ETH Lower Bounds for n-Queens: Time Waits for Nobody

Josh Brunner, Erik Demaine, Timothy Gomez, Markus Hecher, Meryl Zhang

ETH Lower Bounds for n-Queens: Time Waits for Nobody

MIT Hardness Group*, Josh Brunner[1], Erik D. Demaine[1], Timothy Gomez[1], Markus Hecher[2], and Meryl Zhang[3]

[1] MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St., Cambridge, MA 02139, USA, {brunnerj,edemaine,tagomez7}@mit.edu
[2] Univ. Artois, CRIL, CNRS, UMR8188, Lens, France, hecher@cril.fr
[3] Plano Senior High School, Plano, TX, USA, merylz@mit.edu

Abstract. We develop lower bounds with often-matching exponential algorithms for several variants of the n-queens problem. In particular, we prove that placing n queens onto an $n \times n$ board with holes requires $2^{\Theta(n)}$ time for both decision and counting, assuming the Exponential Time Hypothesis (ETH) and #ETH respectively. The same result extends to more general manifolds. If the $n \times n$ board has no holes, but some of the queens are already placed, then completing the placement of n queens is known to be NP-complete and #P-complete; we show that both versions require between $2^{\Omega(\sqrt{n})}$ and $2^{O(n)}$ time, again assuming (#)ETH.

1 Introduction

The classic ***n-queens*** problem asks to place n queens onto an empty $n \times n$ chessboard such that no two queens can capture each other, i.e., no two queens share the same row, column, or diagonal. The $n = 8$ case was first posed in 1848 [2], and was subsequently studied by such luminaries as Gauss and Dijkstra [3]. The *decision version* of the problem is trivial: the answer is YES for all $n > 3$. *Counting* the number of solutions, even for a torus, is "Beyond #P" [8] because the output number requires $\Theta(n \log n)$ bits while the input is merely $\log n$ bits (the binary encoding of n).

To make the n-queens decision problem interesting from a complexity perspective, we can suppose that some k of the n queens are already placed on the board. The ***completion*** problem asks to place the remaining $n - k$ queens to form an n-queens placement. This problem is known to be NP-complete [6]. The known reduction is ***parsimonious*** (preserves the number of solutions), so counting completions is also #P-complete.

While these n-queens problems are famous for being a benchmark for many AI models and computing formalisms [3,6,12,5,10], the *algorithmic complexity*

* Artificial first author to highlight that the other authors (in alphabetical order) worked as an equal group. Please include all authors (including this one) in your bibliography, and refer to the authors as "MIT Hardness Group" (without "et al.").

of the problem remains unresolved. Assuming P $\neq$ NP, n-queens completion requires superpolynomial time, but how superpolynomial?

The most common approach to answering this type of question is to assume the ***Exponential Time Hypothesis (ETH)*** of Impagliazzo and Paturi [9]: solving 3SAT requires $2^{\varepsilon n}$ time for some $\varepsilon > 0$. As essentially all NP-hardness reductions can be traced back from SAT, we only need to track the polynomial growth of the problem size. If a reduction from 3SAT to a problem X increases the size of the instance from n to $O(n^c)$, then X requires $2^{\Omega(n^{1/c})}$ time assuming ETH. In particular, if we can find a reduction that expands the problem size only linearly, we obtain an $2^{\Omega(n)}$ lower bound. There is also a counting complexity analog of ETH called ***#ETH*** [4]: solving #3SAT requires $2^{\varepsilon n}$ time for some $\varepsilon > 0$. Further techniques such as sparsification [9] allow us to assume restrictions on 3SAT such as only a linear number of clauses, or stronger that each variable appears a constant number of times.

So how fast can we solve n-queens completion? Do we obtain ETH-tight bounds? What about counting solutions? Are there other board shapes where even n-queens decision becomes hard?

1.1 Our Results

Table 1 summarizes our results (and related known results) for n-queens. We consider both the decision and counting problems on several geometries, forming a hierarchy of natural generalizations of the $n \times n$ board. Beyond an $n \times n$ board, we consider allowing already placed queens (completion), or ***holes*** — cells where queens cannot be placed but attacks can still go through. More generally, we consider ***manifolds*** made from gluing together a oriented unit squares to form l horizontal/vertical/diagonal lines.

Our lower bounds start in Section 2 by analyzing a chain of reductions to show that ETH and #ETH implies $2^{\Omega(n)}$ lower bounds for many variants of SAT such as 3SAT-3, 1-in-3SAT-3, and a recently introduced variant called Generalized #SAT [7] which allows assignments to be weighted. Next, in Theorem 3.1, we compute the ETH bound given by the chain of reductions in [6] to n-queens, where the last step of the reduction blows up the size by a quadratic factor, so we achieve a weaker bound of $2^{\Omega(\sqrt{n})}$ for both the decision and counting version with preplaced queens. Then, in the rest of Section 3, we improve this result to a singly exponential lower bound of $2^{\Omega(n)}$ when we have cells that are not allowed to have queens ($n \times n$ board with holes). Along the way, we build a reduction to the more general case of a manifold. In Section 4, we prove the same set of lower bounds for the counting versions of these problems.

Our algorithms in Section 5 generalize the $O^*(8^n)$-time[4] dynamic program from [13] to handle boards with holes, establishing that our lower bounds are tight up to constant factors assuming (#)ETH. We use the same technique to get a $O^*(2^l)$ algorithm on a manifold, which is tight according to this parameter.

[4] We use $O^*(f(n))$ to represent $O(f(n) \cdot n^{O(1)})$. Hiding such polynomial factors is reasonable when f is superpolynomial.

Type	Geometry	Lower Bound		Upper Bound	
Decision	$n \times n$ grid completion	NP-complete	[6]	$O^*(8^n)$	[Thm. 5.1]
Decision	$n \times n$ grid completion	$2^{\Omega(\sqrt{n})}$ (ETH)	[Thm. 3.1]	$O^*(8^n)$	[Thm. 5.1]
Decision	$n \times n$ grid with holes	$2^{\Omega(n)}$ (ETH)	[Thm. 3.3]	$O^*(8^n)$	[Thm. 5.1]
Decision	l-line manifold	$2^{\Omega(l)}$ (ETH)	[Cor. 3.1]	$O^*(2^l)$	[Thm. 5.2]
Decision	a-area manifold	$2^{\Omega(a)}$ (ETH)	[Thm. 3.2]	$O^*(2^a)$	[Thm. 5.3]
Counting	$n \times n$ grid	Beyond #P	[8]	$O^*(8^n)$	[13]
Counting	$n \times n$ grid completion	#P-complete	[6]	$O^*(8^n)$	[Thm. 5.1]
Counting	$n \times n$ grid completion	$2^{\Omega(\sqrt{n})}$ (#ETH)	[Thm. 3.1]	$O^*(8^n)$	[Thm. 5.1]
Counting	$n \times n$ grid with holes	$2^{\Omega(n)}$ (#ETH)	[Thm. 4.2]	$O^*(8^n)$	[Thm. 5.1]
Counting	l-line manifold	$2^{\Omega(l)}$ (#ETH)	[Cor. 4.1]	$O^*(2^l)$	[Thm. 5.2]
Counting	a-area manifold	$2^{\Omega(a)}$ (#ETH)	[Thm. 4.1]	$O^*(2^a)$	[Thm. 5.3]

Table 1: Complexity of n-queens placement/completion on the stated geometry. Lower bounds assume ETH, or #ETH for counting. Upper bounds use O^* to omit polynomial factors.

2 Preliminaries

These problems all form a hierarchy. We start with the canonical version of this problem which is on a $n \times n$ chess board which we will call a *grid*. Each of these problems has two versions, *decision* which asks if a solution exists, and *counting* which asks how many solutions exist.

Definition 2.1 (n-Queens Placement).
INSTANCE: $\langle n \rangle$ *where* n *is an integer* ≥ 1.
SOLUTION: *A set* Q *of queens which all fit on board size* n *such that:* $|Q| = n$*; and for any two distinct queens* $\mathbf{q}_1, \mathbf{q}_2 \in Q$*,* $\mathbf{q}_1$ *does not attack* $\mathbf{q}_2$.

Definition 2.2 (n-Queens Completion).
INSTANCE: $\langle n, P \rangle$ *where* n *is an integer and* P *is a set of queens and their locations on board size* n.
SOLUTION: *A set* S_2 *of queens which is a solution to* n*-Queens Placement for* n *and such that* $P \subseteq S_2$.

Various variants were given as intermediate problems in [6] with given disallowed rows, columns, and diagonals which can simulate preplaced queens. Most of these reduce to n-queens completion with only a linear blow up. The next generalization we investigate is a *grid with holes* [5], where we have a $n \times n$ grid with certain squares where we cannot place queens but which do not prevent attacking over.

Definition 2.3 (n-Queens Placement with holes).
INSTANCE: $\langle n, H \rangle$ *where* n *is an integer,* H *is a set of locations where queens cannot be placed.*
SOLUTION: *A set* S_2 *of queens which is a solution to* n*-Queens Placement for* n *and such that for all* $S_2 \bigcap H = \emptyset$.

[5] In [2], this version is referred to as "blocked n-queens".

We refer to rows, columns, and diagonals as *lines*. More generally a line is a set of colocated points. A generalization of all these is n-queens on a *manifold*. The area a of a manifold is the number of cells on the manifold, or the total number of cells. Here each square has up to 8 neighbors each assigned a unique direction from the set north, south, east, west, north-east, north-west, south-east, and south-west. Each direction has an opposite in the obvious way. Here lines are formed by picking a cell and building a set by picking a direction recursively adding the next and previous cell to the set. Each direction is paired with the opposite side. We refer to the total number of lines as l. It is easy to see that grid and torus are a special case of a manifold. For a grid with holes we could not include those cells and connect the adjacent cells. More specifically if there is a hole at position (x, y) we do skip adding a cell for that point and instead setting $(x-1, y)$ and $(x+1, y)$ as east/west neighbors and $(x, y-1)$ and $(x, y+1)$ as north/south neighbors. We do similar for the diagonals. Using these observations we can describe a manifold as a collection of grids with holes which are interconnected in some way.

Definition 2.4 (n-Queens Placement on a manifold).
INSTANCE: $\langle n, M \rangle$ *where* n *is an integer* ≥ 1 *and a graph* M *which is a manifold.*
SOLUTION: *A set* Q *of queens which all fit on the manifold such that:* $|Q| = n$*; and for any two distinct queens* $\mathbf{q}_1, \mathbf{q}_2 \in Q$*,* $\mathbf{q}_1$ *is not on the same line as* q_2*.*

2.1 Assumptions and Satisfiability

The canonical problem we will use for showing hardness is Satisfiability (SAT). This decision problem asks given a v-variable CNF formula does there exists an assignment to the variables which satisfies all the clauses. kSAT restricts each class to have size k. The corresponding counting problem #3SAT asks to count the number of satisfying assignments. 3SAT is NP-complete and #3SAT is #P-complete, so a polynomial-time algorithm is likely to not exist. However in order to get stronger lower bounds ruling out subexponential time algorithms we use assumptions such as the Exponential Time Hypothesis.

Conjecture 2.1 (Exponential Time Hypothesis (ETH) [9]). There exists a constant ε such that 3SAT with v variables requires $2^{\varepsilon v}$ time.

For counting we refer to the analogous assumption as #ETH, which is implied by ETH. Let $\#(x)$ for an instance x of a problem X denote the number of its solutions. A ***c-monious reduction*** f is a function from problem X to problem Y such that for any instance x of X, $\#(x) = c \cdot \#(f(x))$. We only use polynomial-time computable c-monious functions; we say f is ***parsimonious*** if $c = 1$.[6]

Conjecture 2.2 (#ETH [4]). There exists a constant ε such that #3SAT with v variables requires $2^{\varepsilon v}$ time.

[6] Such a strong reduction is not required as obtaining an #ETH lower bound can be done with only a c-monious subexponential time reduction.

Let 3SAT-c be the version of 3SAT where each variable appears only in up to c clauses, note that when c is constant all instances are sparse. For many reductions we require 3SAT-3 which has been shown to be NP-hard [15]. It is not immediately clear that this version has the same lower bound assuming ETH.

Lemma 2.1. *Assuming ETH, there exists a constant ε_1 such that 3SAT-3 with v variables requires $2^{\varepsilon_1 v}$ time. Assuming #ETH, there exists a constant ε_2 such that #3SAT-3 with v variables requires $2^{\varepsilon_2 v}$ time.*

Proof. To start we use "sparsification" [9] which reduces any SAT instance to $2^{\varepsilon_1 v}$ many ***sparse*** instances with $m = O(v)$ clauses. Surprisingly this reduction also results in formulas where each variable only appears in poly($\frac{1}{\varepsilon_1}$) clauses. The reduction in [15] from 3SAT-c to 3SAT-3 (shown in Figure 1) introduces cv new variables. Because ε_1 is a constant, this reduction preserves linear size.

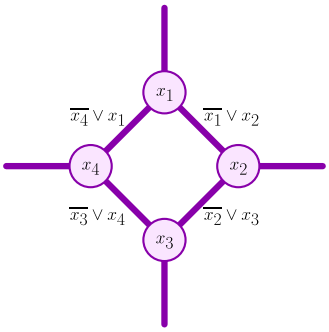


Fig. 1: Converts a degree-4 variable x into four degree-3 variables x_1, x_2, x_3, x_4 connected by clauses of size 2.

The sparsification lemma for counting in [4, Thm 1.1, A.1] has $O(v)$ clauses which for kSAT has $f(k, \frac{1}{\varepsilon_2})$ appearances. Thus for #3SAT we get a constant #3SAT-c. The reduction in [15] turns a single variable into a chain of variables (Figure 1) that are all forced to be equal, so the reduction is parsimonious. □

A variant of 3SAT we use is 1-in-3SAT. Here we still have CNF clauses but each clause must have exactly 1 literal true. There exists a reduction that is parsimonious, preserves #ETH, and preserves constant variable appearance [1].

Lemma 2.2. *Assuming ETH there exists a constant ε_1 such that 1-in-3SAT-3 with v variables requires $2^{\varepsilon_1 v}$ time. Assuming #ETH there exists a constant ε_2 #1-in-3SAT-3 with v variables requires $2^{\varepsilon_2 v}$ time.*

Proof. We know that #1-in-3SAT-3 is NP-hard and #P-hard, even under parsimonious reductions from #3SAT-3 [1, Proposition 3; Appendix B: Figures 21,22,23]. The number of appearances of the original variables remains the same during the reduction, and any new variables are used max 3 times. This reduction linearly preserves the number of variables, so under ETH and #ETH we can not solve either problem in time $2^{o(n)} \cdot n^c$ for some constant c. □

2.2 Generalized #SAT

Generalized #SAT has been defined in [7]. Each version of Generalized #SAT is specified by a set of ***clause types***, non-negative integer functions $f\colon \{0,1\}^k \to \mathbb{Z}_{\geq 0}$, which describes the number of ways a given assignment of k literals satisfies the clause. The two clause types we use in this paper are $\boldsymbol{f(x+y+z)}$ and $\boldsymbol{f(x+y)}$. We allow negations of variables, denoted with a bar (like $\bar{x}$), to be used freely in clauses. An input to Generalized #SAT consists of a number n, the number of variables (denoted $x = (x_1, \ldots, x_n)$), and a set of m clauses $C = \{\phi_1, \ldots, \phi_m\}$. Each ϕ_i is a k-tuple of literals, like $(x_{i_1}, \bar{x}_{i_2}, \ldots, x_{i_k})$. Let x_ϕ denote the restriction of a variable assignment x to the variables in a clause ϕ. The goal is to compute the number of ways to satisfy all clauses:

$$\sum_{(x_1,\ldots,x_n)\in\{0,1\}^n} \prod_{\phi\in C} f(x_\phi). \tag{1}$$

Previous work [7] showed that with just clause type $f(x+y)$ was #P-complete via reduction from #2SAT. Here we prove a $2^{\Omega(n)}$ lower bound when using both $f(x+y)$ and $f(x+y+z)$ assuming #ETH via reduction #1-in-3SAT-3.

Theorem 2.1. *Assuming #ETH, there exists a ε such that Generalized #SAT with clause types $f(x+y+z)$ and $f(x+y)$ with v variables requires $2^{\varepsilon v}$ when each variable appears in up to three clauses.*

Proof. By Lemma 2.2, it suffices to reduce from #1-in-3SAT-3. The goal of this reduction is for a given 1-in-3SAT clause (x, y, z) to create a set of Generalized #SAT clauses, which multiply to 1 when exactly 1 literal is true and 0 otherwise. We must preserve the number of variables and clauses linearly as well.

$$(x+y+z)(\overline{x}+\overline{y})(\overline{y}+\overline{z})(\overline{x}+\overline{z})$$

The first term will be 0 if there are no true variables. The next clauses all are 0 when any two variables are both true. If exactly 1 variable is true then the values of this expression is 2. Each satisfying assignment has a weight of 2^m, thus this is a 2^m-monious reduction. We do not introduce any new variables and increase clauses by a factor of 4. For appearances so far we have increase by a factor of 3. We can use the same degree reduction technique shown in Figure 1. For each variable x which appears 9 times, we add variables $\{x_1 \ldots x_9\}$ and clauses $(\overline{x_i} + x_{(i+1)\%9})$ for $1 \leq i \leq 9$. When all these variables are equal, all the clauses are weight 1. When any two clauses are different some clause will equal 0. Thus this last step is parsimonious. Now our whole chain is 2^m-monious, preserves linear size, and has up to 3 appearances of each variable. □

3 Decision Hardness

First we point out the lower bound given by the reduction in [2] for the grid, given an n variable 3SAT instance this creates a instance of n^2-queens, however

this lower bound is not tight with the best algorithms. Next we show the cases where there does exist an ETH tight lower bound. Even though our definition of manifold was very general, we still attempt to describe the manifold reduction as a collection of $O(1) \times O(1)$ grids, or gadgets, which are then linked together. For the manifold case we show that we create only a linear number of gadgets and thus we get singly exponential lower bounds in Theorem 3.2. We do this to make it easier to describe how to linearly embed all the gadgets into a $n \times n$ grid with holes to get a $2^{\Omega(n)}$ lower bound for a grid with holes.

Theorem 3.1. *n-queens completion on a grid requires $2^{\Omega(\sqrt{n})}$ time assuming ETH. #n-queens completion on a grid requires $2^{\Omega(\sqrt{n})}$ time assuming #ETH.*

Proof. We follow the chain of reductions performed in [6] from 1-in-3SAT-3[7] to n-queens completion. These reductions are between intermediate variants of n-queens which have constraints on (1) Allowed Rows and Columns and (2) Allowed Rows and Columns, Disallowed Diagonals. The reductions from (2) to (1) to the grid case are all linear blow up and involve preplacing queens to simulate constraints. However the reduction, which shows (2) is NP-hard, has super-linear blow up. This is from a variant consisting of M instances of n-queens completion, and a solution must be found for each instance such that none of the queens capture each other along one of the diagonals, but can along all other lines. The reduction, summarized in Figure 2, embeds multiple chessboards spaced apart, thus the parameter board size n increases by a factor of M. Proving this last variant hard reduces from 1-in-3SAT-3 and creates $M = O(n)$ subinstances, a constant number for each variable and clause. The reduction in [6] is in fact parsimonious so it preserves #ETH as well. □

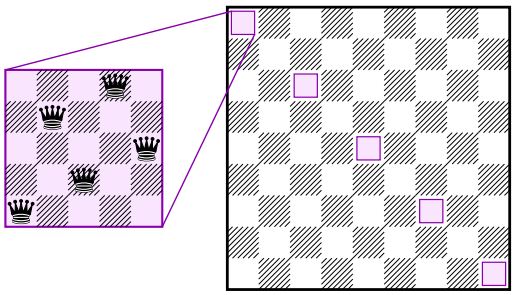


Fig. 2: Summary of the reductions in [6].

The bottleneck here is the intermediate problem which has $O(n)$ subinstances with limited interactions. Breaking it up into these gadgets made arrangement easier as the authors did not have to worry about many of the interactations between them. However in order to reduce this problem back to n-queens completion each subproblem needs its own set of rows, columns, and opposite diagonals.

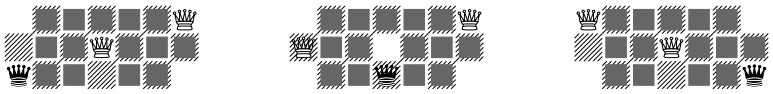
[7] This problem is called "Restricted 1-in-3SAT" in [6].

3.1 Manifold

We use the gadgets almost identical to those used in [6], however by linking them on a manifold we can avoid the blow up given by the previous reduction. Each gadget consists of a $O(1) \times O(1)$ grid, the neighbors internally are given by adjacency in this grid. The variable gadget is a 4×4 grid shown in Figure 3. There exists two ways to satisfy this gadget shown in each figures. The variable gadget has holes filling the second row from the bottom and the second column from the right. The clause gadget is a 3×7 grid with the three possible configurations shown in Figure 4. Each column where a queen is not placed is filled with holes. Each column with a queen represents a literal of the clause. The queen being in the bottom cell of column indicates that a clause is satisfied by that literal.

(a) True variable gadget (b) False variable gadget

Fig. 3: Two possible arrangements of the variable gadget.



(a) (b) (c)

Fig. 4: The three possible arrangements of the clause gadget. The queen in the bottom row forces the truth value of a corresponding variable.

Gadgets are connected by diagonals, each variable gadget has up to three outgoing diagonals each connecting to the bottom cell of a clause gadget. This is shown in Figure 5.

Theorem 3.2. *n-queens completion on a manifold with area a requires $2^{\Omega(a)}$ time assuming ETH.*

Proof. We reduce from 3SAT-3 which has a lower bound of $2^{\Omega(v)}$ from Lemma 2.1 for v variables. The reduction we use $O(1)$ cells for each variable and clause. Because the number of clauses is also $O(v)$, the total number of cells in this reduction $a = O(v)$. Thus we achieve a lower bound of $2^{\Omega(a)}$. Here $n = 3(v + c)$.

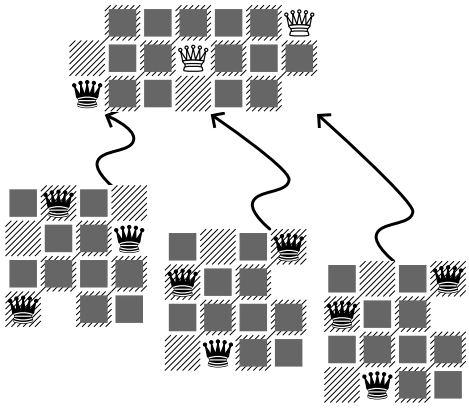


Fig. 5: How to link variable gadgets to clauses. The clause is $(x_1 \vee x_2 \vee \overline{x_3})$ and the variables are assigned 100. The configuration of the clause uses x_1 to satisfy it thus the queen in the bottom row is placed in the first column.

For correctness assume that there exists a satisfying assignment for 3SAT-3, then we may place the queens in the variable gadgets each corresponding to the assignment of that variable. Each clause is satisfied so we may select some literal which satisfies it and place the queen in that column. This cell is legal to place a queen because the configuration of the variable gadget represents the assignment of the variables.

If there exists a valid n-queens configuration then there exists a satisfying assignment. First note that each gadget only has 3 valid columns so they can only contain 3 queens each. The configurations shown in Figure 3 are the only two configurations that are valid thus can represent an assignment to the variables. Each clause gadget needs to place a queen in the bottom row since the gadget only contains 3 rows. The bottom row cells each connect to a variable gadget so the clause must be satisfied by that variable. □

Corollary 3.1. *n-queens completion on a manifold with area l requires $2^{\Omega(l)}$ time assuming ETH.*

Proof. Each cell can only be in up to 4 lines which we use 8 directions. Thus $l \leq O(a)$ and the lower bound holds for l as well. □

3.2 Grid with Holes

Theorem 3.3. *n-queens completion on a $n \times n$ grid with holes requires $2^{\Omega(n)}$ time assuming ETH.*

Proof. We will use the same variable and clause gadgets that are used in the proof of Theorem 3.2 above. The difference is that now, we need to describe an embedding of these gadgets in a $O(n) \times O(n)$ grid such that the only rows,

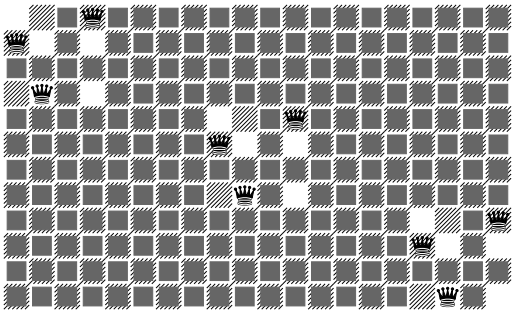


Fig. 6: How to arrange the variable gadgets with holes so that they don't unintentionally interact.

columns and diagonals that are shared between gadgets are the diagonals that variables use to interact with clauses. When placing the gadgets, we refer to two squares of the chessboard as *interacting* if they are on the same row, column, or diagonal (i.e. a queen on one of the squares attacks the other, and thus constrains queens from being on both squares). For the clause gadget, we note that we can stretch the gadget by adding additional rows or columns between the queens as long as the nine non-hole squares where queens can go do not interact with each other along any diagonals. Thus, a clause gadget is really just nine squares, located in 3 (not necessarily adjacent) rows and 3 (not necessarily adjacent) columns, which do not interact with each other on any diagonals.

To lay out the variables, we place each variable along a 2:1 slope line, as shown in figure 6. This ensures that all of the variable gadgets are on distinct rows and columns, and the diagonals they will use to interact with clauses are all parallel diagonals.

To the upper left of the variable line, we will place our clauses. We'll place the rows of the clauses one at a time, greedily picking the lowest row which does not create any undesired interactions. For each clause, we place its bottom row on the lowest row which only interacts with the three variables it contains. On that row, we fill the entire row with holes except for the 3 squares which interact on the diagonals with the three variables present in the clause. Such a placement is *valid* if those 3 squares do not interact with any other clauses. The other two rows will be placed similarly on the lowest rows such that the three squares that are part of that clause do not interact with any other clause or variable.

Now we need to show that this greedy placement of clauses will guarantee that all of the clauses fit within $O(n)$ rows. We use induction on the number of clauses. Suppose we can fit the first n clauses in $O(n)$ rows. When placing the last clause, we place it in the lowest row which has no unintended interactions. For each square from a previous clause, there are up to 6 rows which contain a square that interacts with both that square and one of the three desired variables, preventing the clause from being placed on that row. These 6 possible interactions come from the intersection of the column and upward diagonal of the square in question

intersecting each of the three variable diagonals. Since there are $9n$ squares in previous clauses, there are at most $54n = O(n)$ rows which have an unintended interaction with a previous clause. Thus, the lowest row which has no such interaction must be at a height of $O(n)$, so this clause can be placed (and along with all of the other clauses) can fit within $O(n)$ rows. □

4 Counting Hardness

We now utilize the lower bound proved in Theorem 2.1 for generalized #SAT. First note that the reduction from 3SAT-3 in Theorem 3.2 is not parsimonious. The clause gadget has exactly 3 legal configurations each corresponding to a literal. When a literal is satisfied then that configuration is possible. Thus a clause satisfied by multiple literals can have multiple configurations. The original #P-completeness in [2] overcame this by using a larger clause gadget and reducing from #1-in-3SAT-3. However we handle this "at the SAT" level and reduce from generalized #SAT.

Theorem 4.1. *#n-queens completion on a manifold with area a requires* $2^{\Omega(a)}$ *time assuming #ETH.*

Proof. Consider the reduction from Theorem 3.2, connect clause and variables in the same way. We must also create clauses of size 2 which we may do by create a slightly smaller gadget, dropping the rightmost and topmost column and row. Let $X = (x_1, \ldots, x_n)$ be some assignment to the formula, define $Q(X)$ to to be the number of valid configurations with variable gadgets in states representing the bits of X. We then wish to show that for each X,

$$Q(X) = \prod_{\phi \in C} f(x_\phi)$$

Since we are essentially fixing the variable gadgets to represent X, the only degree of freedom is the clause gadgets. Since no two clause gadgets share a line they are independent thus we multiply the numbers of configurations of each. The number of possible configurations of a clause gadget is the number of true literals, in other words $f(x+y+z)$. Thus the reduction is parsimonious and still preserves linear size. □

The result extends to lines in the same way as Corollary 3.1. Finally changing the problem we're reducing from does not change the embedding size so we get our holes lower bound as well.

Corollary 4.1. *#n-queens completion on a manifold with l lines requires* $2^{\Omega(l)}$ *time assuming #ETH.*

Theorem 4.2. *#n-queens completion on an* $n \times n$ *grid with holes require* $2^{\Omega(n)}$ *time assuming #ETH.*

5 Algorithm

Previous work [13] has given a $O^*(8^n)$ algorithm for n-queens counting on both the grid and the torus. It easily generalizes to when we have a grid with holes, and thus for completion. We first prove this generalization showing that our lower bound is tight up to ETH for grids with holes. We further generalize the algorithm to a manifold and show our lower bounds are tight when parameterized by both l and a.

Theorem 5.1. *n-queens and #n-queens on a $n \times n$ grid even with holes can be solved in $O^*(8^n)$.*

Proof. n-queens counting can be solved using dynamic programming in $O^*(8^n)$ time [13]. The key to this dynamic program over the set of currently blocked lines, or lines which already contain a queen which we will refer to as line configurations. The algorithms stores line configurations and the number of ways to place queens to reach that it. Then they loop through each available position, attempting to place a queen there, then for each placement incrementing the count of line configurations or adding new ones to the queue. When starting with holes we can just skip these in the placement step. For the decision version we just check whether the output is greater than 0. □

Next we point out that the general form of the algorithm in [13] achieves $O^*(2^l)$ run time. We note naively running this version on an $n \times n$ grid gives a worse run time. In total an $n \times n$ grid contains $6n - 2$ lines however by merging partial configurations in [13, Theorem 3.2] the authors are able to decrease this search space to only these semi-exhausted lines when processing in row-major order. However this doesn't immediately extend to more general manifolds to decrease the run time.

Theorem 5.2. *n-queens and #n-queens on a manifold on l-line manifold can be solved in $O^*(2^l)$.*

Proof. As stated in [13] the run time is exponential in the number of lines. The dynamic program represents configurations uses currently blocked lines, thus the number of objects that can be pushed in the queue is $O(2^l)$. On a manifold we can run the same algorithm selecting each cell and placing a queen there and adding the new configurations to the queue. □

Finally we give a singly exponential upper bound even parameterized by area. Here this algorithm would be much worse on a grid as its area is n^2.

Theorem 5.3. *n-queens and #n-queens on a manifold with area a can be solved in $O^*(2^a)$.*

Proof. There are up to 2^a possible configurations as each cell either contains a queen or does not. We can check each one to see if it is a solution. □

6 Conclusion and Open Problems

In this paper we gave ETH bounds for generalized version of the n-queens problem. We first give the $2^{\Omega(\sqrt{n})}$ lower bound proven by previous reductions. We then provide singly exponential lower bounds starting at a grid with holes. We finish by analyzing the best deterministic upper bounds.

We believe that n-queens and its variants are a great candidate to better study the complexity of NP-complete and #P-complete problems. While we make progress toward this we leave a number of open problems.

- Can we get a $2^{\Omega(n)}$ lower bound for n-queens completion, i.e. can we eliminate the quadratic blow up?
- Can we decrease the base to get a $O^*(2^n)$ or smaller algorithm for counting? Can we further get a tighter bound on the number of lines we must process?
- Can we get any $\Omega(2^n)$ lower bounds using the Strong Exponential Time Hypothesis (SETH) on a manifold?
- On an $n \times n$ torus is the problem still hard?
- Are there any manifolds which are easier, perhaps parameterized?

In more detail we wish to close the gap for completion on a $n \times n$ board. While we do not believe a subexponential time algorithm exists, we proved that such an algorithm must take advantage of the board being a perfect grid, not even missing any holes. The botttleneck of our reduction is embedding the manifold into the grid which requires arbiratry placement of holes. If we could instead only block complete rows, columns, and diagonals, we could use the reductions from [6] to get a tight bound. While there exists techniques to handle some of the holes we include[8] the problematic ones are those that allow us to arrange variables and gadgets together as in Figure 6. Currently the fastest algorithms on grids [13] decrease the naive base of 64 to 8 by merging configurations to bound the total number of lines we must process. Are there any other insights we can use to get a smaller bound? What is the smallest the base can be? ETH alone is not powerful enough, however using SETH we can get lower bounds of the form $\Omega(2^n)$ by reducing from kSAT. However in this case we also are still missing tools such as sparsification in this setting. To our knowledge there are no hardness results or lower bounds for the torus, can any of these reductions be extended? Finally we can further analyze these algorithms by studying simple manifolds such as those having bounded treewidth.

Acknowledgements

We found this open problem in thanks to MIT Future Tech's series of algorithmic surveys [14,11]. We would like to thank Jayson Lynch for helpful discussion and guiding early steps of the project and Jenny Diomidova for help with Generalized #SAT. This collaboration originated through the Research Science Institute (RSI), jointly sponsored by the Center for Excellence in Education and MIT. We thank the program for enabling this research partnership.

[8] such as reducing from 1-in-3SAT and having larger clause gadgets as in [6]

References

1. Régis Barbanchon. On unique graph 3-colorability and parsimonious reductions in the plane. *Theoretical Computer Science*, 319(1-3):455–482, 2004.
2. Max Bezzel. Schachfreund. *Berliner Schachzeitung*, 3:363, 1848.
3. Ole-Johan Dahl, Edsger W. Dijkstra, and Charles Antony Richard Hoare. *Structured Programming*, volume 8 of *A.P.I.C. Studies in Data Processing*. Academic Press, 1972.
4. Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlen. Exponential time complexity of the permanent and the Tutte polynomial. *ACM Transactions on Algorithms (TALG)*, 10(4):1–32, 2014.
5. Martin Gebser, Marco Maratea, and Francesco Ricca. The Seventh Answer Set Programming Competition: Design and Results. *Theory and Practice of Logic Programming*, 20(2):176–204, 2020.
6. Ian P. Gent, Christopher Jefferson, and Peter Nightingale. Complexity of n-queens completion. *Journal of Artificial Intelligence Research*, 59:815–848, 2017.
7. MIT Hardness Group, Josh Brunner, Erik D. Demaine, Jenny Diomidova, Timothy Gomez, Markus Hecher, Frederick Stock, and Zixiang Zhou. Easier Ways to Prove Counting Hard: A Dichotomy for Generalized #SAT, Applied to Constraint Graphs. In Julián Mestre and Anthony Wirth, editors, *35th International Symposium on Algorithms and Computation (ISAAC 2024)*, volume 322 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 51:1–51:14, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
8. Jieh Hsiang, D. Frank Hsu, and Yuh-Pyng Shieh. On the hardness of counting problems of complete mappings. *Discrete mathematics*, 277(1-3):87–100, 2004.
9. Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.
10. Donald .E. Knuth. *The Art of Computer Programming, Volume 4B: Combinatorial Algorithms*. Pearson Education, 2022.
11. Emily Liu. *A metastudy of algorithm lower bounds*. PhD thesis, Massachusetts Institute of Technology, 2021.
12. Google OR-Tools. The N-queens problem. Available at https://developers.google.com/optimization/cp/queens (2025/01/28).
13. Igor Rivin and Ramin Zabih. A dynamic programming solution to the n-queens problem. *Information Processing Letters*, 41(5):253–256, 1992.
14. Yash Sherry and Neil C. Thompson. How fast do algorithms improve? *Proceedings of the IEEE*, 109(11):1768–1777, 2021.
15. Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discrete applied mathematics*, 8(1):85–89, 1984.