hexagon:2611.00729v1

Peaceable queens on boards of size 16 to 20

Jason Davies*

November 20, 2026

Abstract

Let $a(n)$ be the largest m such that m white and m black queens can be placed on an $n \times n$ board with no queen attacking a queen of the other colour (OEIS A250000). The refereed literature and the OEIS gave $a(n)$ for $n \leq 15$, and in 2026 Arevalo determined $a(16)$, $a(17)$ and $a(18)$ in an unrefereed, AI-assisted computational note. We prove $a(19) = 52$ and $a(20) = 58$, which were open, and we confirm $a(16) = 37$, $a(17) = 42$ and $a(18) = 47$ independently, with two separately written programs. In each case $a(n) = \lfloor 7n^2/48 \rfloor$, so the constructions that Ainley gave in 1977 are optimal on these boards. The proof is computer-assisted. As in earlier work, we label each of the $6n-2$ lines of the board white or black, which turns the problem into maximising the smaller of two cell counts. Our new ingredient is the bound: for fixed rows and columns we bound that quantity by Lagrangian duality, and each Lagrangian subproblem is a minimum cut on the $4n-2$ diagonals. With symmetry breaking, the search for $n = 20$ meets 5.7×10^{10} row/column labellings up to symmetry, and a single family of minimum cuts prunes every one of them. That search took 221.9 CPU hours, mostly on cloud machines. A second implementation repeats the whole search for every $n \leq 18$. Orbit counts by Burnside's lemma confirm that every search is complete, and exact LP certificates and SAT proofs checked by `drat-trim` give further checks. As side results we find 26 and 5 inequivalent optimal configurations for $n = 16$ and $n = 17$ (OEIS A260680), and we confirm by computation that the 20×20 and 22×22 tori admit 51 and 64 queens of each colour but not more (A279405), values that also follow from Harries's exact formula for even tori.

AI transparency statement. The results, the computer-assisted proofs, the code and this manuscript were produced by artificial intelligence. The human author initiated the broader research and directed the AI systems, but did not write the text and has not checked the arguments. The AI system's first literature search missed the work of Arevalo and of Harries discussed below, and an earlier version of this paper, and our edit of OEIS A250000, presented $a(16)$–$a(18)$ as new. Details are given at the end of the paper.

1 Introduction

Two queens of different colours *attack* each other if they share a row, a column or a diagonal and no queen stands between them. The *peaceable queens* problem asks for $a(n)$, the largest m such that m white and m black queens can be placed on an $n \times n$ board with no white queen attacking a black queen. It is Problem C1 of Ainley's *Mathematical Puzzles* [1]. Bosch posed it again in 1999 and solved small cases by integer programming [3, 4], and it is sequence A250000 of the OEIS [14].

Ainley gave constructions for every $n \leq 30$ and a general construction of asymptotic density $7/48$. Ainley's values agree with $\lfloor 7n^2/48 \rfloor$ for $14 \leq n \leq 30$ except at $n = 27$, where a board with

*jason@jasondavies.com

106 queens of each colour is now known; for every other n with $16 \le n \le 30$ Ainley's values are still the best known [14]. The exact values $a(1), \dots, a(15) = 0, 0, 1, 2, 4, 5, 7, 9, 12, 14, 17, 21, 24, 28, 32$ were obtained with several methods. Smith, Petrie and Gent [20] used constraint programming with symmetry breaking (see also [16]), and Prestwich and Beck [18] obtained $a(12)$ and $a(13)$. In 2014 Pratt confirmed $a(1), \dots, a(13)$ by integer programming and gave upper bounds such as $a(16) \le 64$ and $a(19) \le 90$. Tabatabai found $a(14)$ and $a(15)$ by integer programming in 2018 (all three in [14]). The problem is also an exercise in Knuth's volume on satisfiability [12].

Since $\lfloor 7n^2/48 \rfloor$ matches every known value except $n = 5$ and $n = 9$ (where $a(n)$ is one larger), it has been asked whether Ainley's construction is optimal. Yao and Zeilberger [23] studied the problem numerically and symbolically, and Clinch, Drescher, Huynh and Saffidine [5] record their conjecture that it is. Clinch et al. also proved $a(n) \le 0.1716\, n^2$ for large n, improving the bound $n^2/4$ of van Bommel and MacEachern [21], and they noted that only the first fifteen terms were known. Kane studied the asymptotics [11].

Prior and concurrent work. Describing a placement by labels on lines is not new. Smith, Petrie and Gent [20, §5] used binary line variables (a line has no white queen, or no black queen) and remarked that one could search on the $6n$ line variables alone. Prestwich and Beck [18] wrote an integer program with 0/1 line variables, and the upper bounds of Clinch et al. [5] use the regions cut out by the lines that meet one army. Arevalo [2] states the line-labelling reformulation of Lemma 2.1 below, with a proof (Lemma 1 of [2]). With a search that also takes rows and columns before diagonals, the product bound $m \le |R||C|$, a bound for each diagonal family on its own, and the group $D_4 \times \{\text{colour swap}\}$, Arevalo determined $a(16) = 37$, $a(17) = 42$ and $a(18) = 47$ in August and September 2026. The note is AI-assisted and unrefereed; its runs for $n = 17$ and 18 use a single engine, and its DRUP proofs go up to $n = 7$. It lists $a(19)$ as future work, and it asks, as an open question, for a joint bound over both diagonal families. Our minimum-cut Lagrangian bound (Lemmas 3.1 and 3.2) is such a joint bound; we have not found it elsewhere. On the torus, Harries [7] proved an exact formula $t(2q) = H(2q)$ for even orders, with a Lean proof of the main theorem, using a line-colouring equivalence (Lemma 6 of [7]) and a partially dualized Lagrangian. The formula gives the torus values for $n = 20$ and 22 in Section 7, and Harries also found $t(15) = 20$ and $t(17) = 28$. Thus our new results are $a(19)$ and $a(20)$, the counts A260680(11..17) and the minimum-cut bound; for $a(16)$–$a(18)$ and the torus our computations are independent confirmations. An earlier version of this paper, and our edit of OEIS A250000, presented $a(16)$–$a(18)$ as new, because the AI system's first literature search missed Arevalo's note.

Theorem 1.1. *$a(16) = 37$, $a(17) = 42$, $a(18) = 47$, $a(19) = 52$ and $a(20) = 58$.*

The lower bounds come from known constructions. For $n = 16$ we use the Ainley arrangement shown in the OEIS entry (Figure 1). For $n = 17, \dots, 20$ we use the boards in Kamenetsky's file of best known solutions [14]. We checked each of them directly for attacks. The new content of Theorem 1.1 is the upper bounds for $n = 19$ and 20; for $n = 16, 17, 18$ the upper bounds confirm Arevalo's [2]. The upper bounds say that no placement has 38, 43, 48, 53 or 59 queens of each colour, respectively.

2 Line labellings

The board has n rows, n columns, $2n-1$ diagonals ($i-j$ constant) and $2n-1$ antidiagonals ($i+j$ constant), so $6n-2$ *lines* in all. Every cell lies on exactly four lines, and distinct cells have distinct (diagonal, antidiagonal) pairs. A *labelling* x assigns W or B to each line. A cell is a W-*cell* of x if

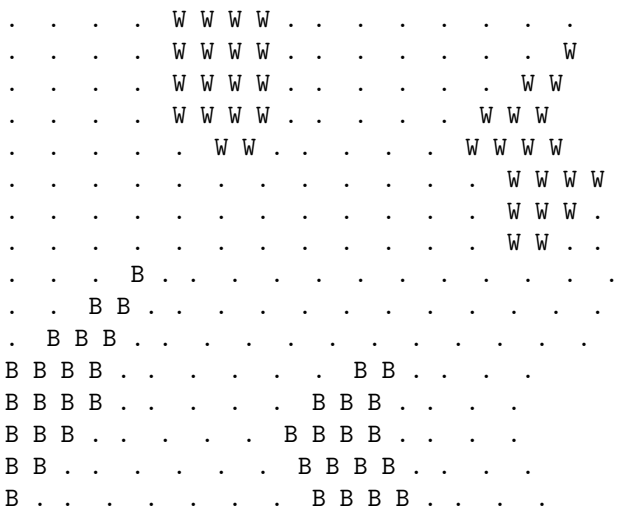


Figure 1: An Ainley-type arrangement of 37 white and 37 black queens on the 16×16 board, as reproduced in [14].

all four of its lines are labelled W, and a B-*cell* if all four are labelled B. Let $W(x)$ and $B(x)$ be the numbers of W-cells and B-cells. No cell is both.

Lemma 2.1. $a(n) = \max_x \min\{W(x), B(x)\}$.

Proof. Take a peaceable placement with m queens of each colour. Suppose some line contains queens of both colours. Walking along it, we meet two consecutive queens of different colours, with no queen between them, and these two attack each other. So every line contains queens of at most one colour. Label a line by the colour of its queens, and label empty lines arbitrarily. Then every white queen stands on a W-cell and every black queen on a B-cell, so $\min\{W, B\} \geq m$. Conversely, given x, put $m = \min\{W, B\}$ white queens on W-cells and m black queens on B-cells. A line containing queens of both colours would carry both labels, which is impossible. So no white queen shares a line with a black queen, and in particular none attacks one. □

This reformulation is known: it is Lemma 1 of Arevalo [2], its torus form is Lemma 6 of Harries [7], and binary line variables go back to [20, 18]. The proof shows that it does not matter whether "attack" allows blocking. So the question is whether some labelling has $W \geq T$ and $B \geq T$, where $T = \lfloor 7n^2/48 \rfloor + 1$.

3 The bound

Let R and C be the sets of rows and columns labelled W. Every W-cell lies in $R \times C$ and every B-cell in $R^c \times C^c$. This gives the *count bound*: if some rows and columns are labelled, then $W \leq r_{\mathsf{W}} c_{\mathsf{W}}$, where r_{W} and c_{W} are the numbers of rows and columns not labelled B. The same bound with the colours exchanged holds for B. Once (R, C) is fixed, call the cells of $R \times C$ W-*candidates* and those of $R^c \times C^c$ B-*candidates*. Write $x_d, y_e \in \{0, 1\}$ for the labels of diagonal d and antidiagonal e, with 1 meaning W. A W-candidate on d and e is a W-cell if and only if $x_d = y_e = 1$. A B-candidate is a B-cell if and only if $x_d = y_e = 0$.

Lemma 3.1 (Lagrangian bound)**.** *For any set X of labellings and any $\lambda \in [0, 1]$,*

$$\max_{x \in X} \min\{W(x), B(x)\} \leq \max_{x \in X} \{\lambda W(x) + (1 - \lambda) B(x)\}.$$

Proof. $\min\{W, B\} \le \lambda W + (1-\lambda)B$ holds for each x. □

Lemma 3.2 (Min-cut evaluation)**.** *Fix (R, C), integers $0 \le p \le Q$ and any partial labelling of the diagonals and antidiagonals. There is a network N_p, on a source s, a sink t and one node for each unlabelled diagonal or antidiagonal, and a constant G, such that each completion x satisfies $pW(x) + (Q-p)B(x) = G - \mathrm{cap}(S_x)$, where S_x is the s–t cut whose source side consists of s and the lines labelled* W *by x. Hence $U_p := \max_x\{pW + (Q-p)B\} = G - \mathrm{mincut}(N_p)$.*

Proof. For 0/1 variables,

$$p\,xy = p - p\,[x=0] - p\,[x=1, y=0],$$
$$(Q-p)(1-x)(1-y) = (Q-p) - (Q-p)\,[x=1] - (Q-p)\,[x=0, y=1].$$

With the source side meaning W, the four bracketed indicators are exactly the costs of the arcs $s \to x$, $x \to y$, $x \to t$ and $y \to x$, each of capacity p or $Q-p$, in the cut S_x. We sum these identities over the W-candidates (with $x = x_d$, $y = y_e$) and the B-candidates. If a line is already labelled, we substitute its value. The term then becomes a constant (its gain, or 0) or a single arc at the other line. Every node has arcs from s and to t, and exactly one of these two is cut. So we may subtract the smaller of the two capacities from both and from G. Distinct completions give distinct cuts, and every cut arises from a completion, so the maximum over completions is G minus the minimum cut. □

This bound couples the two diagonal families, which a bound for each family separately does not (compare [2, §7]); we have not found it in the literature. Equivalently, $pW + (Q-p)B$ is a supermodular pseudo-Boolean function of the free lines in which every term involves at most two variables. Such functions are maximised by minimum cuts [8, 13]. Without the row/column stage every cell would couple four lines, and the same function would be a maximum-weight closure [17]. The program uses $Q = 256$.

Corollary 3.3 (soundness needs only a feasible flow)**.** *If f is the value of any feasible s–t flow in N_p and $G - f < QT$, then no completion has $W \ge T$ and $B \ge T$.*

Proof. Any feasible flow has value at most the capacity of every cut, so for every completion x we have $pW + (Q-p)B = G - \mathrm{cap}(S_x) \le G - f < QT$. By Lemma 3.1 with $\lambda = p/Q$, $\min\{W, B\} < T$. □

So the correctness of the pruning does not depend on the max-flow routine finding an optimal flow or on how p is chosen. These only affect how much is pruned. The program picks p by bisection: U_p is convex in p, and $W(x^*) - B(x^*)$ is a subgradient at the optimal completion x^*. The completion x^* is itself a labelling, and if $\min\{W(x^*), B(x^*)\} \ge T$ the search has found a solution.

4 Symmetry and search

The group $G = D_4 \times \{\text{colour swap}\}$ has order 16 and acts on labellings. A symmetry of the board maps the four lines of each cell to the four lines of the image cell, and the colour swap exchanges W and B. So $\min\{W, B\}$ and the conditions $W, B \ge T$ are G-invariant. The row and column lines are enumerated in groups $\{\text{row } e,\ \text{row } n-1-e,\ \text{column } e,\ \text{column } n-1-e\}$ for $e = 0, 1, \ldots$; when n is odd the last group is $\{\text{row } m,\ \text{column } m\}$ with $m = (n-1)/2$. Each group is mapped onto itself by every element of G.

Lemma 4.1. *Order the row and column lines group by group. It suffices to search the labellings whose row/column part is lexicographically minimal in its G-orbit. Moreover, at the end of every group the search may discard any prefix that is not lexicographically $\leq$ each of its 16 images.*

Proof. Let S be the set of row and column lines. It is G-invariant, so G acts on labellings of S. Each orbit of labellings contains one whose restriction to S is lexicographically minimal among the restrictions of its images, and that labelling has the same value of $\min\{W, B\}$. Let P be a union of initial groups. It is G-invariant, so the restriction of $g \cdot x$ to P depends only on the restriction of x to P. The order lists P first, so lexicographic comparison of whole sequences compares their restrictions to P first. If $x|_S \leq (g \cdot x)|_S$ then $x|_P \leq (g \cdot x)|_P$. So the lex-minimal representative survives every prefix test. □

The test at the end of the last group keeps a labelling of S only if it is the lex-minimum of its orbit, so at most one labelling per orbit survives. Lemma 4.1 says that exactly one does. We also check this numerically. By Burnside's lemma we count the orbits of pairs (R, C) that pass the count bound, and this number must equal the number of leaves of the search (Section 6).

The search has two stages. Stage 1 is a depth-first search over row and column labels with the pruning of Lemma 4.1 and the count bound. At each complete (R, C), Stage 2 evaluates Lemma 3.2 with all diagonals free. If no p prunes, it branches on the diagonals and antidiagonals and recomputes the bound at every node. For $n \geq 16$ the work is split into *cubes*: the canonical prefixes of the first four groups (sixteen lines). There are 4708 of them for each $n = 16, \ldots, 20$. Each cube is solved by a separate process and its result is logged together with its node and flow counts.

5 Computation

Table 1 lists the runs. For $n = 20$ and $T = 59$ the 4708 cubes contain 57,257,809,825 orbits of (R, C) that pass the count bound. The root bound pruned all of them, using 9.4×10^{10} maximum-flow computations, so no diagonal search was needed. For $n = 19$ and $n = 20$ most cubes were run on rented cloud containers (Modal), with the same source code compiled there by gcc 12.2; there a cube takes about 1.3 times as long as on the local machine. Every line returned from the cloud was checked (cube number, prefix and format) before it was logged, and random cubes were rerun locally with identical results. From $n = 15$ on, the number of orbits grows by a factor between 4.0 and 4.4 per unit increase of n, and the running time roughly in proportion. The same program reproduces every known value $a(n)$, $n \leq 15$: it finds a placement with $a(n)$ queens of each colour, and it proves that $a(n) + 1$ is impossible.

6 Verification

We checked the computation in the following ways. The first two use different code but the same method; the others avoid parts of the method.

A second implementation. It is written in Rust, separately from the first program (in C). It differs in four ways:

- it enumerates pairs of bitmasks (R, C) and keeps a pair if its integer code is minimal among its 16 images, a different canonical form;

n	$a(n)$	T	(R, C) orbits	diagonal searches	CPU time
3	1	2	3	0	< 0.01 s
4	2	3	10	0	< 0.01 s
5	4	5	21	0	< 0.01 s
6	5	6	169	0	< 0.01 s
7	7	8	686	0	< 0.01 s
8	9	10	2 450	0	0.01 s
9	12	13	8 526	0	0.02 s
10	14	15	46 481	0	0.22 s
11	17	18	189 337	0	0.93 s
12	21	22	688 447	0	2.34 s
13	24	25	2 994 131	0	13.0 s
14	28	29	12 441 976	0	58.2 s
15	32	33	51 084 014	0	303 s
16	37	38	204 573 866	0	1457 s
17	42	43	817 258 210	0	6489 s
18	47	48	3 562 197 100	0	9.7 h
19	52	53	14 324 893 730	0	50.4 h
20	58	59	57 257 809 825	0	221.9 h

Table 1: Searches proving that no labelling has $W, B \geq T$, with $T = a(n) + 1$. The fourth column counts the (R, C) orbits that pass the count bound; each was pruned by Corollary 3.3 at the root, so the fifth column, which counts the orbits whose diagonals had to be searched, is 0 throughout. Times are total CPU times, on a shared local machine (at most 8 processes at a time) and, for $n = 19, 20$, mostly on cloud containers. For $n \leq 15$ the same program also finds a placement with $a(n)$ queens of each colour.

- it evaluates the Lagrangian bound on the generic closure network, with one auxiliary node per candidate cell;
- it solves that network with FIFO push–relabel [6], with $\lambda = p/1000$;
- it searches any unpruned (R, C) by plain depth-first search that uses only counting bounds.

Its pruning is sound by the analogue of Corollary 3.3. For $n = 16$ and $n = 17$ it repeated the whole search. It reported infeasibility, and its orbit counts (204,573,866 and 817,258,210) agree exactly with those of the first program. For $n = 18$ we ran it on slices: slice j of $m = 199$ consists of the canonical pairs with $R \equiv j \pmod{199}$, where bit i of R is row i, and the slices $j = 0, \ldots, 198$ partition the search. All 199 slices were run and every orbit was pruned; together they contain 3,562,197,100 orbits, exactly the orbit count of the first program and of Burnside's lemma. So for $n \leq 18$ two independent implementations cover the whole search. The second implementation was not run for $n = 19$ and $n = 20$; we estimate that it would need about 290 and 1400 CPU hours on our machine.

Orbit counts by Burnside's lemma. A short independent script counts, for every $n \leq 20$, the G-orbits of pairs (R, C) with $|R||C| \geq T$ and $|R^c||C^c| \geq T$. It uses Burnside's lemma and the action of G on pairs (reversals, transposition and complementation). The script checks the action against the cell maps and checks the orbit counts against brute-force enumeration for $n \leq 8$.

For $n = 16, \ldots, 20$ the counts are 204,573,866, 817,258,210, 3,562,197,100, 14,324,893,730 and 57,257,809,825. They agree with the total numbers of leaves over all cubes of the first program, and for $n \leq 15$ with the first program and the second implementation. So the cubes together visit exactly one pair from each orbit, and none is missing.

Exact LP certificates. For fixed (R, C), the LP relaxation

$$\begin{aligned} \max\ s \quad \text{s.t.} \quad & s \leq \textstyle\sum_{\mathsf{W\text{-}cand}} z_c, \quad s \leq \textstyle\sum_{\mathsf{B\text{-}cand}} u_c, \\ & z_c \leq x_d,\ z_c \leq y_e \quad \text{for each W-candidate } c, \\ & u_c \leq 1 - x_d,\ u_c \leq 1 - y_e \quad \text{for each B-candidate } c, \end{aligned}$$

with $x_d, y_e, z_c, u_c \in [0, 1]$ and $s \geq 0$, bounds $\max \min\{W, B\}$ over the diagonal labels. This uses neither supermodularity nor minimum cuts. We solve it in floating point with HiGHS [9] through SciPy [19], round the dual multipliers to rationals and repair them to a feasible dual solution. Its objective, evaluated in exact rational arithmetic, is then a valid upper bound by weak duality. For each $n = 16, \ldots, 20$ we certified 2000 uniformly random pairs (R, C) passing the count bound and 2000 random "balanced" pairs (both products $\geq T + n$). We also certified the pair (R, C) of the best known board (rows and columns holding a white queen are labelled W) and all pairs within Hamming distance 2 of it, from 529 pairs for $n = 16$ to 821 for $n = 20$. Every certified bound was below T. The largest bounds for $n = 16, \ldots, 20$ were 37.50, 42.11, 47.50, 52.56 and 58.44, against $T = 38, 43, 48, 53, 59$.

SAT with checked proofs. We used two encodings, each with sequential-counter cardinality constraints, and the Glucose solver through PySAT [10]. The first encodes placements directly by cell variables, with a conflict clause for every pair of cells sharing a line, so it does not use Lemma 2.1. The second is the line-labelling encoding. Every unsatisfiability claim was checked with `drat-trim` [22]. Both encodings confirm $a(n)$ for $n \leq 8$. In addition, random canonical (R, C) for $n = 16$ were checked by SAT: 200 uniform and 200 balanced ones (both products at least 56). We also checked 8 random cubes for $n = 12$, $T = 22$, and 3 for $n = 13$, $T = 25$, in which the first 12 row and column labels are fixed. All were unsatisfiable with verified proofs. These checks do not depend on Lemmas 3.1–4.1.

The definition itself. A depth-first search over sets of white queens finds $a(n)$ for $n \leq 8$ directly from the definition, using only $f(S) \geq T$, where $f(S)$ is the number of cells sharing no line with the white set S. A search over all 3^{n^2} placements with the blocking rule of chess gives the same values for $n \leq 4$. A direct CP-SAT model [15] of placements gives the same values for $n \leq 7$.

Unit tests and replays. The min-cut evaluation of Lemma 3.2 agrees with brute-force maximisation over all completions on 18,000 random instances with $3 \leq n \leq 8$, including partially labelled diagonals. The same holds for the second implementation's network on random instances with $n \leq 6$. Switching off the symmetry breaking gives identical answers for $n \leq 11$, and so does switching off the min-cut bound in the second implementation (plain search). Random cubes of every run were rerun and gave log lines identical to the recorded ones, apart from the time.

Placements. Every placement that we report was checked directly for attacks.

Review. An AI referee, run separately from the sessions that designed the method and wrote the programs, read the proofs of Lemmas 2.1–4.1 and both programs adversarially. It wrote the Burnside, LP, brute-force and CP-SAT checks above. It found no error affecting the results. Its minor corrections (a misstated number of fixed lines for the $n = 13$ SAT cubes in an earlier draft, Corollary 3.3 as the precise statement of what the pruning needs, and the Burnside test of completeness) are incorporated. Three rounds of external AI review (ChatGPT Pro) of the code also reran the search for $n = 16$, recomputed the Burnside counts for $n \le 20$ and for the torus runs, and recomputed the largest LP certificates for $n = 20$. The first two rounds found thirteen defects, now corrected, none of which affected the claimed values; the third confirmed the corrections. The resumable drivers trusted logged results in a reused directory whenever the cube list matched, even for a different T; every run reported here used a fresh directory for its (n, T), and the drivers now bind each directory to a manifest of all parameters and refuse a mismatch. The torus program shifted signed integers beyond their width while building symmetry representatives (undefined behaviour); with the corrected program the torus runs for $n = 20$ and $n = 22$ give identical counters. The verification script ignored a failure of the leaf sampler and accepted misspelled stage names; the completion log of the A260680(16) run was missing from the repository; the pinned `drat-trim` did not compile under GCC 14; the LP above was stated with $s \in [0, 1]$; and the solvers accepted out-of-range arguments (a huge Q overflowed, and an empty part of a partition reported infeasibility). The second round found that two drivers starting at the same moment in one directory could publish each other's manifest (creation is now locked and atomic), that $K = 0$ produced an empty list of cubes which a driver reported as infeasible (now rejected, and no driver reports a verdict for an empty work list), that the cube-range argument accepted trailing characters, and that the torus checks did not require each part exactly once; it also asked for a stale sentence in the README to be updated. Auditing for the same patterns, we also made the second implementation reject a slice that cannot contain any row mask. Each correction has a regression test in `scripts/verify.py`.

What has to be trusted. The upper bounds rest on Lemmas 2.1–4.1 and on the first program. Table 2 lists what checked each value. For $n \le 18$ a second, separately written program repeats the whole search with the same result. For $n = 19$ and $n = 20$ only the first program covers the whole search. There the Burnside counts confirm that it visits every orbit; a sample of the cubes computed in the cloud was rerun locally with identical results; and the exact LP certificates and SAT proofs on samples check the bound without Lemmas 3.1–4.1. A self-contained proof certificate would give a value of p and a feasible flow for each of the 5.7×10^{10} orbits for $n = 20$. That would take hundreds of gigabytes. Instead each cube is logged with its counts, and any cube can be rerun in seconds to minutes.

7 Related sequences

The same method gives side results on two related sequences. Each was computed by a single implementation only, and each reproduces all the known terms of its sequence. The counts for A260680 are new as far as we know (the OEIS gives them up to $n = 10$); the torus values confirm known ones.

- *Counting optimal configurations (A260680).* Two solutions are equivalent if they differ by an element of G. We enumerate, without symmetry breaking, all labellings with $\min\{W, B\} \ge a(n)$. We expand each into its placements and count orbits under G. This reproduces the known terms $1, 1, 1, 10, 3, 35, 19, 71, 18, 380$ for $n \le 10$ and gives 696, 4, 2155, 579 and 2298

n	first program	second implementation	other checks
≤ 15	complete	complete	definition search ($n \leq 8$), CP-SAT ($n \leq 7$), SAT+DRAT ($n \leq 8$)
16	complete	complete	Burnside; SAT+DRAT on 400 leaves; LP; replays
17	complete	complete	Burnside; LP certificates; replays
18	complete	complete (199 slices)	Burnside; LP certificates; replays
19	complete	—	Burnside; LP certificates; 24 cloud cubes rerun locally
20	complete	—	Burnside; LP certificates; 24 cloud cubes rerun locally

Table 2: What checked each upper bound. "Burnside" means that the number of orbits visited equals the Burnside count; "LP" the exact LP certificates on sampled (R, C); "replays" reruns of random cubes with identical log lines.

for $n = 11, \ldots, 15$. For $n = 16$ the enumeration, split into 65,536 cubes, finds 416 labellings with $\min\{W, B\} \geq 37$, all with different cell sets, and 26 inequivalent optimal configurations. Since $416 = 16 \cdot 26$, no optimal configuration for $n = 16$ is fixed by a nontrivial element of G. For $n = 17$ the same enumeration finds 80 labellings with $\min\{W, B\} \geq 42$ and 5 inequivalent optimal configurations, again $80 = 16 \cdot 5$. The split version reproduces the values for $n = 10, 12, 13, 14$ first.

- *The torus (A279405).* On the $n \times n$ torus the lines are the rows, the columns and the diagonals $i \pm j \bmod n$. The same method, with the subgroup of symmetries generated by translations, the reflections $i \mapsto -i$ and $j \mapsto -j$, transposition and the colour swap, reproduces the known values for $4 \leq n \leq 14$, $n = 16$ and $n = 18$, each at $a(n)$ and $a(n) + 1$. For $n = 20$ it gives 51, the value that Harries's formula $t(2q) = H(2q)$ [7, Cor. 4] already gives: a placement with 51 of each colour exists, and 52 is impossible (163,664,896 orbits, of which 4675 needed a diagonal search, in 33 CPU minutes). The repository version of the program, which can split the search into parts, caches the representatives of complements and uses unsigned masks, reproduces all these values and all the $n = 20$ counters exactly. For $n = 22$ it gives $64 = H(22)$ [7, Table 1]: it finds a placement with 64 white and 66 black queens, which we checked directly for toroidal attacks, and it shows that 65 of each colour is impossible (2,101,660,623 orbits, of which 80 needed a diagonal search). These two computations are therefore independent confirmations by a different method, not new values; Harries also determined $t(15) = 20$ and $t(17) = 28$. The value for $n = 19$ is still unknown. Odd tori are harder for the method, because the root bound fails for many orbits.

Availability. The code, logs and verification scripts are in the accompanying repository[1]. The script `scripts/verify.py` rebuilds the programs and checks every number quoted here. By default this takes a few minutes. Its `--full` option reruns the complete search for any n. The cloud runs used the dispatcher in `code/modal/`, whose returned results are logged in the same format as the local runs.

[1] https://github.com/jasondavies/peaceable-queens.

Declaration of AI-generated research and authorship

This manuscript is the product of an AI-driven research process. The human author initiated a broader investigation of open problems in the OEIS, and AI systems worked on this problem within it. AI systems produced all of the following: the method, the proofs, the code, the computations and their checks, and the organisation and writing of the text. The system was Claude Opus 5.5 (Anthropic), used through Claude Code. The human author directed the AI systems but wrote no part of the manuscript; the human author also started some of the cloud computations, using commands prepared by the AI system. The AI system's first literature search missed the work of Arevalo [2] and of Harries [7], which a later check found; this version credits it, and corrects the earlier version's claim that $a(16)$–$a(18)$ and the torus values for $n = 20, 22$ were new.

The computational claims are reproduced by `scripts/verify.py`. At the time of writing, the mathematics and the code have been checked only by AI systems and by computer verification. These checks include an adversarial review of the proofs and of both search programs by an independent AI referee, who also wrote the Burnside, LP-certificate, brute-force and CP-SAT checks of Section 6, and three rounds of external AI review of the code (ChatGPT Pro), whose findings are corrected (Section 6). No independent human expert has reviewed the work. The human author has not read this manuscript or checked its arguments. The author takes responsibility for the accuracy of this disclosure.

References

[1] S. Ainley, *Mathematical Puzzles*, G. Bell & Sons, London, 1977.

[2] F. Arevalo, *The sixteenth, seventeenth and eighteenth peaceable queens numbers: 37, 42, 47*, computational research note (AI-assisted), August 17, 2026, updated September 3 and 7, 2026, https://fabianarevalo.com/peaceable-queens; code at https://github.com/farev/Matematica/tree/main/conjectures/peaceable-queens.

[3] R. A. Bosch, *Peaceably coexisting armies of queens*, Optima (Mathematical Programming Society Newsletter) **62** (1999).

[4] R. A. Bosch, *Armies of queens, revisited*, Optima (Mathematical Programming Society Newsletter) **64** (2000), 15.

[5] K. Clinch, M. Drescher, T. Huynh, A. Saffidine, *Constructions, bounds, and algorithms for peaceable queens*, in: Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX 2025), SIAM, 2025, 160–167. doi:10.1137/1.9781611978339.13; arXiv:2406.06974, doi:10.48550/arXiv.2406.06974.

[6] A. V. Goldberg, R. E. Tarjan, *A new approach to the maximum-flow problem*, J. ACM **35** (1988), 921–940. doi:10.1145/48014.61051.

[7] J. P. Harries, *Peaceable queens on even tori: an exact parity formula*, preprint, September 6, 2026 (first version July 27, 2026), with a Lean proof of the main theorem, https://github.com/jphme/math-problems/tree/main/peaceable-queens-even-torus.

[8] P. L. Ivănescu (Hammer), *Some network flow problems solved with pseudo-Boolean programming*, Oper. Res. **13** (1965), 388–399. doi:10.1287/opre.13.3.388.

[9] Q. Huangfu, J. A. J. Hall, *Parallelizing the dual revised simplex method*, Math. Program. Comput. **10** (2018), 119–142. doi:10.1007/s12532-017-0130-5.

[10] A. Ignatiev, A. Morgado, J. Marques-Silva, *PySAT: a Python toolkit for prototyping with SAT oracles*, in: SAT 2018, Lecture Notes in Comput. Sci. **10929**, Springer, 2018, 428–437. doi:10.1007/978-3-319-94144-8_26.

[11] D. M. Kane, *Asymptotic results for the queen packing problem*, preprint (2017), arXiv:1703.04538. doi:10.48550/arXiv.1703.04538.

[12] D. E. Knuth, *The Art of Computer Programming, Volume 4, Fascicle 6: Satisfiability*, Addison-Wesley, 2015, p. 180, exercise 488, and pp. 282–283.

[13] V. Kolmogorov, R. Zabih, *What energy functions can be minimized via graph cuts?*, IEEE Trans. Pattern Anal. Mach. Intell. **26** (2004), 147–159. doi:10.1109/TPAMI.2004.1262177.

[14] OEIS Foundation Inc., *The On-Line Encyclopedia of Integer Sequences*, sequences A250000 (revision 379, including the comments, the extensions, and D. Kamenetsky's file of best known solutions for $12 \leq n \leq 30$), A260680 and A279405, https://oeis.org.

[15] L. Perron, F. Didier, *CP-SAT*, OR-Tools version 9.15, Google, https://developers.google.com/optimization.

[16] K. E. Petrie, *Symmetry breaking in peaceably coexisting armies of queens*, in: CP 2002, Lecture Notes in Comput. Sci. **2470**, Springer, 2002, 763. doi:10.1007/3-540-46135-3_61.

[17] J.-C. Picard, *Maximal closure of a graph and applications to combinatorial problems*, Management Sci. **22** (1976), 1268–1272. doi:10.1287/mnsc.22.11.1268.

[18] S. Prestwich, J. C. Beck, *Exploiting dominance in three symmetric problems*, in: Proceedings of the Fourth International Workshop on Symmetry and Constraint Satisfaction Problems (SymCon'04), 2004, 63–70.

[19] P. Virtanen et al., *SciPy 1.0: fundamental algorithms for scientific computing in Python*, Nature Methods **17** (2020), 261–272. doi:10.1038/s41592-019-0686-2.

[20] B. M. Smith, K. E. Petrie, I. P. Gent, *Models and symmetry breaking for 'peaceable armies of queens'*, in: CPAIOR 2004, Lecture Notes in Comput. Sci. **3011**, Springer, 2004, 271–286. doi:10.1007/978-3-540-24664-0_19.

[21] M. F. van Bommel, K. T. MacEachern, *Armies of chess queens*, Math. Intelligencer **40**(2) (2018), 10–15. doi:10.1007/s00283-017-9771-5.

[22] N. Wetzler, M. J. H. Heule, W. A. Hunt, *DRAT-trim: efficient checking and trimming using expressive clausal proofs*, in: SAT 2014, Lecture Notes in Comput. Sci. **8561**, Springer, 2014, 422–429. doi:10.1007/978-3-319-09284-3_31.

[23] Y. Yao, D. Zeilberger, *Numerical and symbolic studies of the peaceable queens problem*, Exp. Math. **31** (2022), 269–279; arXiv:1902.05886. doi:10.1080/10586458.2019.1616338.