

# A Chessboard Coloring Problem for SAT Solvers

January 23, 2003

## Abstract

In this paper we introduce a new satisfiability problem built over an original chessboard coloring problem. The SAT encoding of this problem provide us a new benchmark for SAT solvers. Therefore, we propose an experimental study of this benchmark using state of the art SAT solvers.

## 1 Introduction

This chessboard coloring problem was proposed as the first problem of the Fifth Annual U.S.A Mathematical Olympiad, in 1976. First, one should prove that there is no way to color either black or white each square of a  $4 \times 7$  chessboard, such that the four distinct corners of all rectangles included in the chessboard are not all of the same color, as the thin one in figure 1. A rectangle with 4 corners of the same color is then called a *chromatic rectangle* (CR) (plain line in figure 1). Therefore, chessboards containing at least one CR will be called CR chessboard and those containing no CR will be NCR boards. The second part of the problem was to exhibit a black-white coloring of a  $4 \times 6$  chessboard, with the above property (see figure 2).

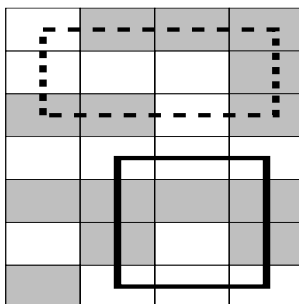


Figure 1: A  $4 \times 7$  CR Chessboard

Beresin, Levine, and Winn [1] give the proof for the first part, and a solution for the second part. Further, they generalize their results to any  $m \times n$  chessboard painted with  $t$  colors. They establish conditions under which the chessboard must have at least a CR. These conditions follow their principle 1:

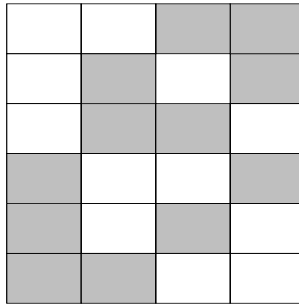


Figure 2: A  $4 \times 6$  NCR Chessboard

**Principle 1** *Let  $N$  squares of the  $m \times n$  chessboard be of the same color, say, black, where  $c_j$  is the number of black squares in column  $j$ . If*

$$\binom{c_1}{2} + \binom{c_2}{2} + \cdots + \binom{c_n}{2} > \binom{m}{2}$$

*then there is a chromatic rectangle.*

As consequence of 1, they state the following principle, easier to invoke, since there is no need to know the colors distribution in each column.

**Principle 2** *Let (at least)  $N$  squares of the  $m \times n$  chessboard be of the same color where  $N = nq + r$  with  $0 \leq r < n$ . If*

$$r \cdot \binom{q+1}{2} + (n-r) \cdot \binom{q}{2} > \binom{m}{2}$$

*then the chessboard has a chromatic rectangle.*

Applying their principle and the above properties to the two and the three colors cases, they find necessary and sufficient conditions on  $n$  and  $m$  to decide if a  $m \times n$  board is a CR-board or a NCR-board. The CR-board set is given by application of principle 2 (and another one for the equality case) and the property 1. The NCR-board set is defined by constructing NCR-board and applying property 2. To summarize, the problem of deciding if a  $m \times n$  board is CR or NCR board is solved for two and three colors. Things begin more complicated as the number of colors increases. As quoted in [1] “to extend the results beyond three colors seems formidable and the authors have made no attempt at this”.

In the following, we reduce our analysis to square  $n \times n$  chessboard, where  $n$  represents the *size* of the board. Now, we are interested in finding the minimal number of colors, i.e. the chromatic number, which allows a valid coloration, i.e. without chromatic rectangle, of a size  $n$  board.

## 2 Encoding of the Problem

An instance of the SAT problem is defined by a set of boolean variables (also called atoms)  $\mathcal{X} = \{x_1, \dots, x_n\}$  and a boolean formula  $\phi: \{0, 1\}^n \rightarrow \{0, 1\}$ . A literal is a variable or its negation. A clause is a disjunction of literals. The formula  $\phi$  is in conjunctive normal form (CNF) if it is a conjunction of clauses. A (truth) assignment is a function  $v: \mathcal{X} \rightarrow \{0, 1\}$ . The formula is said to be satisfiable if there exists an assignment satisfying  $\phi$  and unsatisfiable otherwise. In this problem,  $\phi$  will be in CNF.

Given a chessboard of size  $n \times n$ , the set of its squares  $\mathcal{B} = [1..n] \times [1..n]$  is of size  $n^2$ , therefore elements of  $\mathcal{B}$  can be mapped to a set of variables of size  $n^2$  by an isomorphism:

$$\begin{aligned} flat: \mathcal{B} &\rightarrow \{X_1, \dots, X_{n^2}\} \\ \forall X = (x, y) \in \mathcal{B}, flat(X) &= X_{x+n \times y} \end{aligned}$$

This function provides a flattened representation of the board.

A rectangle in the chessboard can be associated to any pair of points  $A = (a_1, a_2)$  and  $B = (b_1, b_2)$  such that  $\forall i, a_i \neq b_i$ . The corresponding rectangle is then fully defined by its 4 corners:

$$R(A, B) = (X_{a_1+n \times a_2}, X_{b_1+n \times a_2}, X_{a_1+n \times b_2}, X_{b_1+n \times b_2})$$

Note that each rectangle can be referred to by two complementary pairs of points (i.e.  $R((x_1, y_1), (x_2, y_2)) = R((x_1, y_2), (x_2, y_1))$ ). We consider  $\mathcal{R}$  the set of all distinct rectangles for a given chessboard.

From now on, instances of the problem can be fully defined by their two main parameters: number of allowed colors  $N_{col}$  and size of the board  $n$ .

Given a  $(n, N_{col})$  CCP, we first introduce a set of propositional variables  $\{X_{ij} \mid i \in [1..n^2], j \in [1..N_{col}]\}$  ( $X_{ij}$  is true when the  $i^{th}$  variable of the initial problem is of color  $j$ ). We have to state that each square has one and only one color:

$$\forall i \in [1..n^2] \left( \bigvee_{j \in [1..N_{col}]} X_{ij} \wedge \bigwedge_{j, k \in [1..N_{col}], j \neq k} (\neg X_{ij} \vee \neg X_{ik}) \right)$$

Then, we state that there are no chromatic rectangle, by using our previous characterization of the rectangles :

$$\begin{aligned} \forall i_1, i_2, i_3, i_4, (X_{i_1}, X_{i_2}, X_{i_3}, X_{i_4}) &\in \mathcal{R} \\ \bigwedge_{k \in [1..N_{col}]} (\neg X_{i_1 k} \vee \neg X_{i_2 k} \vee \neg X_{i_3 k} \vee \neg X_{i_4 k}) \end{aligned}$$

This states that for a given color  $k$  the four propositional variables  $X_{i_1 k}, X_{i_2 k}, X_{i_3 k}, X_{i_4 k}$  forming a rectangle are not simultaneously true.

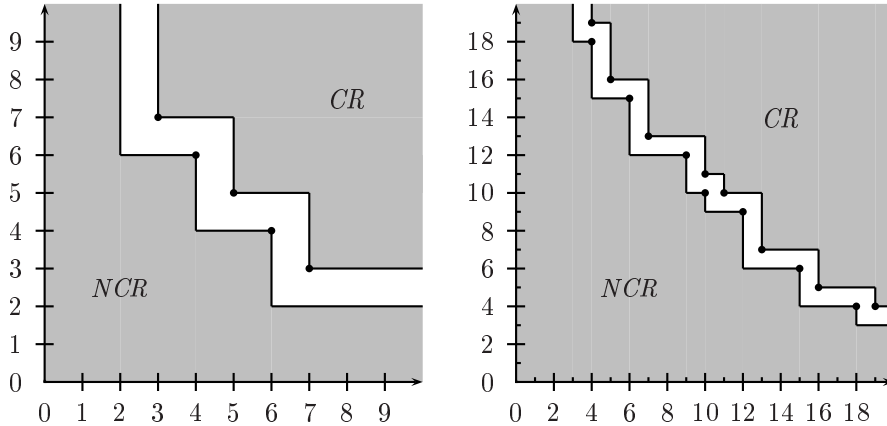


Figure 3: Board sizes with two colors (left graphic) and boards sizes with three colors (right graphics).

The conjunction of these two statements is the instance of the SAT problem that encodes a CCP. This SAT instance involves  $n^2 \times N_{Col}$  variables and  $n^2(1 + 0.5(N_{Col}-1)N_{Col} + 0.25N_{Col}(n^2 - 2n + 1))$  clauses. This corresponds for instance to 172 clauses for a (4, 3) CCP and 63 clauses for a (3, 3).

An obvious upper bound for the chromatic number of a size  $n$  board is  $n$ . Indeed, one has just to paint each row with a different color, so there is no chromatic rectangle. Better upper bounds will be given in the experimental results section. The lower bound follows principle 2.

One could easily generalize this problem to multi-dimensional chessboards and rectangles. But one should remark that any NCR-board of dimension 2 can be immediately extended to a solution for dimension 3 by repeating it  $n$  times (if  $n$  is the size). Then, it is obvious that any 3D-rectangle has at least one of its face in a 2D-NCR chessboard and thus its 8 corners are not of the same color. Hence, if  $k$  is the chromatic number for a 2D board, the chromatic number for greater dimensions is less or equal to  $k$ . This remark is still valid for any dimensional extension.

### 3 Experimental Results

#### 3.1 SAT Solver Algorithms

We briefly describe here the SAT solver we have used in our experiments. There are two classes of solvers: the complete solvers based on the branch and backtracking algorithm called the Davis Putnam Logemann Loveland (DPLL) algorithm [3] and the incomplete solvers with local search method.

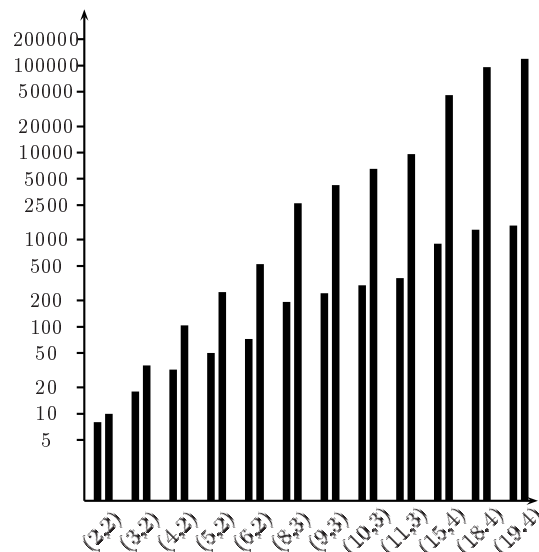


Figure 4: Number of variables (left bar) and clauses (right bar) generated with a chessboard of size  $n \times n$  and  $c$  colors (couple  $(n,c)$ ).

DPLL builds a binary trie where each node is a recursive call to the DPLL procedure with a reduce set of clauses (elimination of the clauses where the variable is TRUE and reduction of the clauses where the variable is FALSE). All the leaves represent an empty clause except one if the problem is satisfiable.

The complete solvers :

- Zchaff [11, 8]: The DPLL algorithm was augmented with an efficient implementation of Boolean constraint propagation and a new decision heuristic (VSIDS).
- BerkMin [4]: Clauses are chronologically sorted. This “age” is taken into account in all the choices and gives new heuristics.
- Limmat [2]: No publication allows to understand clearly Limmat but we know the principal idea is to detect early conflicts in the boolean constraint propagation queue.
- Satz [7]: A powerful heuristic based on unit propagation is added to DPLL.

Local search method consider a set of all possible configurations. The search takes a random assignment and directs it step by step to a correct assignment thanks to a good neighborhood function and a good evaluation function.

The incomplete solvers :

- GASAT [5]: GASAT is a hybrid evolutionary algorithm with a specific crossover operator and a tabu search procedure.
- UnitWalk [6]: UnitWalk combines local search with unit propagation.
- DLMSAT [10]: The Lagrange multiplier is used to lead the search out of a local minimum.
- Walksat [9]: Walksat is a pure local search algorithm.

### 3.2 Experimental Conditions

Lot of algorithms own parameters. We tested them with the default parameters and with a limit time (one hour). Each incomplete solver was tested three times on each instance. If no solution was found, the best number of false clauses was given. For the complete solvers, if the instance is UNSAT we can not say the number of false clauses.

### 3.3 Results

All the tests were executed on a Sun Fire 880 (4 CPU UltraSPARC III 750 Mhz, 8 Go of RAM).

Table 1 show the results of complete solvers on color benchmarks. For little instances (under 250 variables), there are good results. After, the instances are too large and the solver can not solve them (except BerkMin with color-10-3 and Zchaff with color-15-4).

Table 2 show the results of incomplete solvers. The success rate is not written because it is 100% or 0%. When the success rate is 0%, we give the best number of false clauses found durin the tests. For little instances, we can observe no clear dominance of one solver over the other ones. For large instances, two solvers domine, GASAT and DLMSAT. DLMSAT performs on satisfiable instances and GASAT gives best results on unsatisfiable instances.

## 4 Conclusion

## References

- [1] May Beresin, Eugene Levine, and John Winn. A Chessboard Coloring Problem. *The College Mathematics Journal*, 20(2), 1989.
- [2] Armin Biere. Limmat. <http://www.inf.ethz.ch/personal/biere/projects/limmat/>.

Benchmarks				BerkMin	Zchaff	Limmat	Satz
instances	var	cls	sat	sec.	sec.	sec.	sec.
color-3-2	18	36	Y	0.01	0.00	0.00	0.01
color-4-2	32	104	Y	0.00	0.01	0.00	0.01
color-5-2	50	250	N	0.02	0.06	0.04	0.03
color-6-2	72	522	N	0.01	0.06	0.03	0.05
color-8-3	192	2608	Y	0.01	69.17	1.47	-
color-9-3	243	4212	Y	0.09	127.6	14.62	-
color-10-3	300	6475	Y	123.81	-	-	-
color-11-3	363	9559	N	-	-	-	-
color-15-4	900	45675	Y	-	1804.88	-	-
color-18-4	1296	95904	?	-	-	-	-
color-19-4	1444	119491	N	-	-	-	-

Table 1: Complete solvers (if time out or bug then "-")

Benchmarks				GASAT	DLMSAT	UnitWalk	Walksat
instances	var	cls	sat	sec.	sec.	sec.	sec.
color-3-2	18	36	Y	0.00	0.00	0.00	0.00
color-4-2	32	104	Y	0.00	0.00	0.00	0.00
color-5-2	50	250	N	(1 cl.)	(1 cl.)	(1 cl.)	(1 cl.)
color-6-2	72	522	N	(4 cl.)	(4 cl.)	(6 cl.)	(4 cl.)
color-8-3	192	2608	Y	0.02	0.01	0.00	0.01
color-9-3	243	4212	Y	0.09	0.02	0.01	0.17
color-10-3	300	6475	Y	318.243	6.07	(10 cl.)	(2 cl.)
color-11-3	363	9559	N	(4 cl.)	(4 cl.)	(20 cl.)	(13 cl.)
color-15-4	900	45675	Y	1501.39	36.01	(18 cl.)	(8 cl.)
color-18-4	1296	95904	?	(22 cl.)	(46 cl.)	(75 cl.)	(59 cl.)
color-19-4	1444	119491	N	(34 cl.)	(51 cl.)	(82 cl.)	(82 cl.)

Table 2: Incomplete solvers (if not assignment found then the best number of false clauses is written between parentheses)

- [3] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, Jul 1962.
- [4] Eugene Goldberg and Yakov Novikov. BerkMin: A fast and robust SAT-solver. In *Design, Automation, and Test in Europe (DATE '02)*, pages 142–149, Mar 2002.
- [5] Jin-Kao Hao, Frédéric Lardeux, and Frédéric Saubion. Evolutionary computing for the satisfiability problem. *soumis à 3rd European Workshop on Evolutionary Computation in Combinatorial Optimization (EvoCOP 2003)*.
- [6] Edward A. Hirsch and Arist Kojevnikov. UnitWalk: A new SAT solver that uses local search guided by unit clause elimination. PDMI preprint 9/2001, Steklov Institute of Mathematics at St.Petersburg, 2001.
- [7] Chu Min Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *Proc. of the IJCAI'97*, pages 366–371, 1997.
- [8] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an Efficient SAT Solver. In *Proc. of the 38th Design Automation Conference (DAC'01)*, Jun 2001.
- [9] Bart Selman, Henry A. Kautz, and Bram Cohen. Noise strategies for improving local search. In *Proc. of the AAAI, Vol. 1*, pages 337–343, 1994.
- [10] Yi Shang and Benjamin W. Wah. A discrete lagrangian-based global-search method for solving satisfiability problems. *Journal of Global Optimization*, 12(1):61–99, Jan 1998.
- [11] Lintao Zhang, Conor F. Madigan, Matthew H. Moskewicz, and Sharad Malik. Efficient conflict driven learning in a boolean satisfiability solver. In *Proc. of the International Conference on Computer-Aided Design (ICCAD '01)*, pages 279–285, Washington - Brussels - Tokyo, Nov 2001. IEEE.