Resolution Tunnels for Improved SAT Solver Performance

Michal Kouril[1] and John Franco[1]

University of Cincinnati, Cincinnati, OH 45221-0030, USA
mkouril@ececs.uc.edu,
WWW home page: http://www.ececs.uc.edu/~mkouril

Abstract. We show how to aggressively add uninferred constraints, in a controlled manner, to formulas for finding Van der Waerden numbers during search. We show that doing so can improve the performance of standard SAT solvers on these formulas by orders of magnitude. We obtain a new and much greater lower bound for one of the Van der Waerden numbers, specifically a bound of 1132 for $W(2,6)$. We believe this bound to actually be the number we seek. The structure of propositional formulas for solving Van der Waerden numbers is similar to that of formulas arising from Bounded Model Checking. Therefore, we view this as a preliminary investigation into solving hard formulas in the area of Formal Verification.

1 Introduction

Resolution is a general procedure that may be used to determine whether a given CNF expression has a model and to supply a certificate of unsatisfiability if it doesn't. The idea predates the often cited work reported in [22] and for decades resolution has been one of the primary engines for CNF SAT solvers. In the last 10 years tree resolution, in the form of variants of DPLL [10], has given way to DAG resolution through the introduction of clause learning and recording during search. This and other ideas have led to a spectacular improvement in the performance of resolution-based SAT solvers. Hardware and algorithmic improvements have together contributed to perhaps an order 10^9 speed-up in SAT solving over the past 15 years and, consequently, some problems considered very difficult then are now considered trivial. But there remain many problems which are considered hard, for example in the important domains of protocol and hardware verification.

The last 15 years has also seen some brilliant theoretical work that has revealed exponential lower bounds for tree and DAG resolution, and has illuminated the reasons for it, when resolution is applied to "sparse," unsatisfiable CNF formulas (e.g. [2–4, 8, 13, 18, 28]). In such cases, very large resolvents must be created first, then resolved to get the smaller clause constraints that play a significant role in establishing the refutation. Generating the large resolvents is expensive, particularly since exponentially many have to be generated. In DPLL terms, this means a search space of great breadth may have to be explored.

Metaphorically, we may think of search breadth as a mountain that must be climbed; on the other side of the mountain the search breadth may be significantly reduced due to the short resolvents that are finally generated. Clearly, we need to find some way to "tunnel" through this mountain and arrive quickly in the fertile valley of low-breadth search space, if it exists. It is the aim of this paper to explore this possibility for Boolean expressions with a particular structure.

In the Satisfiability literature, the term "tunnel" has been applied to Stochastic Local Search algorithms, a class which includes members of the WalkSAT, GSAT, and other families [14, 16, 17, 20, 25]. In that context, one may think of a location as an assignment of values to variables and the height at a location as the number of constraints falsified by the assignment at that location. Then, for a given, hard Boolean expression, there are generally many mountain peaks and the objective is to locate and enter the deepest valley. Changing the value of a single variable merely moves the current location up and down the side of a mountain but changing values of several variables permits "tunneling" into another valley. This use of tunneling is to be distinguished from the way we use it: in our context there is one principal mountain to get over or around and the valley on the other side can be quite wide. To reach the top of our mountain one must wait for many large constraints to be learned. But, in many cases, we cannot afford to wait: to reach the valley in a reasonable time, *constraints must be efficiently added before they are learned.*

There are several ways to do this, some safe and some risky. A reasonably efficient method for finding a safe tunnel, which is actually more like a cut, through the mountain has been identified in [30] for a class of non-CNF formulas. Given Boolean functions $b_1, b_2, ..., b_m$, let $\phi = b_1 \wedge \cdots \wedge b_m$ and let $V' = \{v_1, \cdots, v_k\}$ be a subset of variables occurring in ϕ. Re-index the functions so that at least one variable of V' occurs in b_i for all $1 \leq i \leq n$, and no variables of V' occur in b_i, for $n < i \leq m$. Let $\mathcal{M}' = \{M'_1, \cdots, M'_{2^k}\}$ be the set of all possible truth assignments to the variables in V'. Write $b_j|_{M'_i}$ or $\phi|_{M'_i}$ to mean the variables of b_j or ϕ, respectively, are assigned values according to assignment M'_i, or are left unassigned if they are not assigned in M'_i.

Theorem 1. ([30]) *For every* $1 \leq i \leq 2^k$*, define*

$$\delta_i = \bigvee_{1 \leq j \leq n} (\overline{b_j|_{M'_i}} \wedge \bigvee_{\substack{1 \leq s \leq 2^k \\ s \neq i}} b_j|_{M'_s}).$$

Then, for any $1 \leq i \leq 2^k$*, if* $\delta_i \equiv$ `False` *then* $\phi|_{M'_i}$ *is satisfiable if and only if* ϕ *is satisfiable.* ∎

According to Theorem 1, it may be possible to assign values to a particular set of variables in such a way that the satisfiability of ϕ is unaffected. Those values may not be inferred at all, but they are nevertheless *safe* to assign and doing so reduces ϕ somewhat. A safe assignment is a generalization of the notion of *autarky* [21], defined for CNF formulas, to formulas that are conjunctions

of Boolean functions. The test implied in the theorem can be conducted with reasonable efficiency (see [30] for details).

Although safe assignments can be useful, substantial decreases in computational effort require more aggressive use of uninferred constraints. This need not lead to errors: a constraint can be retracted during search if it is inferred `False`. The field of non-monotonic reasoning has provided many insights on how this might be done. Of course, we would like to be able to add constraints aggressively without having to worry about retracting them and, as we show in this paper, this can sometimes be done to achieve a result which is an approximation to the actual result. In other words, the extra constraints may prevent an optimal solution from being returned but are weak enough to admit suboptimal solutions that are not far from optimal. But, if the extra constraints are too weak, a solver may take too much time and not find a good suboptimal result. So, we need a good "heuristic" for adding constraints in some optimal way.

At this point in time, it seems the spectacular tunneling success we seek, which will be a consequence of our choice of tunnel heuristic, is practical only by a careful analysis of the specific structure of a given formula. In this paper we underscore this point by developing aggressive tunnel heuristics for formulas associated with the problem of finding Van der Waerden numbers (described in the next section). Such formulas are currently extremely difficult for off-the-shelf SAT solvers, even though most of them are satisfiable. By adding aggressive tunnel constraints to the formulas, however, we are able to find the best bound yet, by far, for $W(2, 6)$. Our analysis serves as an example for attempting aggressive tunnel heuristics for other hard problems.

2 Van der Waerden numbers and Satisfiability

Van der Waerden numbers arise from a set partition problem [29]. Partition the set $S_n = \{1, ...n\}$ of the first n positive consecutive integers into k classes. Let $P_{n,k}(l)$ be a proposition that is `True` if and only if all partitions of S_n into k classes contain at least one arithmetic progression of length l in at least one class. The k, l Van der Waerden number, denoted $W(k, l)$, is the minimum n for which $P_{n,k}(l)$ is `True`.

There is no known closed form expression for $W(k, l)$ and all but five of the first few numbers are unknown. Table 1 shows all the known Van der Waerden numbers. In 1979 $W(3, 4)$ became the most recent addition to this table.

$k \setminus l$	**3**	**4**	**5**
2	9	35	178
3	27		
4	76		

Table 1. Known Van der Waerden numbers.

Upper and lower bounds on some of the remaining numbers have been derived but they are so far apart that they are of little practical use. An unpublished general upper bound is [31]

$$W(k,l) \leq e^{e^{(1/k)^{e^{e^{(l+110)}}}}},$$

and a general lower bound, due to the Lovász local lemma, is [31]

$$W(k,l) > \left(\frac{k^l}{elk}\right)(1+o(1)).$$

Work on specific Van der Waerden numbers has sharpened some of these bounds as the results of Table 2 (taken from [11]) show.

$k \setminus l$	**3**	**4**	**5**	**6**	**7**	**8**
2	9	35	178	> 695	> 3702	> 7483
3	27	> 291	> 1209	> 8885	> 43854	> 161371
4	76	> 1047	> 10436	> 90306	> 262326	
5	> 125	> 2253	> 24044	> 177955		
6	> 206	> 3693	> 56692			

Table 2. Known bounds on van der Waerden numbers.

The number $W(k,l)$ can be found by determining whether solutions exist for certain formulas of a class of CNF formulas described in Table 3. We refer to a formula of this class, with parameters n, k, l, by $\psi^n_{k,l}$. A solution exists for $\psi^n_{k,l}$ if and only if $n < W(k,l)$. So, several of these formulas may be solved for various values of n until that boundary is reached. In this manuscript $\psi^n_{k,l}$ is treated as a set of clauses to make some algorithmic operations easier to express.

Variables	**Subscript Range**	**Meaning**
$v_{i,j}$	$1 \leq i \leq n, 1 \leq j \leq k$	$v_{i,j} \equiv 1$ iff $i \in C_j$
Clauses	**Subscript Range**	**Meaning**
$\{\bar{v}_{i,r}, \bar{v}_{i,s}\}$	$1 \leq i \leq n, 1 \leq r < s \leq k$	i is in at most one class
$\{v_{i,1}, \ldots, v_{i,k}\}$	$1 \leq i \leq n$	i is in at least one class
$\{\bar{v}_{r,j}, \bar{v}_{r+1,j}, \ldots, \bar{v}_{r+l-1,j}\}$ $\{\bar{v}_{r,j}, \bar{v}_{r+2,j} \ldots, \bar{v}_{r+2(l-1),j}\}$ ... $\{\bar{v}_{r,j}, \bar{v}_{r+t,j} \ldots, \bar{v}_{r+t(l-1),j}\}$	$1 \leq r \leq n-l+1$ $1 \leq j \leq k$ ... $t = \lfloor (n-r)/(l-1) \rfloor$	no arithmetic progression of length l in C_j

Table 3. Formula $\psi^n_{k,l}$ for finding Van der Waerden numbers. Equivalence classes are named $C_1, C_2, ..., C_k$ for convenience.

The nature of $\psi_{k,l}^n$ is such that the number of solutions increases with n up to a point, then decreases. The formulas, as expected, become more difficult in the latter range. This difficulty may prevent the boundary from being reached or even approached. In that case, there is no choice but to accept a lower bound which is the largest n for which a solution to $\psi_{k,l}^n$ is found.

SAT solvers have been applied to the above formulations with the results shown in Table 4 (taken from [11]), all lower bounds. Except in one case, all these bounds are greatly inferior to those obtained analytically.

$k \setminus l$	**3**	**4**	**5**	**6**	**7**	**8**
2	9	35	178	> 341	> 614	> 1322
3	27	> 193	> 676	> 2236		
4	76	> 416				
5	> 125	> 880				
6	> 194					

Table 4. Bounds on van der Waerden numbers obtained by SAT solvers ([11]).

We are interested in $W(2, 6)$. When we employ aggressive tunneling techniques, we can push the lower bound to 1132 from the previously known best value of 696 [27]. The reason we have a bound instead of the actual number is that aggressive tunneling forces our SAT solver to be incomplete. We emphasize that although an incomplete solver precludes finding a refutation for the formula, it does provide an apparently tight bound for $W(2, 6)$. We believe the bound is tight because it is obtained using three widely different tunnels.

Our interest in examining tunnels for Van der Waerden numbers is due partly to this being an interesting problem to many mathematicians but mainly because the propositional formulas for solving Van der Waerden numbers have a structure that is similar to formulas found in Bounded Model Checking and other practical applications. Therefore, we view this work as a preliminary to investigations on problems in the area of Formal Verification, among others.

3 Formulation for $W(2, 6)$ and the tunnels

3.1 Formulation

Since we consider a case where $k = 2$, we reinterpret variables to remove some of the constraints shown in Table 3. The formulas we consider are described in Table 5. They use single index variables. For puposes of discussion, variable indices have been translated so that v_0 and v_1 are the middle variables. In doing so, the number of variables is always even. It is straightforward to consider odd variable formulas as well and we leave this for the reader. In what follows, n is even when we consider formulas $\psi_{2,6}^n$.

Variables	Subscript Range	Meaning
v_i	$-n/2 < i \leq n/2$	$v_i \equiv 1$ if $i + n/2 \in C_1$ $v_i \equiv 0$ if $i + n/2 \in C_2$
Clauses	**Subscript Range**	**Meaning**
$\{\bar{v}_i, \bar{v}_{i+1}, \ldots, \bar{v}_{i+5}\}$	$-n/2 < i \leq n/2 - 5$	
$\{\bar{v}_i, \bar{v}_{i+2} \ldots, \bar{v}_{i+10}\}$		no arithmetic progression
...	...	of length 6 in C_1
$\{\bar{v}_i, \bar{v}_{i+t} \ldots, \bar{v}_{i+5t}\}$	$t = \lfloor (n/2 - i + 1)/5 \rfloor$	
$\{v_i, v_{i+1} \ldots, v_{i+5}\}$	$-n/2 < i \leq n/2 - 5$	
$\{v_i, v_{i+2} \ldots, v_{i+10}\}$		no arithmetic progression
...	...	of length 6 in C_2
$\{v_i, v_{i+t} \ldots, v_{i+5t}\}$	$t = \lfloor (n/2 - i + 1)/5 \rfloor$	

Table 5. Formula $\psi^n_{2,6}$, n even, for finding $W(2,6)$. Classes are named C_1, C_2 for convenience.

3.2 Motivating the use of a tunnel - analyzing $\psi^n_{2,l}$

The approach to finding $W(2,6)$ that is described below is the result of an analysis of the performance of an off-the-shelf SAT solver on formulas $\psi^n_{2,l}$ (Table 3), and patterns of variable assignments satisfying those formulas.

Figure 1 shows SAT solver performance on $\psi^n_{2,5}$, for various values of n. The vertical axis measures the number of nodes of the search space at the depth indicated by the horizontal axis. In all cases, the entire search space was explored, even if the input formula was satisfiable, but the results are similar if the solver stops immediately upon discovering a solution. Although the displayed results have been obtained using stock settings, similar results, which are not shown, apply for various settings of SAT solver parameters. According to Figure 1 there is a performance mountain that has roughly the same shape, regardless of n. The mountain tails off to a point of little significance after rising to a formidible peak. In addition to the mountain is a smaller peak which behaves more like a wave since it always appears near n.

From performance curves and known Van der Waerden numbers and bounds we have observed the following:

Observation 1. Consider a performance plot of search breadth vs. depth for any common SAT solver applied to $\psi^n_{2,l}$. Such a plot has two maxima, the greatest of which occurs at approximately the same depth, say $W(2,l)/(2(l-1))$, for $n > W(2,l)/(l-1)$. The value of the greatest maximum is approximately the same for $n > W(2,l)/(l-1)$ and several orders of magnitude greater than the search breadth at depth $W(2,l)/(l-1)$.

Observation 2. $W(2,l) \approx l * W(2,l-1)$, at least for small l.

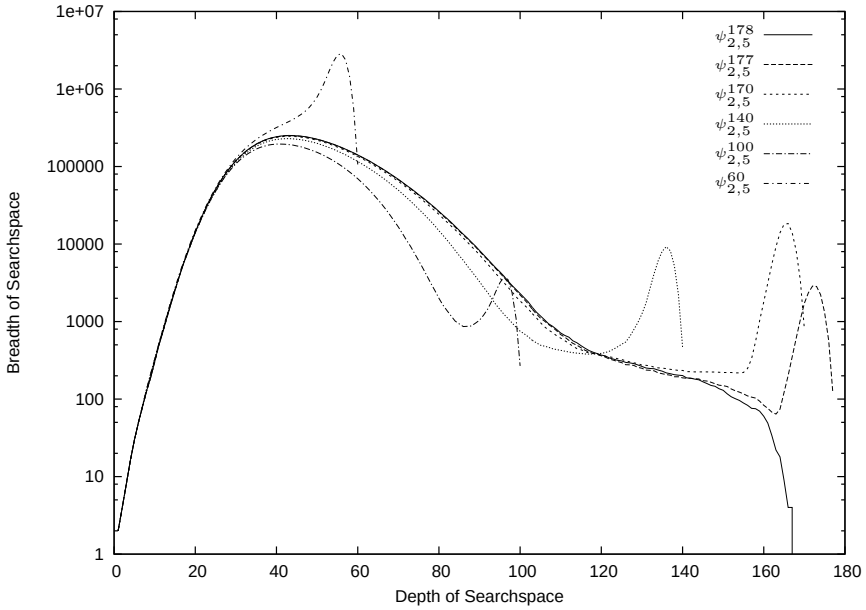

Fig. 1. Typical SAT solver performance on $\psi_{2,5}^n$ for various values of n. Each curve shows the breadth of the search space at the search depth indicated on the horizontal.

From the above observations we propose a way to solve the very difficult $W(2,6)$ formulas. Start by searching for solutions to $\psi_{2,6}^{n_o}$ where n_o is large enough so that the mountain *can* be crossed but smaller than the suspected value of $W(2,6)$. By Observations 1 and 2, n_o should be greater than about 210. To get through the mountain, append a *tunnel* to the formula. We have developed three applicable tunnels which are specially designed to take advantage of certain structural or analytical characteristics of the formulas and are described separately in Section 3.4. The mountain is crossed when the search breadth is considered "low." From then on, by Observation 1, search is accomplished efficiently no matter what. Retract the tunnel. Add clauses so that the solver is effectively seeing $\psi_{2,6}^{n_o+x}$ for $x = 1, 2, 3, ...$ until search breadth falls to 0. The depth at which this occurs is a bound on $W(2,6)$.

Details are given in the following sections.

3.3 Procedure for finding a bound on $W(2,6)$

We used a SAT solver designed specifically to solve formulas $\psi_{2,l}^n$, and which incorporates special optimizations as outlined in Section 3.6. The outline below briefly describes the search process using the special solver.

1. The solver is started on the input $\psi_{2,6}^{n_o}$, for some even n_o, plus a collection of clauses representing the tunnel. The parameter n_o is determined according to the description given in Section 3.2. There are three types of tunnels that

have been applied. Detailed descriptions are given in Section 3.4. All three yield the same lower bound on $W(2,6)$ as stated in Section 3.4. Other tunnels are possible but were not tried.

2. Instead of a depth-first, or even priority-driven evaluation of the search space, as is commonly practiced, the solver conducts a strictly breadth-first evaluation. We chose breadth-first search to find all solutions (not just the first one) so we can at the point of n_o remove the tunnel and continue the search by extending the existing set of solutions. The order in which variables are considered for evaluation is fixed for all branches of the search tree, regardless of values assigned previously. The order represents a reflection around the center variable and is given as follows, from left to right: $v_0, v_1, v_{-1}, v_2, v_{-2}, ...$ This is undoubetdly an inferior choice with respect to building the entire search space. Reasons for this choice are given in Section 3.5.
3. The search reaches depth n_o because there are so many solutions to $\psi_{2,6}^{n_o}$ and the tunnel constraints do not filter some of them. At this depth all variables of $\psi_{2,6}^{n_o}$ have been assigned values on all leaves of the search tree. Clauses from the set $\psi_{2,6}^{n_o+1} \setminus \psi_{2,6}^{n_o}$ are then added to the clause database of the solver and the search commences as before. The tunnel is retracted.

   Remark: The tunnel has been designed so that the mountain has been crossed, for the most part, by this time. From this point on, the breadth of the search space is moderately small because the values of many variables are inferred on all branches, so the search continues quickly.
4. The following is repeated for $m = 2, 3, ...$ until the search breadth becomes 0: when the search depth reaches $n_o + m - 1$, clauses from $\psi_{2,6}^{n_o+m} \setminus \psi_{2,6}^{n_o+m-1}$ are added to the solver's clause database and the search continues. When the search breadth becomes 0, a lower bound of $n_o + m$ is found for $W_{2,6}$.

   Remark: Upon completion of every iteration of this step, a non-zero search breadth means at least one solution for $\psi_{2,6}^{n_o+m-1}$ exists, hence $n_o + m$ is a lower bound for $W(2,6)$.
   Remark: There is no clause recording. Experiments show that zChaff, for example, does not benefit from clause recording on this family of formulas. We believe this is because the structure of the formulas is such that inferences can only be determined at high search depth. This is why we needed the tunnels in the first place.

With the above modifications to the SAT solver, and the improved formulation shown in Table 5, a greatly improved bound for $W(2,6)$ was obtained. However, this was not the case if no tunnels had been added at the outset.

3.4 The tunnels

Three different aggressive tunneling techniques to improve SAT solver performance are described in the subsections below. All three yield the same new lower bound of 1132 for $W(2,6)$. This is significant for two reasons: 1) this is a

big improvement over the previous best bound of 696 [27]; 2) since three different techniques stopped at the same point, we conjecture $W(2,6) = 1132$.

First tunnel: The first tunnel arises from an analysis of solutions to $\psi_{2,l}^{W(2,l)-1}$ (recall $W(2,l)-1$ is the greatest n for which a solution to $\psi_{2,l}^{n}$ exists). Figures 2 and 3 help visualize patterns associated with a typical solution to $\psi_{2,4}^{34}$ and $\psi_{2,5}^{177}$, respectively. For both figures, the solution is shown as a sequence of 0's and 1's representing an assignment of values to variables in increasing order of index, from left to right. Each curve shown is called a *solution curve* and is derived from the solution in the figure. A solution curve rises one unit for every 1 encountered and drops one unit for every 0 encountered when traversing the solution from left to right. Observe that the number of peaks in each figure is $l-1$ and, more importantly, there appears to be a limited length pattern of reverse symmetry, which we call a *reflected pattern*, in the vicinity of at least one of the peaks.

We conjecture the following based on observations such as those depicted in Figures 2 and 3:

Conjecture 1. For every $\psi_{2,l}^{W(2,l)-1}$ there exists a solution that contains at least one reflected pattern of length $W(2,l)/((l-1)*2)$ with the middle positioned somewhere between $W(2,l)/(l-1)$ and $W(2,l)*(l-2)/(l-1)$.

The tunnel is designed as a filter for consecutive variable assignment patterns that are not reverse symmetric. The tunnel consists of clauses involving s consecutive variables where, s is even and by Conjecture 1 and Observation 2, $s < \lfloor 1068/10 \rfloor = 106$. We tried several values for s including 60, 80, 100, even 150 and all worked, but speed increased significantly with increasing s up to 150.

The first family of tunneling constraints is shown in Table 6. By using negative indices in Table 5 these constraints remain fixed as n grows. Otherwise, the tunnel would have to move with n.

Second tunnel: From the results obtained by using the first tunnel alone, it was observed that some small assignment patterns *did not* occur in solutions. The second tunnel filters those patterns. This action is opposite to that of *forcing* patterns to occur which is the objective of the first tunnel. Consequently, the first tunnel spans a small number of variables because longer forced reverse symmetric patterns do not exist in any solution to $\psi_{2,6}^{W(2,6)-1}$, but the second tunnel spans all variables because non-solution patterns can appear anywhere in the clauses of $\psi_{2,6}^{W(2,6)-1}$.

The second family of tunneling constraints is shown in Table 7. The maximum value of 20 for t is a compromise: the tunnels needs to be big enough to have an impact but small enough to keep some solutions around to the end. The number 20 was determined by experiment on $\psi_{2,6}^{n}$ formulas.

Third tunnel: In [27] solutions to $\psi_{2,6}^{n}$ are found for various values of n including 565 and 695. The latter number implies the best known analytic lower bound on $W(2,6)$. Regarding the solution to $\psi_{2,6}^{565}$, re-index the assigned variables

$$v_{-282}, ..., v_0, v_1, ...v_{282}$$

to

$$v_{-564}, v_{-562}, ..., v_0, v_2, ..., v_{562}, v_{564}$$

and add unassigned variables

$$v_{-565}, v_{-563}, ..., v_1, ..., v_{563}, v_{565}.$$

Observe that this operation will not introduce any arithmetic progression among the even indexed variables. The assignment to the even indexed variables is the third tunnel. Search for a solution to $\psi_{2,6}^{1131}$ among the variables above, keeping the values of the even indexed variables fixed. Search completes in a reasonable time with a solution.

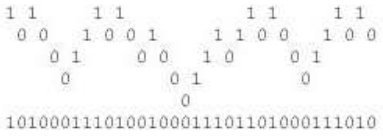

Fig. 2. Typical solution curve for $\psi_{2,4}^{34}$. This is the largest satisfiable formula for $W(2,4)$.

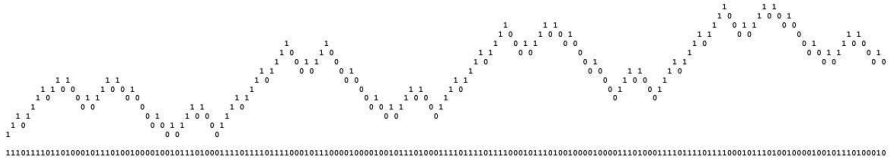

Fig. 3. Typical solution curve for $\psi_{2,5}^{177}$. This is the largest satisfiable formula for $W(2,5)$. For both figures, the solution is shown as a sequence of 0's and 1's representing an assignment of values to variables in increasing order of index, from left to right. The top of each curve rises one unit for every 1 encountered and drops one unit for every 0 encountered when traversing the solution from left to right.

Remarkably, using any of the three tunnels or even combining them leads to the same bound of 1132 for $W(2,6)$.

Tunnel Clauses	Subscript Range	Meaning
$\{v_{-i}, v_{i+1}\}, \{\bar{v}_{-i}, \bar{v}_{i+1}\}$	$0 \le i < s/2$	force $v_{-i} \equiv \bar{v}_{i+1}$.

Table 6. First tunnel constraints added to $\psi_{2,6}^{n_o}$ initially, then retracted after "tunneling."

Tunnel Clauses	Subscript Range	Filters
$\{v_i, \bar{v}_{i+t}, v_{i+2t}, \bar{v}_{i+3t}, v_{i+4t}, \bar{v}_{i+5t}\}$	$-n/2 < i \le n/2 - 5t$	010101
$\{\bar{v}_i, v_{i+t}, \bar{v}_{i+2t}, v_{i+3t}, \bar{v}_{i+4t}, v_{i+5t}\}$	$1 \le t \le 20$	101010
$\{v_i, v_{i+t}, \bar{v}_{i+2t}, \bar{v}_{i+3t}, v_{i+4t}, \bar{v}_{i+5t}, \bar{v}_{i+6t}, v_{i+7t}\}$		00110110
$\{\bar{v}_i, \bar{v}_{i+t}, v_{i+2t}, v_{i+3t}, \bar{v}_{i+4t}, v_{i+5t}, v_{i+6t}, \bar{v}_{i+7t}\}$		11001001
$\{v_i, \bar{v}_{i+t}, \bar{v}_{i+2t}, v_{i+3t}, \bar{v}_{i+4t}, \bar{v}_{i+5t}, v_{i+6t}, v_{i+7t}\}$		01101100
$\{\bar{v}_i, v_{i+t}, v_{i+2t}, \bar{v}_{i+3t}, v_{i+4t}, v_{i+5t}, \bar{v}_{i+6t}, \bar{v}_{i+7t}\}$	$-n/2 < i \le n/2 - 7t$	10010011
$\{v_i, v_{i+t}, \bar{v}_{i+2t}, \bar{v}_{i+3t}, \bar{v}_{i+4t}, v_{i+5t}, v_{i+6t}, \bar{v}_{i+7t}\}$	$1 \le t \le 20$	00111001
$\{\bar{v}_i, \bar{v}_{i+t}, v_{i+2t}, v_{i+3t}, v_{i+4t}, \bar{v}_{i+5t}, \bar{v}_{i+6t}, v_{i+7t}\}$		11000110
$\{\bar{v}_i, v_{i+t}, v_{i+2t}, \bar{v}_{i+3t}, \bar{v}_{i+4t}, \bar{v}_{i+5t}, v_{i+6t}, v_{i+7t}\}$		10011100
$\{v_i, \bar{v}_{i+t}, \bar{v}_{i+2t}, v_{i+3t}, v_{i+4t}, v_{i+5t}, \bar{v}_{i+6t}, \bar{v}_{i+7t}\}$		01100011

Table 7. Second tunnel constraints added to $\psi_{2,6}^{n_o}$ initially, then retracted after "tunneling."

3.5 Choosing the search heuristic

Variables are always considered in the following order, regardless of assignment:

$$v_0, v_1, v_{-1}, v_2, v_{-2}, v_3, v_{-3}, ...$$

The reason is that by assigning values to middle variables first, there is a good chance of inferences developing, symmetrically, for higher and lower indexed variables. If, say, the lower indexed variables were assigned first, perhaps half of the potential future inferences would not be realized early. This is particularly important for the first tunnel where the inferences are needed to appear outside of the tunnel variables.

3.6 Optimizations and special procedures

We were able to get the lower bound of 1132 for $W(2, 6)$ with either tunnel using the special solver described in Section 3.3 with the optimizations mentioned below. However, for stock SAT solvers we needed to apply a combination of the first and second tunnels to get a bound of 1132.

The following optimizations to the special solver were used.

1. Data structures specifically designed for very fast checking of arithmetic progressions and inferences were used. Estimated speed up due to these structures is about a factor of 25.

2. Without optimization, all nodes of the search space would have two children representing `True` and `False` assignments to the variable of that node. However, we generate two children only when inferences force that to be necessary. In other words, we implement a primitive form of conflict-analysis but do not record the result as a clause as is done in modern SAT solvers. Estimated speed up due to this optimization is about a factor of 2.
3. The static search heuristic described in Section 3.5 accounts for an estimated factor of 2 speed up.

4 Conclusions

We have shown how an analysis of performance curves and solution patterns of a class of CNF formula can present insight for designing effective tunnels through search depth of high breadth. We have chosen to experiment with formulas for finding Van der Waerden numbers since they are a difficult class for standard SAT solvers, apparently because clause recording (learning) is ineffective. By tunneling, reasonable yet uninferred constraints are added just long enough to get to a search depth that has relatively low breadth. Then the tunnel is retracted with the hope that not all solutions have been destroyed by the tunnel. Doing so, we found a solution to $\psi_{2,6}^{1131}$ and therefore a new, significantly improved lower bound on the number $W(2,6)$ of 1132.

The symmetric nature of the formulas and solution played an important part in designing effective tunnels and three tunnels were tried with the same result. This type of symmetry is common in many formula classes that arise from practical applications including problems of formal verification. We believe other difficult problems will succumb to tunneling.

We believe performance curves of search breadth vs. depth, such as shown in Figure 1, provide a "fingerprint" for tunnel effectiveness of problem classes. We speculate the fingerprint will not change much qualitatively from one solver to the next but cannot rule out that possibility. Assuming this, we make some claims about the performance curve with respect to hardness as follows. If a performance curve should reach a peak at very high depth, tunneling is unlikely to be effective. The family of `queue` formulas from bounded model checking seem to fall into this category. We speculate that early, large peaks mean delayed inferences and force exhaustive exploration at search depths corresponding to many unassigned variables.

Although the development of general purpose propositional solvers that work well on all inputs is strongly desireable, at this point it is hard to imagine how this is going to be accomplished. However, a general purpose solver can be assisted greatly by giving special consideration to an input problem class, mining its structure for some property that may be used to improve search. This is what we have done with tunneling. Although we do not forsee generally applicable principals for developing tunnels, we do believe that normal human intuition is

enough to uncover exploitable structure in many cases. The Van der Waerden numbers illustrate both points.

Acknowledgements

This work was supported in part by DoD contracts MDA-904-99-C-4547, MDA-904-02-C-1162 and a fellowship grant of the Ohio Board of Regents.

References

1. Akers, S. B.: Binary decision diagrams. IEEE Transactions on Computers **C-27(6)** (1978) 509–516
2. Beame, P., and Pitassi, T.: Simplified and improved resolution lower bounds. Proc. 37th Annual Symposium on Foundations of Computer Science (1996) 274–282.
3. Beame, P., Karp, R. M., Pitassi, T., and Saks, M.: On the complexity of unsatisfiability proofs for random k-CNF formulas. Proc. 30th Annual Symposium on the Theory of Computing (1998) 561–571.
4. Ben-Sasson, E., and Wigderson, A.: Short proofs are narrow - resolution made simple. Journal of the Association for Computing Machinery **48** (2001) 149–169.
5. Brace, K. S., Rudell, R. R., and Bryant, R. E.: Efficient implementation of a BDD package. Proc. 27th ACM/IEEE Design Automation Conf. (1990) 40–45.
6. Bryant, R. E.: Graph-based algorithms for Boolean function manipulation. IEEE Trans. on Comp. **C-35(8)** (1986) 677–691.
7. Burch, J., Clark, E., and Long, D.: Symbolic model checking with partitioned transitions relations. In: Intnl. Conf. on VLSI (Halaas, A., and Denyer, P.B., eds.), IFIP Transactions, North-Holland (1991) 49–58,
8. Chvátal, V., and Szemerédi, E.: Many hard examples for resolution. Journal of the Association for Computing Machinery **35** (1988) 759–768.
9. Cooke, M.: Van der Waerden numbers. Available from `http://home.comcast.net/ rm_cooke/vdw.html`.
10. Davis, M., Logemann, G., Loveland, D.: A Machine Program for Theorem Proving. Communications of the Association of Computing Machinery **5** (1962) 394–397.
11. Dransfield, M.R., Liu, L., Marek, V., and Truszczynski, M.: Using Answer-Set Programming to study van der Waerden numbers. Logic and Artificial Intelligence Laboratory, Computer Science Department, College of Enginnering, University of Kentucky. Available from http://cs.engr.uky.edu/ai/vdw/.
12. Dransfield, M., and Bryant, R. E.: Using ordered binary decision diagrams to solve highly structured satisfiability problems. Unpublished technical report CMU-CS-1996, Carnegie Mellon University (1996).
13. Galil, Z.: On resolution with clauses of bounded size. SIAM Journal on Computing 6 (1977) 444–459.
14. Gent, I.P., and Walsh, T.: Towards an understanding of hill-climbing procedures for SAT. Proc. 11th National Conference on Artificial Intelligence (1993) 28–33.
15. Groote, J. F.: Hiding propositional constants in BDDs. Logic Group Preprint Series 120, Utrecht University (1994).
16. Gu, J.: Efficient local search for very large-scale satisfiability problems. ACM SIGART Bulletin **3**(1) (1992) 8–12.

17. Gu, J., Purdom, P.W., Franco, J., and Wah, J.: Algorithms for the Satisfiability problem: a survey. DIMACS Series on Discrete Mathematics and Theoretical Computer Science **35** (1997) 19–151.
18. Haken, A.: The intractability of resolution. Theoretical Computer Science **39** (1985) 297–308.
19. Lee, C. Y.: Representation of switching circuits by binary-decision programs. Bell System Technical Journal **38** (1959) 985–999.
20. McAllester, D., Selman, B., and Kautz, H.A.: Evidence for invariants in local search. Proc. International Joint Conference on Artificial Intelligence (1997) 321–326.
21. Monien, B., Speckenmeyer, E.: Solving satisfiability in less than 2^n steps. Discrete Applied Mathematics **10** (1983) 117–133.
22. Robinson, J.A.: A machine-oriented logic based on the resolution principle. Journal of the ACM **12** (1965) 23–41.
23. Pan, G., and Vardi, M. Y.: Search vs. symbolic techniques in satisfiability solving. Proc. Seventh International Conference on Theory and Applications of Satisfiability Testing (SAT 2004).
24. San Miguel Aguirre, A., and Vardi, M. Y.: Random 3-SAT and BDDs: The plot thickens further. In: Principles and Practice of Constraint Programming, (2001) 121-136.
25. Selman, B., Kautz, H.A., Cohen, B.: Noise strategies for improving local search. Proc. 12th National Conference on Artificial Intelligence (1994) 337–343.
26. Somenzi, F.: Colorado University Decision Diagram package. Available from `http://vlsi.colorado.edu/~fabio/CUDD/`.
27. Song, H.Y., Golomb, S.W., and Taylor, H.: Progressions in Every Two-coloration of Z_n. Journal of Combinatorial Theory, Series A **61**(2) (1992) 211–221.
28. Urquhart, A.: Hard examples for resolution. Journal of the Association for Computing Machinery **34** (1987) 209–219.
29. Van der Waerden, B.L.: Beweis einer Baudetschen Veermutung. Nieuw Archief voor Wiskunde **15** (1927) 212–216.
30. Weaver, S.A., Franco, J., and Schlipf, J.S.: Extending Existential Quantification in Conjunctions of BDDs. University of Cincinnati Technical Report.
31. Weisstein, E.W., et al.: van der Waerden Number. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/vanderWaerdenNumber.html.