. Addison-Wesley, Reading, Massachusetts, 1984.

16. Polya, G. Uber die 'doppelt-periodischen' Losungen des n-Damen-Problems. In *Mathematische Unterhaltungen und Spiele*, Ahrens, W., Ed.,Teubner, Leipzig, 1918, pp. 364-374.

17. Simon, H. A.. *The Sciences of the Artificial.* The MIT Press, Cambridge, Massachusetts, 1969.

18. Robert Wagner and Robert Geist. The Crippled Queen Placement Problem. Duke University, January, 1984.

19. Wirth, N. "Program Development by Stepwise Refinement." *CACM 14*, 4 (April 1971), 221-227.

20. Wirth, N.. *Algorithms + Data Structures = Programs.* Prentice-Hall, Englewood Cliffs, NJ, 1976.

21. Yaglom, A.M and Yaglom I.M.. *Challenging Mathematical Problems With Elementary Solutions.* Holden Day, San Francisco, California, 1964.

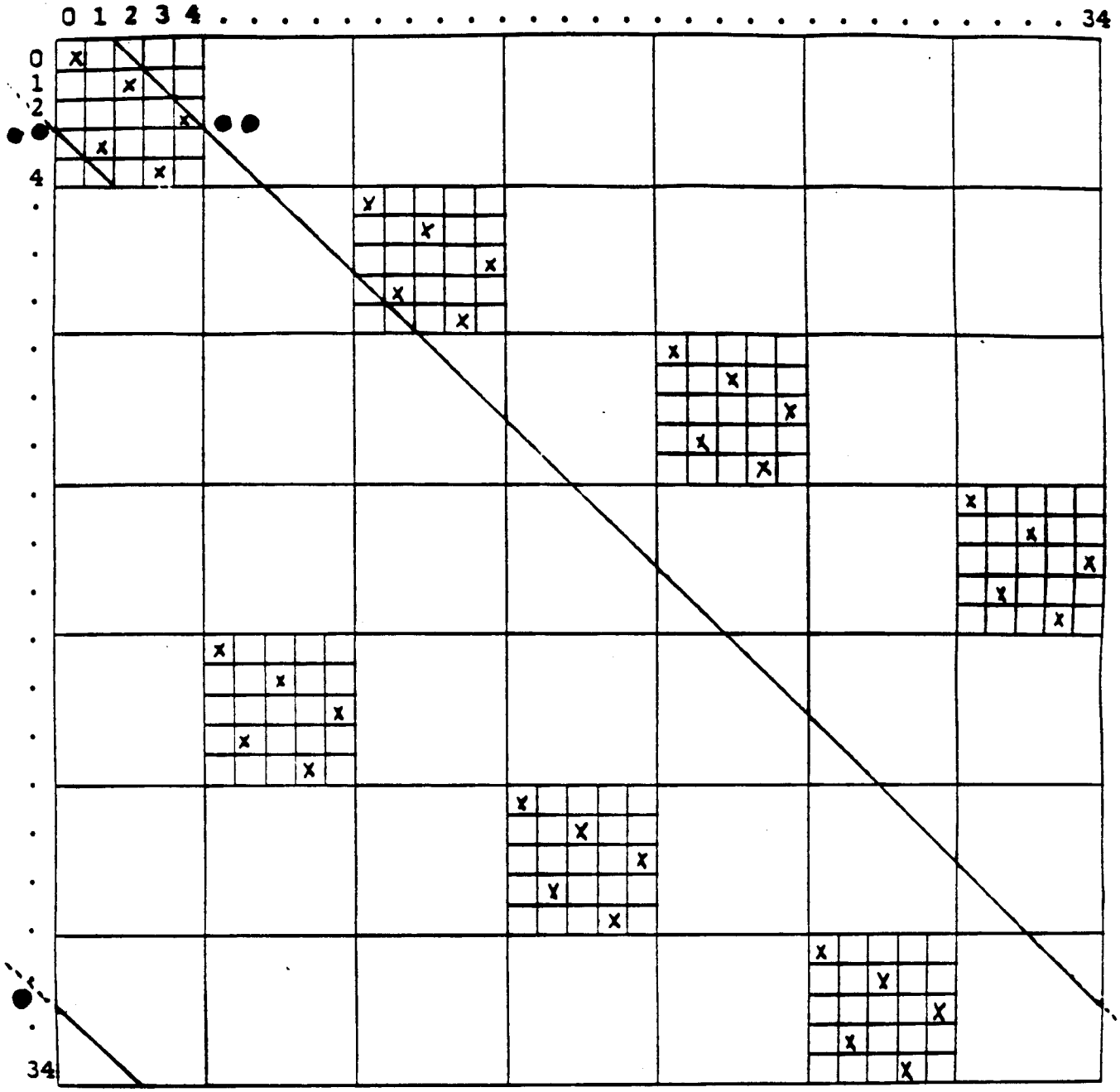

Figure 1: An example of a DS. In this example, N=35, A=7 and B=5. A diagonal is drawn, along with its continuations in both WLR(B) and WLR(N).

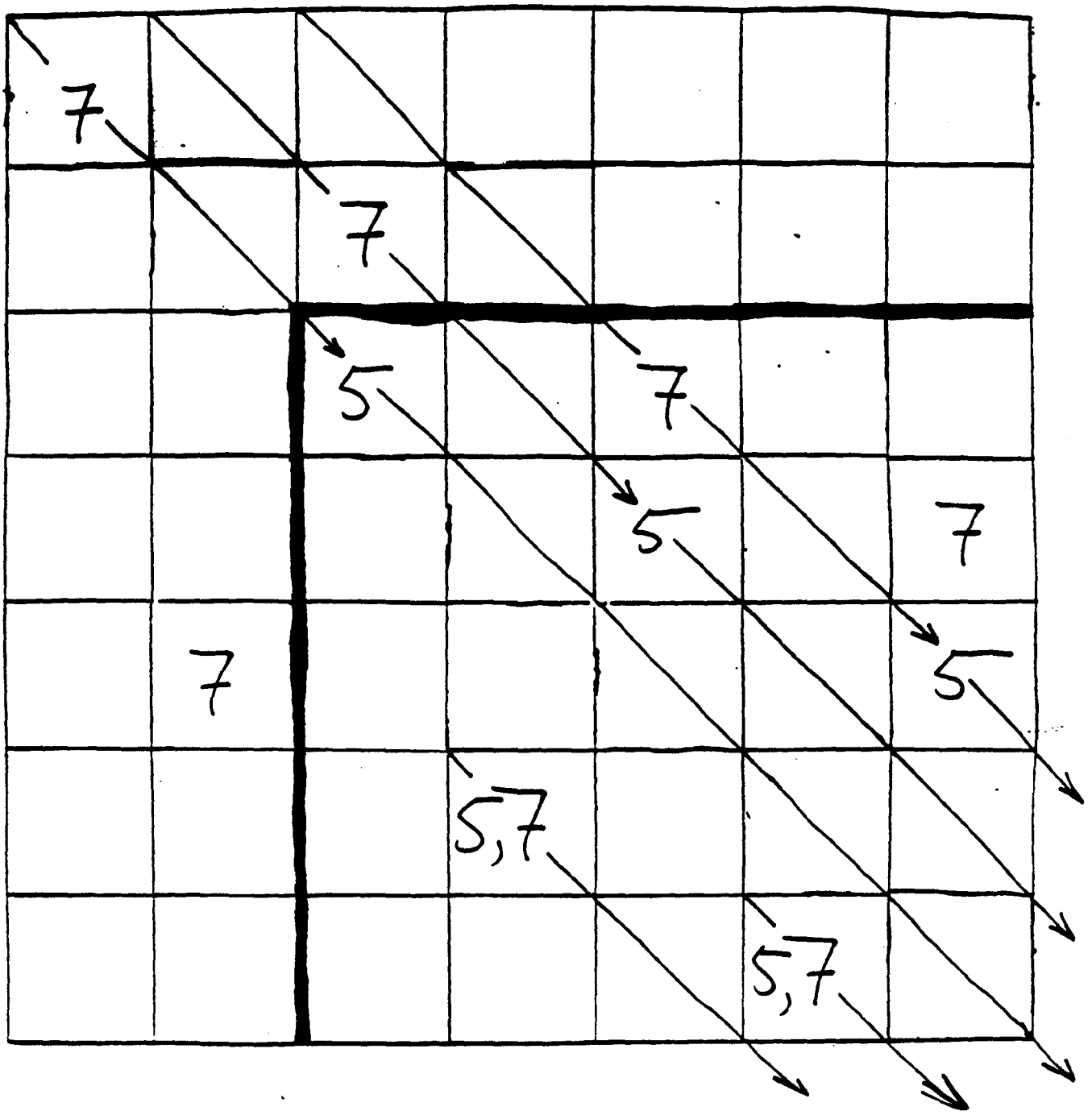

Figure 2: A 7-by-7 board and its lower right subboard of order 5. The knight-walk solutions have been drawn in.

N= 9: P=(0 3 5 7 1 4 2 8 6)

N=15: P=(0 5 14 8 11 4 12 3 6 10 2 13 1 7 9)

N=27: P=(0 15 5 16 9 25 22 19 26 24 14 10 13
23 3 7 4 1 11 21 18 2 17 8 20 12 6)

N=39: P=(0 17 23 29 5 8 21 30 6 20 2 25 34 32 7 27 22 13 35
37 28 12 4 18 33 15 9 31 38 36 1 16 11 19 10 24 3 14 26)

Figure 3:

Solutions to the N-queens problem in special cases. The solutions listed here were found using a standard backtracking procedure with the proviso that a queen must be in the upper left hand corner. This allows the leftmost column and top row to be removed, leaving solutions for N=8,14,26, and 38, respectively.

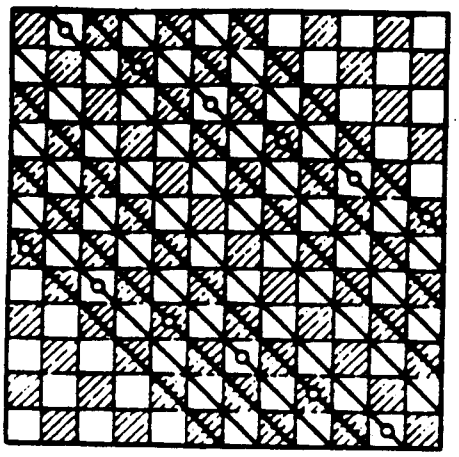

Figure 4a: An example of the solution for N of the form N=6m or N=6m+4. Here, N=12.

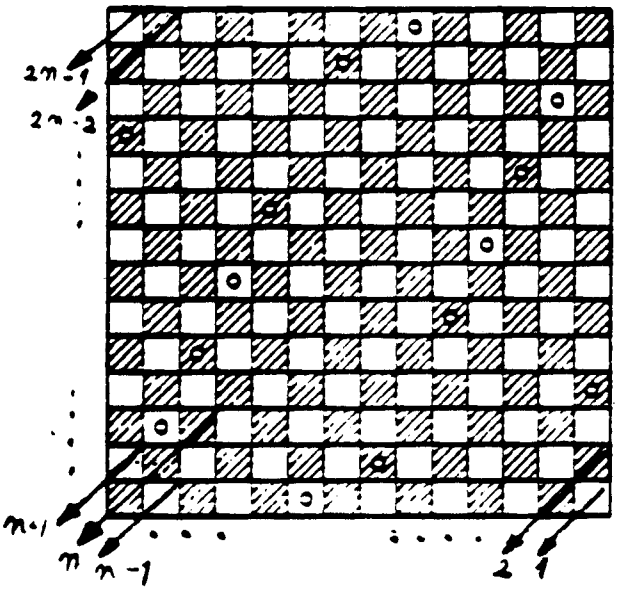

figure 4b: An example of the solution for N of the form N=6m+2. Here, N=14. The diagonal numbers have been drawn in.