#A71 INTEGERS 11 (2011)

ON COMPUTATION OF EXACT VAN DER WAERDEN NUMBERS

Tanbir Ahmed
Department of Computer Science and Software Engineering, Concordia University, Montréal, Canada
ta_ahmed@cs.concordia.ca

Received: 1/30/11, Revised: 9/14/11, Accepted: 11/23/11, Published: 12/13/11

Abstract

The *van der Waerden number* $w(k; t_0, t_2, \ldots, t_{k-1})$ is the smallest integer m such that in every partition $P_0 \cup P_1 \cup \cdots \cup P_{k-1}$ of the set $\{1, 2, \ldots, m\}$, there is always a block P_j that contains an arithmetic progression of length t_j. In this paper, we report the exact value of the previously unknown van der Waerden number $w(2; 4, 9)$, some lower bounds of $w(2; 5, t)$, and polynomial upper-bound conjectures for $w(2; 4, t)$ and $w(2; 5, t)$. We also present an efficient SAT-encoding of $w(k; t_0, t_1, \ldots, t_{k-1})$ for $k \geqslant 3$ using which we have computed the exact value of $w(3; 3, 3, 6)$ and some lower bounds of $w(3; t_0, t_1, t_2)$.

1. Introduction

Van der Waerden's theorem [18] can be formulated (as in Chvátal [5]) as follows: Given any positive integer k and positive integers $t_0, t_1, \ldots, t_{k-1}$, there is an integer m such that given any partition

$$\{1, 2, \ldots, m\} = P_0 \cup P_1 \cup \cdots \cup P_{k-1} \tag{1}$$

there is always a class P_j containing an arithmetic progression of length t_j. Let us denote the least m with this property by $w(k; t_0, t_1, \ldots, t_{k-1})$. We say the van der Waerden number $w(k; t_0, \ldots, t_{k-1})$ *classical* if and only if $t_0 = t_1 = \cdots = t_{k-1}$. Only six such non-trivial numbers are known: $w(2; 3, 3) = 9$ [5], $w(2; 4, 4) = 35$ [5], $w(3; 3, 3, 3) = 27$ [5], $w(2; 5, 5) = 178$ [16], $w(4; 3, 3, 3, 3) = 76$ [3], and $w(2; 6, 6) = 1132$ [12]. We denote partitions as strings; for example, 00110011 means $P_0 = \{1, 2, 5, 6\}$ and $P_1 = \{3, 4, 7, 8\}$. By a *good partition*, we mean a partition of the form (1) such that no P_j contains an arithmetic progression of t_j terms. We have recently published some previously unknown van der Waerden numbers in [1, 2].

To make this article self-contained, we repeat some definitions that we provided in [2]. We construct an instance F of the satisfiability problem (described in the following paragraph) with n variables for the van der Waerden number $w(k; t_0, \ldots, t_{k-1})$ such that F is satisfiable if and only if $n < w(k; t_0, \ldots, t_{k-1})$.

To describe the satisfiability problem, we require a few other definitions. A *truth assignment* is a mapping f that assigns each variable in $\{x_1, x_2, \ldots, x_n\}$ a value in $\{0, 1\}$. The complement $\bar{x}_i$ of each variable x_i is defined by $f(\bar{x}_i) = 1 - f(x_i)$ for all truth assignments f. Both x_i and $\bar{x}_i$ are called *literals*: a *clause* is a set of (distinct) literals and a *formula* is a family of (not necessarily distinct) clauses. A truth assignment *satisfies a clause* if and only if it maps at least one of its literals to 1; the assignment *satisfies a formula* if and only if it satisfies each of its clauses. A formula is called *satisfiable* if it is satisfied by at least one truth assignment, otherwise, it is called *unsatisfiable*. The problem of recognizing satisfiable formulas is known as the *satisfiability problem*, or SAT for short. The definitions given in this paragraph are adapted from Chvátal and Reed [6].

To check the satisfiability of the generated instance, we need to use an algorithm that either solves the instance providing a satisfying assignment, or says that the formula is unsatisfiable. We have, at our disposal, two kinds of algorithms: complete and incomplete. A complete algorithm like DPLL [7] finds a satisfying assignment if one exists, otherwise, correctly says that no satisfying assignment exists and the formula is unsatisfiable. A local-search based incomplete algorithm (see Ubcsat-suit [17] for examples) is generally faster (as we can control the number of iterations, runs, and several other parameters) than a DPLL-like algorithm, but may fail to deliver a satisfying assignment when there exists one. A good partition is a proof of a lower bound for a certain van der Waerden number irrespective of its means of achievement. So incomplete algorithms are handy for obtaining good partitions and improving lower bounds of van der Waerden numbers. When they fail to improve the lower bound any further, we may turn to a complete algorithm. Surely, determining the exact value of a van der Waerden number involves proving the instance corresponding to $w(k; t_0, t_1, \ldots, t_{k-1})$ to be unsatisfiable. We have developed and used an efficient stand-alone implementation of the DPLL [7] algorithm in this purpose. Our implementation is engineered to run faster on unsatisfiable instances corresponding to van der Waerden numbers. For a brief description on this implementation, see Section 3 of [2].

We have also described a distributed application of our stand-alone implementation by splitting the search-space into independent parts (no message-passing between processes). For a brief description, see Section 4 of [2]. In [2], we have reported the exact values of $w(2; 3, 17)$ and $w(2; 3, 18)$ to be 279 and 312, respectively. In Section 2 of this article, we report the exact value of $w(2; 4, 9)$ and lower bounds of $w(2; 5, t_1)$ with $7 \leqslant t_1 \leqslant 11$.

In Section 3.1, we describe an efficient SAT-encoding of $w(k; t_0, t_1, \ldots, t_{k-1})$ for $k \geqslant 3$. Using this encoding, as well as the implementation and distribution techniques mentioned above, we have computed a previously unknown van der Waerden number $w(3; 3, 3, 6)$ and provided some lower bounds of $w(3; t_0, t_1, t_2)$.

For computation, we have used 2.2 GHz AMD Opteron processors of the `cirrus`

cluster at Concordia University and 2.8 GHz Intel Xeon E5462 processors at Université de Sherbrooke (under Quebec High Performance Computing Network, RQCHP). The lower bounds* are computed using some combinations of local-search based algorithms and our implementation of DPLL. The exact values are determined using our implementation of DPLL.

2. On $w(2; t_0, t_1)$

In this section, we report that the exact value of the previously unknown van der Waerden number $w(2; 4, 9)$ is 309. We also provide new lower bounds for $w(2; 5, t_1)$, with $7 \leqslant t_1 \leqslant 11$. Based on the experimental data, we provide conjectures for upper bounds of $w(2; 4, t)$ and $w(2; 5, t)$.

2.1. A New Number: $w(2; 4, 9) = 309$

In [1], we provided the lower bound $w(2; 4, 9) > 254$. The lower bound was improved to $w(2; 4, 9) \geqslant 309$ by Kullmann [13] using local search based algorithms. We have proved the exact value of $w(2; 4, 9)$ to be 309 using our implementation of the DPLL algorithm. A good partition of the set $\{1, 2, \ldots, 308\}$ corresponding to $w(2; 4, 9)$ is as follows:

```
11111111 01110101 10001100 10111010 10011111 01101110 11111111
00111111 11011101 10111110 01010111 01001100 01101011 11111010
11000110 01011101 01001111 10110111 01111111 10011111 11101110
11011111 00101011 10100110 00110101 11A11101 01100011 00101110
10100111 11011011 10111111 11001111 11110111 01101111 10010101
11010011 00011010 11101111 1111 (A is arbitrary).
```

It took us 176.28 years of CPU-time (roughly 330 days of run-time using 200 processors) to prove that there is no good partition of the set $\{1, 2, \ldots, 309\}$ corresponding to $w(2; 4, 9)$. We split (see [2] for the splitting technique and the branching rule) the DPLL-tree into 16 independent parts, each of which were split further into 1024 independent parts.

In such a large computation where thousands of distributed branches of the DPLL-tree have run on hundreds of processors for almost a year, we hope we have not fallen into the trap of an undetected hardware failure (an electricity failure is natural and every detected hardware-failure was re-run from the last state of the search) or a file-manipulation error on a particular branch which unfortunately could contain a good partition of the set $\{1, 2, \ldots, 309\}$. We welcome interested readers with proper resources to conduct another search to verify our result.

*Examples of good partitions corresponding to the lower bounds can be found at `http://users.encs.concordia.ca/~ta_ahmed/vdw.html`

2.2. Some New Lower Bounds of $w(2;5,t)$

In this section, we provide the following new lower bounds of $w(2;5,t)$ for $7 \leqslant t \leqslant 11$:

$w(2;5,7) \geqslant 260$, $w(2;5,8) \geqslant 331$, $w(2;5,9) \geqslant 473$, $w(2;5,10) > 557$, and $w(2;5,11) > 736$.

2.3. Two Conjectures on the Upper Bounds of $w(2;4,t)$ and $w(2;5,t)$

We propose the following polynomial bounds as conjectures for $w(2;4,t)$ and $w(2;5,t)$:

Conjecture 1. There exists a constant $c_1 \geqslant 1$ such that $w(2;4,t) \leqslant c_1 t^3$.

Based on the known values (18, 35, 55, 73, 109, 146, 309) of $w(2;4,t)$ for $t \geqslant 3$, we obtain the following recursion with $d \geqslant (309-146)/9^2 \approx 2$:

$$\begin{aligned} w(2;4,t) \leqslant w(2;4,t-1) + d \cdot t^2 &\leqslant w(2;4,3) + d\left(4^2 + 5^2 + \cdots + t^2\right) \\ &\leqslant 18 + 2\left(1^2 + 2^2 + \cdots + t^2\right) - 28 \\ &< 2 \cdot \left(\frac{t(t+1)(2t+1)}{6}\right) < 2t^3. \end{aligned}$$

Conjecture 2. There exists a constant $c_2 \geqslant 1$ such that $w(2;5,t) \leqslant c_2 t^4$.

Based on the known values and bounds of $w(2;5,t)$ for $t \geqslant 3$, we obtain the following recursion with $d \geqslant 1$:

$$\begin{aligned} w(2;5,t) \leqslant w(2;5,t-1) + d \cdot t^3 &\leqslant w(2;5,3) + d\left(4^3 + 5^3 + \cdots + t^3\right) \\ &\leqslant 22 + 1\left(1^3 + 2^3 + \cdots + t^3\right) - 36 \\ &< \left(\frac{t(t+1)}{2}\right)^2 < 0.25 \cdot 4 \cdot t^4 = t^4. \end{aligned}$$

3. On $w(k;t_0,t_1,\ldots,t_{k-1})$ for $k \geqslant 3$

3.1. A New Encoding

For any partition $P_0 \cup P_1 \cup \cdots \cup P_{k-1}$ of the set $\{1,2,\ldots,n\}$, we want to prohibit the existence of arithmetic progressions of length t_j in block P_j $(0 \leqslant j \leqslant k-1)$. Here, we present a simple but useful idea of binary-encoding[§] of the blocks of partition for SAT-encoding of van der Waerden numbers. Instead of taking nk variables $x_{i,j}$ with $1 \leqslant i \leqslant n$ and $0 \leqslant j \leqslant k-1$ (as described in [1]), we take nr variables $x_{p,q}$ with $1 \leqslant p \leqslant n$ and $0 \leqslant q \leqslant r-1$, where $r = \lceil \log_2(k) \rceil$.

[§] The idea of using binary variables to represent the bits of a binary representation of a non-negative integer is not new. This formulation is well-known in integer linear programming (see Garfinkel and Nemhauser [9]); but the idea was never used before in the current context.

To prove that an instance with n variables is unsatisfiable, the DPLL algorithm implicitly enumerates all the 2^n solutions, that is, systematically evaluates all possible solutions without explicitly evaluating all of them. As we have fewer variables in the proposed encoding, we have less number of solutions to enumerate.

Let a block P_j for $0 \leqslant j \leqslant k-1$ be represented in binary such that $j = \sum_{q=0}^{r-1} b_{j,q} 2^q$, where $b_{j,q}$ is the q-th bit in the r-bit binary representation of j.

An integer i in $\{1, 2, \ldots, n\}$ belongs to a block P_j if and only if j equals $\sum_{q=0}^{r-1} x_{i,q} 2^q$, that is, $x_{i,q}$ has the same truth-value as $b_{j,q}$. To prohibit the existence of arithmetic progressions $a, a+d, \ldots, a+d(t_j-1)$, with $a \geqslant 1, d \geqslant 1, a+d(t_j-1) \leqslant n$ in block P_j, we add the following clauses:

$$\{u_{a,r-1}, \ldots, u_{a,0}, u_{a+d,r-1}, \ldots, u_{a+d,0}, \ldots, u_{a+d(t_j-1),r-1} \ldots, u_{a+d(t_j-1),0}\},$$

where literal $u_{i,q}$ (for $i \in \{a, a+d, \ldots, a+d(t_j-1)\}$) is defined as follows:

$$u_{i,q} = \begin{cases} \bar{x}_{i,q} & \text{if } b_{j,q} = 1, \\ x_{i,q} & \text{otherwise.} \end{cases}$$

The above clauses are added for each j in $\{0, 1, \ldots, k-1\}$. The clauses are long, but can be handled efficiently as described in Section 3 of [2].

To ensure that an integer i is not placed in a block P_j with $k \leqslant j \leqslant 2^r - 1$ (since there is no such block in the partition of $\{1, 2, \ldots, n\}$), we add the following clauses: $\{v_{i,r-1}, v_{i,r-2}, \ldots, v_{i,1}, v_{i,0}\}$ $(i = 1, 2, \ldots, n)$, where literal $v_{i,q}$ is defined as follows:

$$v_{i,q} = \begin{cases} \bar{x}_{i,q} & \text{if } b_{j,q} = 1, \\ x_{i,q} & \text{otherwise.} \end{cases}$$

The double-subscript variables $x_{p,q}$ with $1 \leqslant p \leqslant n$ and $0 \leqslant q \leqslant r-1$ can be converted to single-subcript variables $y_{(p-1)r+q+1}$. In the following example, we write j and $-j$ to mean the literals y_j and $\bar{y}_j$ respectively. An instance corresponding to $w(3; 2, 3, 3)$ for $n = 5$ can be constructed with 10 variables $1, 2, \ldots, 10$ and the following 23 clauses:

(*i*) `{1,2,3,4}, {1,2,5,6}, {1,2,7,8}, {1,2,9,10}, {3,4,5,6}, {3,4,7,8}, {3,4,9,10}, {5,6,7,8}, {5,6,9,10}, {7,8,9,10},`

(*ii*) `{1,-2,3,-4,5,-6}, {1,-2,5,-6,9,-10}, {3,-4,5,-6,7,-8}, {5,-6,7,-8,9,-10},`

(*iii*) `{-1,2,-3,4,-5,6}, {-1,2,-5,6,-9,10}, {-3,4,-5,6,-7,8}, {-5,6,-7,8,-9,10},`

(*iv*) `{-1,-2}, {-3,-4}, {-5,-6}, {-7,-8}, {-9,-10}.`

Clauses (*i*), (*ii*), and (*iii*) prohibit the existence of arithmetic progressions of lengths 2 (in block P_0), 3 (in block P_1), and 3 (in block P_2) respectively. Clauses (*iv*) prohibit the placement of any integer in block P_3.

3.2. A New Number: $w(3; 3, 3, 6) = 107$

It took 992 days of CPU-time (roughly 17 days of run-time) using 64 processors to prove that the instance corresponding to 107 is unsatisfiable, that is, there is no

good partition of the set $\{1, 2, \ldots, 107\}$. A good partition of the set $\{1, 2, \ldots, 106\}$ corresponding to $w(3; 3, 3, 6)$ is as follows:

```
22122112 22212222 00220021 22102021 22200220 22220202 11221222
22110122 02022002 20222122 12202210 11022220 22022221 22.
```

3.3. Some New Lower Bounds of $w(3; t_0, t_1, t_2)$

In this section, we provide the following new lower bounds of $w(3; t_0, t_1, t_2)$:

$w(3; 2, 4, 8) > 155$, $w(3; 3, 3, 7) > 149$, $w(3; 3, 3, 8) > 185$,
$w(3; 3, 3, 9) > 221$, $w(3; 3, 3, 10) > 265$, $w(3; 3, 4, 5) > 163$, and
$w(3; 3, 5, 5) > 243$.

3.4. An Observation

We observe that the more blocks in the partition, the better the performance of the encoding. Here, we present two previously unknown van der Waerden numbers $w(7; 2, 2, 2, 2, 2, 3, 5)$ and $w(8; 2, 2, 2, 2, 2, 2, 3, 4)$. For both these numbers, the encoding in [1, 8] takes more than a couple of months to prove the corresponding instances unsatisfiable.

$w(k; t_0, \ldots, t_{k-1})$	=	Example of a Good Partition	CPU-time
$w(7; 2, 2, 2, 2, 2, 3, 5)$	55	77776777 66767717 66766777 37777477 76267776 76676777 757767	21 days
$w(8; 2, 2, 2, 2, 2, 2, 3, 4)$	40	88787738 88787782 67888488 51778887 8778887	16 days

Table 1: Van der Waerden numbers with more partitions

4. Concluding Remarks

The encoding proposed in Section 3.1 may be useful (though may not be adequate) to determine some classical van der Waerden numbers like $w(3; 4, 4, 4)$ and $w(5; 3, 3, 3, 3, 3)$. Rabung [15] showed $w(3; 4, 4, 4) > 292$ by constructing an example of a good partition (following an observation of Berlekamp [4] and using power residues) of the set $\{1, 2, \ldots, 292\}$. Dransfield et al.[8] showed $w(5; 3, 3, 3, 3, 3) > 125$. Heule and van Maaren recently (2009) improved the bound to $w(5; 3, 3, 3, 3, 3) > 170$ [11]. They have used clauses to force symmetry (see Herwig et al. [10]) in addition to the usual clauses (as described in [1, 8]).

Though we do not have enough data points to generalize upper-bound conjectures given in Section 2.3, it might be possible that for $t \geqslant 3$ and fixed $s \geqslant 3$, there exists $c \geqslant 1$ such that $w(2; s, t) \leqslant ct^{s-1}$.

Acknowledgements The author is grateful to Vašek Chvátal and Clement Lam for their support. He would also like to thank the anonymous referee for his or her valuable suggestions, and Andalib Parvez for carefully reading the manuscript.

References

[1] T. Ahmed, Some new van der Waerden numbers and some van der Waerden-type numbers, *Integers*, **9** (2009), A06, 65–76.

[2] T. Ahmed, Two new van der Waerden numbers: $w(2;3,17)$ and $w(2;3,18)$, *Integers* **10** (2010), A32, 369–377.

[3] M. Beeler, P. O'Neil, Some new van der Waerden numbers, *Discrete Math.* **28** (1979), 135-146.

[4] E. R. Berlekamp, A construction for partitions which avoid long arithmetic progressions, *Canadian Math. Bull.* **11** (1968), 409–414.

[5] V. Chvátal, Some unknown van der Waerden numbers, *Combinatorial Structures and Their Applications (R.Guy et al.,eds.)*, 31–33, Gordon and Breach, New York, 1970.

[6] V. Chvátal, B. Reed, Mick Gets Some (The Odds Are on His Side), *Proceedings of the 33rd Annual Symposium on FOCS*, 1992, 620–627.

[7] M. Davis, G. Logemann, D. Loveland, A machine program for theorem-proving, *Comm. ACM*, **5** (1962), 394–397.

[8] M. R. Dransfield, L. Liu, V. Marek, M. Truszczyński, Satisfiability and Computing van der Waerden numbers, *The Electronic Journal of Combinatorics* **11(1)** (2004), R41.

[9] R. S. Garfinkel, G. L. Nemhauser, Integer Programming, *Wiley-Interscience (John Wiley & Sons): New York*, (1972), Series in Decision and Control.

[10] P. R. Herwig, M. J. H. Heule, P. M. van Lambalgen, H. van Maaren, A new method to construct lower bounds for van der Waerden numbers, *The Electronic Journal of Combinatorics* **14** (2007).

[11] SAT@Delft, van der Waerden numbers, `http://www.st.ewi.tudelft.nl/sat/waerden.php`.

[12] M. Kouril, J. L. Paul, The van der Waerden number W(2,6) is 1132, *Experimental Mathematics* **17(1)** (2008), 53–61.

[13] O. Kullmann, Green-Tao Numbers and SAT, *SAT 2010* (2010), 352–362.

[14] B. Landman, A. Robertson, C. Culver, Some new exact van der Waerden numbers, *Integers: Electronic J. Combinatorial Number Theory* **5(2)** (2005), A10.

[15] J. R. Rabung, Some progression-free partitions constructed using Folkman's method, *Canadian Mathematical Bulletin* **22** (1979) 87–91.

[16] R. Stevens, R. Shantaram, Computer-generated van der Waerden partitions, *Math. Computation* **32** (1978), 635–636.

[17] Dave A.D. Tompkins, Holger H. Hoos, UBCSAT: An implementation and experimentation environment for SLS algorithms for SAT and MAX-SAT. In *Holger H. Hoos and David G. Mitchell, editors, Theory and Applications of Satisfiability Testing of 2004*, Lecture Notes in Computer Science, Springer **3542** (2005), 306–320.

[18] B. L. van der Waerden, Beweis einer Baudetschen Vermutung, *Nieuw Archief voor Wiskunde* **15** (1927), 212–216.