Chess: A Cover-Up
Author(s): Eric K. Henderson, Douglas M. Campbell, Douglas Cook and Erik Tennant
Source: *Mathematics Magazine*, Vol. 78, No. 2 (Apr., 2005), pp. 146-157
Published by: Taylor & Francis, Ltd. on behalf of the Mathematical Association of America
Stable URL: https://www.jstor.org/stable/30044146
Accessed: 10-02-2023 02:16 UTC

Example Suppose our diagonal measurements are $x = 36$ inches and $y = 37$ inches. Then $d(36, 37) = \sqrt{(36^2 + 37^2)/2} = 36.503424$ and $L(36, 37) = 36.5$.

The reader may wish to estimate the error in this linear approximation using Taylor's theorem in two variables [**1**]. If y is the longer diagonal and x the shorter, and if both $|x - x_0|$ and $|y - x_0|$ are known to be less than h, the error can be seen to be less than $y^2h^2/(2x^3)$.

REFERENCE

1. Jerrold E. Marsden and Michael J. Hoffman, *Elementary Classical Analysis*, 2nd ed., W. H. Freeman and Co., New York, 1993.

Chess: A Cover-Up

ERIC K. HENDERSON
DOUGLAS M. CAMPBELL
DOUGLAS COOK
ERIK TENNANT
Computer Science Department
Brigham Young University
Provo, Utah 84602-6576

The game of chess has always proved a rich source of interesting combinatorial problems to challenge mathematicians, logicians, and computer scientists. Apart from playing strategies and end-games, many chess-based problems have been posed over the centuries that tax the limits of symbolic reasoning, such as the n-queens and re-entrant knight's tour problems. However, the modern computer has enabled new approaches to these types of problems, and some of these questions have been explored (and even decided) in ways not previously possible.

One such problem has been attributed to Joseph Kling [**8**], a music producer who operated a chess-oriented coffee house in London from 1852. Kling, who migrated to England from Germany, is described as "a pioneer of the modern style of chess" [**5**]; he published several studies on the game including the popular but short-lived journal *Chess Player*, co-edited with the chess professional Bernhard Horwitz. Kling posed the following question: using a player's eight major chess pieces and no pawns, can all squares on the chessboard be covered (attacked)? At first glance the problem looks easy—the eight pieces collectively have more than enough attack power. Determining whether there is a solution, however, is nontrivial. No simple logical argument has yet been discovered.

Because a chessboard is small, the combinatorial size of chess problems is theoretically tractable. However, only the recent advances in computing power have made this true in practice. In 1989, Robison, Hafner, and Skiena [**8**] applied an exhaustive search to prove that no solutions exist to the Kling cover problem. To accomplish this using the computing power available at the time, they developed a novel technique for reducing, or "pruning," the number of solutions that need to be searched. Their approach, although not immediately intuitive, reduced the search space by more than 99.9%.

With the computing power available today, problems such as this can now be exhaustively searched in a reasonable time with no need for creative pruning of the search space. However, there will always be larger problems pushing the limits of computing power, and methods for reducing the search space help put more of these within reach.

To illustrate the application of computing techniques to perform exhaustive search, we revisit the Kling cover problem in depth. We first attempt to formalize it and present a clear definition of the scope of the problem. We then apply a standard computing algorithm to organize the search space, and present an analysis of three alternative pruning techniques. These techniques, although much less complex computationally and conceptually, give comparable reductions in the size of the search space. In our conclusions we discuss the limitations of exhaustive search techniques in the context of traditional proofs, and the role these methods may play in the future of scientific progress.

Bounding the search space In order to enumerate all the possible ways that the eight pieces may be positioned on the chessboard, we must first select an ordering of the pieces, that is, which piece to place first, second, and so on. Although all orderings produce equivalent sets of final chessboard configurations, certain orderings will prove to allow better optimization of the solution search. We use three different orderings that facilitate our pruning methods:

$$O_1 = [q, r_1, r_2, b_w, b_b, h_1, h_2, k]$$
$$O_2 = [q, r_1, r_2, h_1, h_2, b_w, b_b, k]$$
$$O_3 = [q, b_w, b_b, h_1, h_2, k, r_1, r_2],$$

where q is the position of the queen, r_1 and r_2 the position of the rooks, b_w the position of the bishop on a white square, b_b the position of the bishop on a black square, h_1 and h_2 the position of the knights, and k the position of the king. We indicate the position of a piece on the chessboard using a number from 1 to 64 that corresponds to the occupied square, as shown in FIGURE 1.

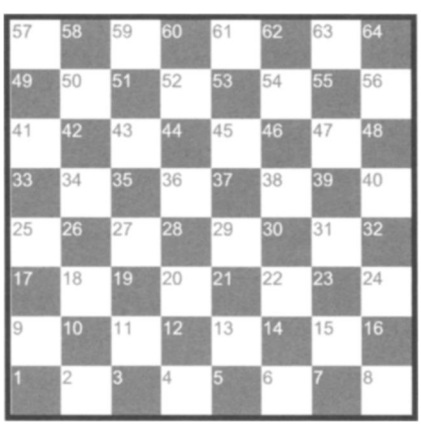


Figure 1 Numbered chessboard

We simplify the implementation by initially removing the restriction that pieces occupy distinct squares, allowing what we refer to as *superpositioning*. We define a *configuration vector* $\hat{c}$ as a vector that holds the numbers indicating the positions of each of the eight pieces relative to a given ordering, which we call a *configuration*. We call the set of all possible configuration vectors for a given ordering the *search space*. Clearly, the size of the search space is independent of the ordering used.

Let us first *bound* the size of the search space by determining the number of possible configuration vectors. Since there are 64 possible positions for each of the eight

pieces (a naïve upper bound that allows for superpositioning), the total number of configuration vectors is bounded by $64^8 = 2^{48}$. Kling's problem restricts one bishop to the 32 black squares and the other bishop to the 32 white squares (following the rules of chess), dropping the bound on the number of configurations to $32 \times 32 \times 64^6$. (FIGURE 2 shows a solution to Kling's problem if both bishops can be on the same color.)

Figure 2 A solution with both bishops on the same color

We can further reduce the bound on the number of configurations by noting that due to the symmetry of the board, some configurations are *equivalent*. Two configurations $\hat{c}_1$ and $\hat{c}_2$ are equivalent if one or more of the following three transformations takes $\hat{c}_2$ to $\hat{c}_1$:

1. *Interchanging* the position of r_1 and r_2 or h_1 and h_2 in $\hat{c}_2$
2. *Rotating* the position of all the pieces in $\hat{c}_2$ through angles of 90°, 180°, or 270° about the center of the board
3. *Reflecting* the position of all the pieces in $\hat{c}_2$ about one of the diagonal axes

Equivalent configurations attack the same number of squares, so it is only necessary to include one configuration in a set of equivalent configurations in the search space. By restricting the positions of certain pieces, we can prevent equivalent configurations from appearing in our search space.

First, we eliminate configurations equivalent due to interchanging rooks and knights by limiting the position of the second rook and second knight: we require that $r_1 \leq r_2$ and $h_1 \leq h_2$. This produces

$$\sum_{k=1}^{64} k = 65 \times 32$$

distinct positions for each of the two pairs of interchangeable pieces and reduces the upper bound on the number of configurations in the search space to

$$32 \times 32 \times (65 \times 32) \times (65 \times 32) \times 64^2.$$

Second, we eliminate configurations equivalent due to rotation. Each configuration has three equivalent forms (90°, 180°, and 270° rotations). We divide the board into four quadrants and limit the position of one of the pieces—in our experiments, the queen—to the lower-left quadrant. This restriction on the queen eliminates all

rotationally equivalent configurations and reduces the upper bound on the number of configurations by a factor of four to $32 \times 32 \times (65 \times 32) \times (65 \times 32) \times 16 \times 64$.

Third, we eliminate the configurations equivalent due to reflections. Observe: limiting the queen to the lower-left quadrant (as described above) also eliminated the diagonal symmetry along the upper-left to lower-right axis. We then limit the white bishop to the 16 white squares underneath the lower-left to upper-right diagonal to eliminate the diagonal symmetry along the lower-left to upper-right axis and reduce the upper bound on the number of configurations by a factor of two to

$$16 \times 32 \times (65 \times 32) \times (65 \times 32) \times 16 \times 64 = 2^{29} \times 65^2 = 2{,}268{,}279{,}603{,}200 \quad (1)$$

or about 2.27 trillion possible configurations in the search space.

Assuming 8 CPU cycles to check a configuration, a computer from the late 1980s running at 25 MHz would take on the order of one week to exhaustively search each of these configurations. Today, a computer running at 2 GHz could finish in less than three hours. If a pruning technique is employed that achieves a 99.9% prune rate, the running time can be reduced to less than 10 *seconds*.

Backtracking The search space can be organized as a tree with eight levels. We place the ith piece (relative to an ordering O_1, O_2, or O_3) on the ith level of the tree. The ith level contains *nodes* representing the positions that the ith piece can occupy on the chessboard. Each node is the parent of a sub-tree containing descendant nodes representing all the possible positions of the remaining unplaced pieces. A leaf node occupies the last level (level eight) in the tree. A *leaf node* represents a configuration where all pieces have been placed. FIGURE 3 illustrates the tree organization relative to O_1.

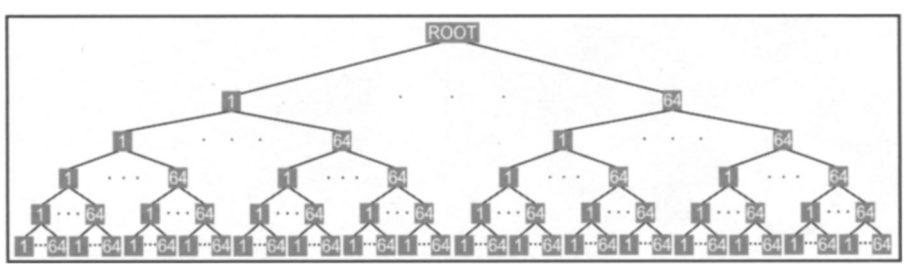


Figure 3 The first five levels of the tree organization for the search space. To simplify the illustration, pieces are allowed to occupy any of the 64 squares.

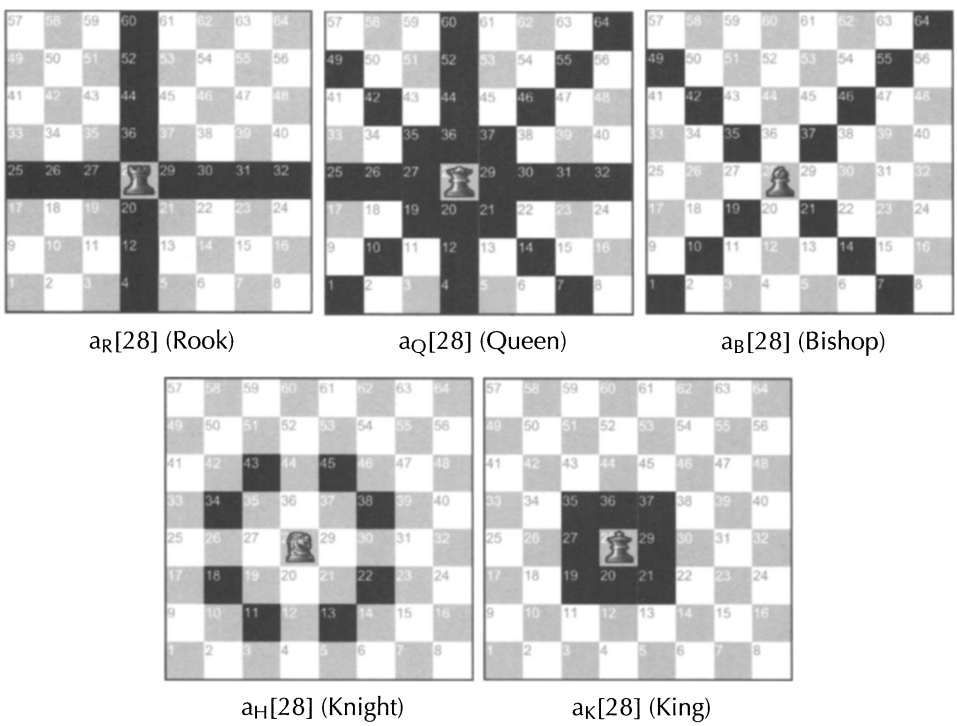
Let $\hat{c}[i]$ be a *partial configuration vector*, by which we mean a placement of the first i pieces according to one of the orderings O_1, O_2, or O_3. In the tree search space, a partial configuration is represented as an *internal node*, by which we mean any nonleaf node. The level of an internal node corresponds to the index i in its partial configuration vector $\hat{c}[i]$. Note that $\hat{c}[8]$, which is $\hat{c}$, can be interpreted as a path from the root to a leaf on the tree (relative to O_1, O_2, or O_3).

An algorithm commonly applied to a tree search space is *recursive backtracking* (also called *depth-first* traversal). Depth-first traversal means that we probe the children of a node x before we probe x's siblings. When we can probe no other child of a node, we backtrack and try the next child of its parent node, that is, its nearest sibling.

Backtracking itself is simply an ordered traversal of the search space, allowing for an exhaustive search of all nodes in the tree. However, by organizing the tree intelligently, it is possible to develop criteria for skipping the evaluation of a node's descendants. We refer to these as *prune rules*. If a prune rule for a node n determines that no

child of n can possibly be a solution, then node n is pruned and none of its descendants are visited. Effective prune rules can yield dramatic reductions in the number of nodes that need to be traversed.

Strong vs. weak solutions We define $a_p[s]$, the *attack pattern of a piece p relative to position s* to be the set of squares attacked, or "covered," when chess piece p is at position $s \in \{1 \ldots 64\}$, without regard to other pieces that may be on the board. FIGURE 4 shows an example of the attack pattern for each piece (relative to position 28). Observe that neither a king nor a knight's attack set is affected by intervening pieces. In contrast, the queen, bishop, and rook's attack patterns may be affected by an intervening piece, that is, a part of the attack pattern may be *blocked*. FIGURE 5 shows an example of a knight blocking a bishop's attack to square 15 and square 8.



$a_R[28]$ (Rook) $a_Q[28]$ (Queen) $a_B[28]$ (Bishop)

$a_H[28]$ (Knight) $a_K[28]$ (King)

Figure 4 Attack patterns for each piece relative to position 28

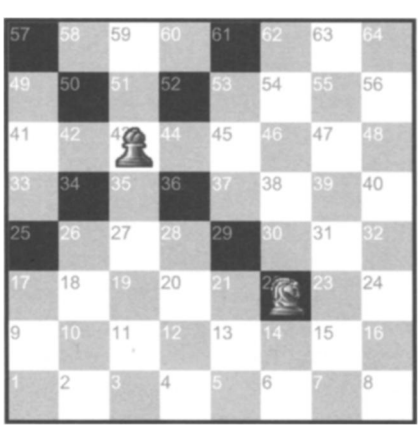


Figure 5 A knight blocking a bishop's attack on squares 15 and 8

Following Robison et al. [**8**], we say that configuration $\hat{c}$ *strongly covers* square s if and only if there is at least one piece in $\hat{c}$ whose attack pattern covers square s and whose attack pattern for square s is not blocked by any other piece. We say that configuration $\hat{c}$ *weakly covers* square s if and only if there is at least one piece in $\hat{c}$ whose attack pattern covers square s regardless of blocking.

A configuration $\hat{c}$ is a *strong (weak) solution* to the Kling problem if and only if (1) each of the 64 squares (including the eight squares occupied by the eight pieces) is *strongly (weakly) covered*, and (2) each piece is on a distinct square (no superpositioning).

A strong solution will also necessarily be a weak solution. It is much easier computationally to check for weak solutions then strong solutions, since blocking effects can be ignored. Thus, to search for strong solutions, it suffices to first compute the set of weak solutions, and then examine this much smaller set for strong solutions.

Three prune rules Let $|\hat{c}[i]|$ denote the cardinality of the set of chessboard squares *weakly* covered by the partial configuration $\hat{c}[i]$. Let $|\hat{c}[i]|_b$ ($|\hat{c}[i]|_w$) denote the cardinality of the set of black (white) chessboard squares weakly covered by $\hat{c}[i]$. Clearly, $|\hat{c}[i]|$ is a nondecreasing function of i, and

$$\left|\hat{c}[i]\right|_b + \left|\hat{c}[i]\right|_w = \left|\hat{c}[i]\right|.$$

PRUNE RULE 1. Order the pieces relative to O_1. Define $mwa[j]$ to be the maximum possible attack potential of the jth piece placed *anywhere* on the board. TABLE 1 gives these values for each piece. At node $\hat{c}_1[i]$, let

$$r[i] = \sum_{j=i+1}^{8} mwa[j].$$

Prune rule 1 is: if

$$\left|\hat{c}_1[i]\right| + r[i] < 64,$$

then prune the node. (That is, the node should be pruned if, at the ith level, the weakly attacked squares and the squares that the remaining pieces can attack cannot cover the board.)

TABLE 1: Maximum weak attack *mwa* potential per level j for pieces ordered as in O_1

Piece	j	$mwa[j]$
Queen	1	27
Rook	2	14
Rook	3	14
Bishop	4	13
Bishop	5	13
Knight	6	8
Knight	7	8
King	8	8

PRUNE RULE 2. Order the pieces relative to O_2, (the last three pieces being white bishop b_w at level 6, the black bishop b_b at level 7, and the king at level 8). Let $m_{black}[i]$ be the maximum number of black squares that can be attacked by the unplaced pieces *after* level i. Let $m_{white}[i]$ be the maximum number of white squares that can be attacked by the unplaced pieces *after* level i (see Table 2). Prune rule 2 is: if

$$\left|\hat{c}_2[i]\right|_b + m_{black}[i] < 32 \quad or \quad \left|\hat{c}_2[i]\right|_w + m_{white}[i] < 32,$$

then prune the node. (That is, the node should be pruned if, at the ith level, the black (white) squares weakly attacked plus the largest possible number of black (white) squares that the remaining pieces can attack is less than 32.)

TABLE 2: Maximum black and white attack potential after positioning level i. Note the asymmetry.

	Attack Potential	
Level i	$m_{black}[i]$	$m_{white}[i]$
1	49	49
2	41	41
3	33	33
4	25	25
5	17	17
6	17	4
7	4	4
8	0	0

PRUNE RULE 3. Order the pieces relative to O_3 (the king at level 6 and the rooks at levels 7 and 8). Let $m_{rows}[i]$ be the number of rows in $\hat{c}_3[i]$ containing three or more nonattacked squares (counting rows with three or more nonattacked squares ensures that the nonattacked squares are in more than two different columns). Since each rook can attack only one row, prune rule 3 is: if

$$m_{rows}[6] > 2 \quad or \quad m_{rows}[7] > 1$$

then prune the node.

Results We measured the effectiveness of our prune rules by running backtracking to traverse the entire search space four times: once with no prune rules, once with only prune rule 1, once with only prune rule 2, and once with only prune rule 3. All four times we found 813 weak solutions to the chess cover problem. It is easy to determine if any of the 813 weak solutions are strong solutions. In fact, none were, confirming that there are no strong solutions to the Kling Chess problem.

One interesting trend to observe is the position taken by the queen. Of the 813 weak solutions the queen occupies one of the four central squares 19, 20, 27, 28 (see FIGURE 1) in 71% of the cases. This central location maximizes the initial attack potential of the queen, and the trend appears to be even stronger when superpositioning is allowed. There are 8,715 weak solutions allowing for superpositioning, with the queen occupying one of the four central squares in 87% of the cases.

Of the 8,715 weak superposition solutions, 1,984 used only seven of the eight pieces. In each of these 1,984 cases, one knight was superimposed on the queen and the

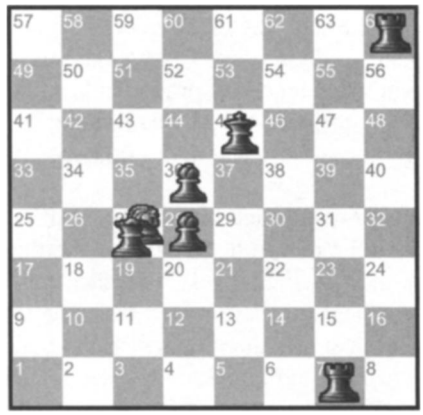


Figure 6 A weak superposition solution using only seven pieces. Notice the queen and a knight both occupy square 27.

second knight was not necessary. FIGURE 6 shows an example of a weak seven-piece superposition solution.

When we checked the 8,715 weak solutions in which superpositioning is allowed, 350 strong solutions were found. Of these 350 strong superposition solutions, the queen occupies one of the four central squares 19, 20, 27, 28 (see FIGURE 1) 97% of the time. In addition, each rook belongs to the outer perimeter (the set of squares that border the edge of the chessboard) 97% of the time. FIGURE 7 shows an example of a strong superposition solution with the queen on square 28 and the rooks on the outer perimeter.

Figure 7 A strong solution allowing superpositioning. Notice the queen and the knight both occupy square 28.

Upon examining the weak solutions, we discovered that the knights sometimes block more of the covered squares than they themselves attack (that is, the addition of a knight *reduced* the total number of squares strongly covered). This phenomenon occurred in 35% of the 813 weak solutions. The effective attack potential of the knights is much less than first appears and may offer the key to a traditional proof.

Effectiveness of the prune rules Equation (1) gives about 2.27 trillion complete configurations in the search space that we must explore. These correspond to leaf nodes

in the tree search space. We measure the effectiveness of a prune rule by determining how many of these complete configurations (leaf nodes) the prune rule removes from the search space. Recall that when a node is pruned, all of its children are pruned from the tree. Since each node is a progenitor to some number of leaf nodes (and internal nodes), pruning the node removes these leaf nodes (and internal nodes) from the search space.

Table 3 shows the results of using each of our prune rules on the Kling Chess Problem. For each prune rule we list the prune results for the three levels 5, 6, and 7. No prunes were recorded for the previous levels. For each level we record the number of prunes and the number of leaf nodes removed from the search space as a result of these prunes. The final column shows the number of leaf nodes removed as a percentage of the total number of leaf nodes in the search space (calculated from (1)). The last row under each prune rule shows the total results for all prunes.

TABLE 3: Results of the prune rules

Tree Level	Prunes	Leaf Nodes Pruned	% of Total Leaf Nodes
		Prune Rule 1	
5	500,796	66,665,963,520	2.94
6	565,321,622	1,211,853,924,352	53.43
7	15,346,724,150	982,190,345,600	43.3
Total	15,912,546,568	2,260,710,233,472	99.67
		Prune Rule 2	
5	2,551,017	83,591,725,056	3.69
6	979,235,545	2,005,474,396,160	88.41
7	2,757,235,103	176,463,046,592	7.78
Total	3,739,021,665	2,265,529,167,808	99.89
		Prune Rule 3	
5	0	0	0
6	1,069,115,100	2,223,759,408,000	98.04
7	706,198,686	24,748,310,328	1.09
Total	1,775,313,786	2,248,507,718,328	99.13

In all three cases our prune rules were most effective at level 6. At this level, prune rule 1 removed 53% of the leaf nodes, prune rule 2 removed 88% of the leaf nodes, and prune rule 3 removed 98% of the leaf nodes. Although each of the three prune rules removed very close to the same number of total leaf nodes (99%), they had very different numbers for total prunings. Prune rule 1 had 15.9 billion prunes, prune rule 2 had 3.7 billion prunes, and prune rule 3 had just 1.7 billion prunes.

The reason prune rule 3 could remove the same number of leaf nodes with much fewer prunes is because it was more effective at level 6, and could therefore remove more leaf nodes with fewer prunes than the other prune rules that had prunes at level 7. The higher a node is on the tree, the more leaf nodes it has as children. Prunes that occur at a higher level remove a larger number of leaf nodes than prunes at lower levels. For instance, prune rule 1 had 565 million prunes at level 6 and 15.3 billion prunes at level 7, yet the prunes at level 6 removed 19% more leaf nodes than the prunes at level 7. Prune rule 2 had 1,000 times more prunes at level 7 than at level 5, but these prunes amounted to just 2 times the number of leaf nodes removed. We were unable to devise a computationally feasible prune rule that could prune a node higher up in the tree than level 5.

A greedy solution The odds of randomly picking a weak solution are 813 in 2,268,279,603,200 or about 1 in 2.79 billion. One might wonder if a simple algorithmic design paradigm such as greedy, which is designed to maximize some parameter at each step in the algorithm, can stumble upon a weak solution. The answer is yes:

> *Take pieces in the order of configuration O_3 and greedily place each piece on a square that maximizes the number of weakly covered squares (break ties by choosing squares closest to the center).*

This greedy formulation produces the weak solution of FIGURE 8. Notice that the queen lies in the four central squares 19, 20, 27, 28 (see FIGURE 1) and the rooks lie on the perimeter.

Figure 8 The weak solution found using the greedy algorithm

Implementation Even with the massive amount of pruning we achieved, it was critical to have an optimized implementation to keep the runtime of the program reasonable. In our implementation, the data representation of a *board* $b[i]$, $1 \leq i \leq 64$, corresponding to the attack pattern $a_p[s]$ (piece p on square s) is an array of Booleans that represents the state of the squares on a chessboard numbered from 1 to 64 as in FIGURE 1. The Boolean $b[i]$ is true (one) if square i is attacked by piece p on square s; $b[i]$ is false (zero) if square i is not attacked by piece p on square s. We implement a *board* as eight contiguous bytes where each Boolean value is represented by one bit. To optimize the search space computation, we initially generate and store as *boards* the *attack patterns* for each of the pieces at every possible position.

Given a configuration $\hat{c}$ we define its *cover pattern*, as the union of the *attack patterns* in its configuration. The *cover pattern* is the result of logically OR-ing the appropriate *boards* representing these *attack patterns*. FIGURE 9 illustrates a cover pattern for the partial configuration with $a_K[18]$, $a_B[16]$, and $a_R[64]$. The chessboard in the figure shows graphically the squares that are attacked by this partial configuration. Next to this is the corresponding cover pattern shown numerically, where each 1 represents an attacked square and each 0 represents a nonattacked square. A configuration is a weak solution if and only if all squares in the representation of the corresponding *cover pattern* have a value of one ($|\hat{c}| = 64$).

To optimize the computation of $|\hat{c}|$, we interpret each row of the chessboard as a single byte. For a given configuration, each bit will be 0 or 1 depending on whether the corresponding chess square is safe or attacked. The value of this byte is then used as an index in a lookup table containing a count of the number of attacked squares,

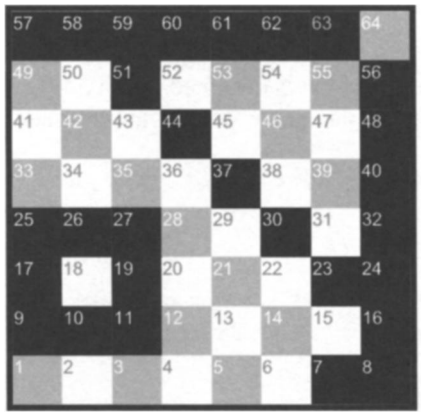


Figure 9 A cover pattern shown graphically and numerically. Attacked squares are shown in black.

which amounts to the number of ones for that particular binary pattern. For example, the top row in FIGURE 9 has all but the last square attacked and the corresponding byte value in this instance would be 254 (FE_{HEX}). This value is then used as an index to the lookup table entry containing the number 7, which corresponds to the number of binary ones in the byte value FE_{HEX}. The computation of $|\hat{c}|$, the number of attacked squares for a given configuration, can thus be computed with eight lookups and a sum, significantly reducing the computational complexity of the operation.

Conclusion We applied backtracking to exhaustively compute the Kling Cover Problem and presented three rules for reducing the search space. Our prune rules, though effective, used simple principles and did not operate close to the root of the tree. But our results revealed interesting trends that might be exploited to create more effective prune rules. Finding effective prune rules requires insight into the nature of the problem. The more effective a prune rule is, the more it begins to resemble a logical argument. A traditional proof that a problem has no solutions is essentially a prune rule that operates on the first node of the tree! Because effective prune rules are difficult to create for the Kling chess problem, we do not expect a general proof to be found easily. But finding more sophisticated prune rules that operate closer to the root may help provide the insight necessary for a traditional proof.

For some combinatorial problems like this one, applying the computing power available today requires little more than the most basic implementation, allowing us to construct an exhaustive proof with a limited understanding and no real insight into the problem itself—in short, using "brute force." In contrast, a traditional proof reveals the nature of a problem using symbols, words, and logic that can be verified by other human beings. A traditional proof appeals to us because it elegantly captures the truth we are attempting to explain. Exhaustive computation may provide us with an answer we can believe, but it leaves the question in some sense unresolved. In spite of this, it is a powerful tool and some classic problems have already benefited from this relatively new technique.

One of the earliest examples of incorporating exhaustive computation into a proof is the Haken-Appel proof of the four-color problem. The problem, which dates to 1852 [**9**], asks whether one can color the regions on an arbitrary two-dimensional map using only four colors, such that no two regions that share a border have the same color. This remained an open problem until 1976, when Wolfgang Haken and Kenneth Appel proved that any map could be shown to be equivalent to one of 1900 special

cases. Then, using a computer, they checked each special case [**1**]. Even though their proof consisted of a traditional logical argument, it made essential use of exhaustive computation.

Haken-Appel's proof has been disparagingly referred to as a "silicon proof," a type of proof for which only another computer can carry out the customary validity check. The mathematical community raised two objections. One objection was aesthetic: the proof failed to reveal a simple, single, fundamental understanding of the problem. A second objection was analytical: the program was thousands of lines long and depended on a compiler and operating system, which themselves had not been proven correct. Despite these limitations, the Haken-Appel proof remains the accepted solution to the four-color problem.

Applying exhaustive techniques to study, and perhaps solve, combinatorial problems is becoming more feasible with each advance in computing power. A traditional proof is naturally preferred to a proof by exhaustion, but some problems remain unsolved using traditional means leaving us to wonder if there are limits to purely symbolic methods. The computer presents us with a tool for attacking problems by exhaustion although its use is somewhat unsatisfying because it involves unverified components such as operating systems, compilers, and chip design. (The floating point error found in Intel's Pentium III processor is a good example of the problems unverified chip design can bring.) Are we as mathematicians going to be forced to give up verifying proofs by hand, just as scientists have been forced to accept from microscopes and telescopes evidence that cannot be directly perceived?

REFERENCES

1. K. Appel and W. Haken, Every planar map is four colorable, *Contemporary Mathematics* **98** (1989), American Mathematical Society, 1–741.
2. ———, The four color proof suffices, *Mathematical Intelligencer* **8** (1986), 10–20.
3. M. J. Beeson, *Foundation of Constructive Mathematics*, Springer-Verlag, Heidelberg, 1985.
4. Bledsoe and Loveland, *Automated Theorem Proving: After 25 Years*, American Mathematical Society, 1989.
5. F. Boase, *Modern English Biography*, Vol. 3, Truro, UK, 1897.
6. M. Gardner, *The Unexpected Hanging and Other Mathematical Diversions*, Simon and Schuster, New York, 1969.
7. ———, *Wheels, Life, and Other Mathematical Amusements*, W. H. Freeman, New York, 1983, pp. 183–193.
8. A. D. Robison, B. J. Hafner, and S. S. Skiena, Eight pieces cannot cover a chess board, *Comp. J.* **32** (1989) 567–570.
9. T. Saaty and P Kainen, *The Four Color Problem*, McGraw Hill, 1977.

50 Years Ago in the MAGAZINE

From "Science in the Modern World," by Marston Morse, Vol. **28**, No. 4, (Mar.–Apr., 1955), 209–211:

> Small wonder, then, that a large proportion of the young mathematicians become technicians in limited fields mostly connected with the foundations. Some leap over the foundations and proceed at once to the front as represented by the material world; these are the ones whom we call applied mathematicians.... Then there are the few—all too few—who aim to build the whole edifice of mathematics, neither lingering too long over the foundations, nor too hastily testing their strength at the front. Such a mathematician was Riemann, who, fifty years before Einstein, built the structure in mathematics whose counterpart in physics is relativity.