Computer-Aided Proof of Erdős Discrepancy Properties

Boris Konev and Alexei Lisitsa

Liverpool University

Abstract

In 1930s Paul Erdős conjectured that for any positive integer C in any infinite ± 1 sequence (x_n) there exists a subsequence $x_d, x_{2d}, x_{3d}, \dots, x_{kd}$, for some positive integers k and d, such that $\mid \sum_{i=1}^{k} x_{i \cdot d} \mid > C$. The conjecture has been referred to as one of the major open problems in combinatorial number theory and discrepancy theory. For the particular case of $C = 1$ a human proof of the conjecture exists; for $C = 2$ a bespoke computer program had generated sequences of length 1124 of discrepancy 2, but the status of the conjecture remained open even for such a small bound. We show that by encoding the problem into Boolean satisfiability and applying the state of the art SAT solvers, one can obtain a discrepancy 2 sequence of length 1160 and a *proof* of the Erdős discrepancy conjecture for $C = 2$, claiming that no discrepancy 2 sequence of length 1161, or more, exists. In the similar way, we obtain a precise bound of 127 645 on the maximal lengths of both multiplicative and completely multiplicative sequences of discrepancy 3. We also demonstrate that unrestricted discrepancy 3 sequences can be longer than 130 000.

1 Introduction

The high mental activity of mathematical inquiry has been the battleground for artificial intelligence, and speaking more broadly for computer science, from their very dawn. The rigorous and structured nature of mathematical reasoning, the work on foundations of mathematics and on formalisation of mathematical proof, the invention of digital computers and the development of automated reasoning enabled mechanisation of proof search and proof checking. As a result, over the last 80 years we have witnessed the realisation of the dream of Leibniz and Babbage through the development of computer tools which can be used to make progress in mathematics itself by assisting humans in proving new statements and in formal verification of existing mathematical knowledge.

Yet, mathematically significant open questions whose status has been settled by a computer, that is, interesting mathematical problems for which no human solution or even solution sketch existed prior to the computer being applied, can be counted virtually on the fingers of one hand. Notably the success in these flagship cases was due to a combination of a significant computational effort with a non-trivial human effort either in the form of customising and fine-tuning proof assistants and their strategies [1] or in the form of developing specialised software programs [2, 3] that perform computations underpinning the proof.

Mathematicians generally have little or no issues when computers are used to discover a readable proof in a large space of possibilities. For example, even though it took the computer several weeks to find a solution to the Robbins problem [1], the proof itself consists of a dozen of steps, which can be inspected. On the other hand, when the answer is obtained by a complicated software system going through myriads of special cases—humans have doubts [4], and the degree of trust in computer mathematics often depends on the computer technology used.

Famously, it took four years to accept for publication the paper on the computer proof of Kepler's conjecture and still reviewers were not completely satisfied as they could not verify the entire computer program that was used to find the answer [5]. The paper on a computer proof of the non-existence of a finite projective plane of order 10 explicitly analyses the probability of a computational error [6]. The original Appel and Haken proof [2] of the Four Colour Theorem involved a deep theoretical argument followed by the vast computational case analysis carried out by a program written in the assembly language. The possibility of unaccounted errors in the software sparked a discussion whether such an answer can be accepted [7], which led to further progress and clarification by Robertson et al [8], who considerably simplified the theoretical part and used computer programs written in a high-level programming language. The culmination of the effort on verification and clarification of the Four Colour Theorem has been achieved in the recent work on computer-checked proof by Gonthier using the Coq proof assistant [9].

The Four Colour Theorem story reflects the general trends in computer mathematics to formalise computer-generated proofs in a formal inference system and verify them by a trusted computer program. The availability of a well-defined independently verifiable proof makes computer mathematics more palatable for mathematicians, even if the proof itself is still too large for any human ever to inspect [5, 9, 10, 11].

In this article we attack a discrepancy theory question, which stood open for more than 80 years, by reduction to Boolean satisfiability. We then apply a general purpose solver in a fully automated unguided manner. To find a solution, the solver goes through a very large number of possibilities; however, unlike a simple computer-aided enumeration of cases for proof by exhaustion, it also produces a *certificate* assuring that the computer did not make a mistake going through these possibilities. The certificate can be reliably and independently verified. Coupled with the fact that the computer program generating the input to the solver is short and clear, the certificate constitutes a rigorous *formal proof* of the statement.

Discrepancy theory is a branch of mathematics dealing with irregularities of distributions of points in some space in combinatorial, measure-theoretic and geometric settings [12, 13, 14, 15]. The paradigmatic combinatorial discrepancy theory problem can be described in terms of a hypergraph $\mathcal{H} = (U, S)$, that is, a set U and a family of its subsets $S \subseteq 2^U$. Consider a colouring $c : U \to \{+1, -1\}$ of the elements of U in *blue* $(+1)$ and *red* (-1) colours. Then one may ask whether there exists a colouring of the elements of U such that colours are distributed uniformly in every element of S or a discrepancy of colours is always inevitable. Formally, the discrepancy (deviation from a uniform distribution) of a hypergraph $\mathcal{H}$ is defined as $\min_c(\max_{s \in S} |\sum_{e \in s} c(e)|)$. Discrepancy theory has found applications in computational complexity [13], complexity of communication [16] and differential privacy [17]. Although other forms of combinatorial discrepancy, for example, *hereditary* and *multicolour* discrepancies have been defined, the classical two-coloured discrepancy defined above remains a focus of research both from the mathematical and algorithmic viewpoints [18].

One of the oldest problems of combinatorial discrepancy theory is the discrepancy of hypergraphs over sets of natural numbers with the subsets (hyperedges) forming arithmetical progressions over these sets [19]. Roth's theorem [20], one of the main results in the area, states that for the hypergraph formed by the arithmetic progressions in $\{1, \ldots, n\}$, that is $\mathcal{H}_n = (U_n, S_n)$, where $U_n = \{1, 2, \ldots, n\}$ and elements of S_n being of the form $(d \cdot i + c)$ for arbitrary d, c, the discrepancy grows at least as $\frac{1}{20}n^{1/4}$. While proving this result, Roth introduced a pioneering proof technique that played a key role in the development of discrepancy theory and other areas [21].

Surprisingly, for the more restricted case of *homogeneous* arithmetic progressions of the form $(d \cdot i)$, the question of the discrepancy bounds has been open for more than eighty years. In 1930s Erdős conjectured [22] that discrepancy is unbounded. Independently the same conjecture has been made by Čudakov [23]. Proving or disproving this conjecture has become one of the major open problems in combinatorial number theory and discrepancy theory. It is often referred to as the *Erdős discrepancy problem* (EDP) [12, 15, 24].

The expected value of the discrepancy of random ± 1 sequences of length n grows as $n^{1/2+o(1)}$ and the explicit constructions of a sequence with slowly growing discrepancy at the rate of $\log_3 n$ have been demonstrated [25, 26]. By considering cases, one can see that any ± 1 sequence containing 12 or more elements has discrepancy at least 2; that is, Erdős's conjecture holds for the particular case $C = 1$ (also implied by a stronger result of Mathias [27]). Until February 2014 the status of the conjecture remained unknown for all other values of C. Although widely believed not to be the case, there was still a possibility that an infinite sequence of discrepancy 2 existed.

The EDP has attracted renewed interest in 2009-2010 as it became a topic of the fifth Polymath project [28] a widely publicised endeavour in collective mathematics initiated by Gowers [29]. As part of this activity an attempt has been made to attack the problem using computers (see the discussion in [28]). A purposely written computer program had successfully found ± 1 sequences of length 1124 and discrepancy 2; however, no further progress has been made leading to a claim "given how long a finite sequence can be, it seems unlikely that we could answer this question just by a clever search of all possibilities on a computer" [28].

The status of the Erdős discrepancy conjecture for $C = 2$ has been settled by the authors of this article [30, 31] by encoding the problem as a propositional satisfiability problem and using state of the art SAT solvers to prove that the longest ± 1 sequence of discrepancy 2 contains 1160 elements. A 13 900 long ± 1 sequence of discrepancy 3 was also constructed.

This article is a revised and extended version of [31]. We use a different smaller

SAT encoding of the Erdős discrepancy problem, which is based on the sequential counter encoding of the *at-most* cardinality constraints[1]. The impact of the new encoding is twofold. Firstly, it allows us to significantly reduce the size of the machine-generated proof of the fact that any sequence longer than 1160 has discrepancy at least 3. Secondly, by combining the new encoding with additional restrictions that the sequence is multiplicative, or completely multiplicative[2], we improve significantly the lower bound on the length of sequences of discrepancy 3. We prove the surprising result that $127\,645$, the length of the longest completely multiplicative sequence of discrepancy 3, is also the maximal length of a multiplicative sequence of discrepancy 3, which is not the case for $C = 1$ and $C = 2$. The article also contains detailed argumentation, examples and complete proofs.

The article is organised as follows. In Section 2 we introduce the main terms and definition. In Section 3 we describe the new SAT encoding of the Erdős discrepancy problem. Results and conclusions are discussed in Sections 4 and 5 respectively.

2 Preliminaries

We divide this section into two parts: main definitions for the Erdős discrepancy problem and some background and definitions for SAT solving. Since number 1 is used both as an element of ± 1 sequences and as the logical value *true*, to avoid confusion, in what follows we write 1 to refer to the logical value *true* and $+1$ to refer to elements of ± 1 sequences. We also use the following naming convention: we write $x_1, \dots x_n$ for ± 1 sequences, $p_1, \dots, p_n$ for sequences of propositions, and $a_1, \dots, a_n$ for $0/1$ sequences.

2.1 Discrepancy of ± 1 Sequences

A ± 1 sequence of length n is a function $\{1, \dots, n\} \to \{-1, +1\}$. An infinite ± 1 sequence is a function $\mathbb{N}^+ \to \{1, -1\}$, where $\mathbb{N}^+$ is the set of positive natural numbers. We write $x_1, \dots, x_n$ to denote a finite ± 1 sequence of length n, and (x_n) to denote an infinite sequence. We refer to the i-th element of a sequence x, that is the value of $x(i)$, as x_i. A (finite or infinite) ± 1 sequence x is *completely multiplicative* [33] if

$$x_{m \cdot n} = x_m \cdot x_n, \text{ for all } m, n \in \mathbb{N}^+. \tag{1}$$

The sequence is *multiplicative* if (1) is only required for coprime m and n.

It is easy to see that a sequence x is completely multiplicative if, and only if, $x_1 = +1$ and for the canonical representation $m = \prod_{i=1}^{k} {p_i}^{\alpha_i}$, where $p_1 < p_2 < \cdots < p_k$ are primes and $\alpha_i \in \mathbb{N}^+$, we have $x_m = \prod_{i=1}^{k} (x_{p_i})^{\alpha_i}$. This observation leads to a more computationally friendly definition of completely multiplicative sequences: x is completely multiplicative if, and only if,

$$x_1 = +1 \text{ and for every composite } m \text{ we have } x_m = x_i \cdot x_j, \text{ for } \textit{some } i \leq j, \text{ non-trivial divisors of } m. \tag{2}$$

[1]We are grateful to Donald E. Knuth for pointing us in that direction.

[2]After this article had been submitted, it came to our notice that for completely multiplicative sequences this result has independently been obtained by Le Bras, Gomes and Selman [32].

The EDP can be naturally described in terms of ± 1 sequences (and this is how Erdős himself introduced it [22]). Then Erdős's conjecture can be formulated as follows.

Conjecture (Erdős, 1930s)**.** *For any $C > 0$ and any infinite ± 1 sequence (x_n) there exists its subsequence $x_d, x_{2d}, x_{3d}, \dots, x_{kd}$, for some positive integers k and d, such that $|\sum_{i=1}^{k} x_{i\cdot d}\,| > C$.*

Notice that the general definition of discrepancy given in the introduction can be specialised in terms of partial sums of subsequences of a ± 1 sequence. To simplify notation, we introduce an auxiliary notion of C-boundedness. We say that a ± 1 sequence $x_1, \dots, x_l$ is *C-bounded*, for some $C > 0$ if $|\sum_{i=1}^{j} x_i| \leq C$, for all $j : 0 < j \leq l$. Notice that every ± 1 sequence whose length does not exceed C is always C-bounded.

Then the discrepancy of a finite ± 1 sequence $x_1, \dots, x_n$ of length n is a *minimal* C such that for every $d : 1 \leq d \leq \lfloor \frac{n}{C+1} \rfloor$ the subsequence $x_d, x_{2d} \dots, x_{\lfloor n/d \rfloor \cdot d}$ is C-bounded. For an infinite sequence (x_n) its discrepancy is the supremum of discrepancies of all its initial finite fragments.

The Erdős discrepancy conjecture is equivalent to its variant where "infinite ± 1 sequence" is replaced by "infinite completely multiplicative ± 1 sequence" [22]. This observation, in particular, explains an interest to multiplicativity properties in this context.

Example 1. *The finite sequence $x_1, \dots x_5 = -1, -1, -1, +1, +1$ has discrepancy 3 as for $d = 1$ and $j = 3$ we have $|\sum_{i=1}^{j} x_{i\cdot d}| = |\sum_{i=1}^{3} x_i| = |-3| = 3$, and it can be readily checked that all other sequences in the definition of discrepancy are 3-bounded.*

The finite sequence $y_1, \dots y_6 = -1, +1, -1, +1, -1, +1$ also has discrepancy 3, a value achieved by setting $d = 2$ and $j = 3$ in the definition above.

The infinite sequence $(x_n) = (-1^n)$ has an unbounded discrepancy as its initial finite segment $x_1, \dots, x_m$, for $m \geq 2$, contains a subsequence whose all members are $+1$, namely $x_2, x_4, \dots x_l$, where $l = 2\lfloor \frac{m}{2} \rfloor$. Thus, the discrepancy of the initial segment $x_1, \dots, x_m$ grows as $\lfloor \frac{m}{2} \rfloor$. In fact, any infinite periodic ± 1 sequence (x_n) has an unbounded discrepancy. Indeed if $x_{n+p} = x_n$, for all values of n, then $x_p = x_{2p} = x_{3p} = \dots$, so $|x_p + x_{2p} + \dots + x_{jp}| = j$, for any $j > 0$ and thus there is no C such that every sequence $x_p, x_{2p}, \dots, x_{jp}$ is C-bounded for all $j > 0$. A similar argument applies to eventually periodic sequences, so no eventually periodic sequence has a bounded discrepancy.

It is easy to see why any ± 1 sequence containing 12 elements has discrepancy at least 2.

Example 2. *For the proof by contradiction, suppose that the discrepancy of some ± 1 sequence $x_1, \dots, x_{12}$ is 1. Assume that x_1 is $+1$. We write*

$$(+1, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_)$$

to track progress in this example, that is, we put specific values $+1$ or -1 into positions $i : 1 \leq i \leq 12$, to indicate decisions on x_i which have been taken so far, and mark positions of x_i, for which no decision has been made by an underscore.

Notice that x_2 must be -1 for otherwise $x_1 + x_2 = 2$. So, we progress to

$$(+1, -1, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_).$$

Then the 4th element of the sequence must be $+1$ for otherwise for $d = 2$ the sum $x_d + x_{2d} = x_2 + x_4 = -2$. So we progress to

$$(+1, -1, \_, +1, \_, \_, \_, \_, \_, \_, \_, \_).$$

Then the 3rd element of the sequence must be -1 for otherwise $x_1 + \cdots + x_4 = 2$ and so we come to

$$(+1, -1, -1, +1, \_, \_, \_, \_, \_, \_, \_, \_).$$

Repeating the reasoning above for x_3 and x_6 followed by x_5, for x_4 and x_8 followed by x_7, for x_5 and x_{10} followed by x_9 and finally for x_6 and x_{12} followed by x_{11} we progress to

$$(+1, -1, -1, +1, -1, +1, +1, -1, -1, +1, +1, -1). \qquad (3)$$

But then for $d = 3$ we have $x_d + x_{2d} + x_{3d} + x_{4d} = x_3 + x_6 + x_9 + x_{12} = -2$. So we derive a contradiction. It can be checked in a similar way that the other possibility of x_1 being -1 also leads to a contradiction.

The first eleven elements of the sequence (3) form a discrepancy 1 sequence. It is multiplicative but not completely multiplicative as x_9 is -1. Reasoning similar to the one above shows that there exists a unique longest completely multiplicative ± 1 sequence of discrepancy 1 which has nine elements:

$$(+1, -1, -1, +1, -1, +1, +1, -1, +1).$$

2.2 Propositional Satisfiability Problem

We assume standard definitions for propositional logic (see, for example, [34]). Propositional formulae are defined over Boolean constants *true* and *false*, denoted by 1 and 0, respectively, and the set of Boolean variables (or propositions) PV as follows: Boolean constants 0 and 1 as well as the elements of PV are formulae; if Φ and Ψ are formulae then so are $\Phi \wedge \Psi$ (conjunction), $\Phi \vee \Psi$ (disjunction), $\Phi \rightarrow \Psi$ (implication), $\Phi \leftrightarrow \Psi$ (equivalence) and $\neg\Phi$ (negation). We typically use letters p, q and s to denote propositions and capital Greek letters Φ and Ψ to denote propositional formulae. Whenever necessary, subscripts and superscripts are used. We use $\mathrm{vars}(\Phi)$ to denote the set of all propositions occurring in the formula Φ.

Every propositional formula can be reduced to conjunctive normal form. Propositions and negations of propositions are called *literals*. When the negation is applied to a literal, double negations are implicitly removed, that is, if l is $\neg p$ then $\neg l$ is p. A disjunction of literals is called a *clause*. A clause containing exactly one literal is called a unit clause. A conjunction of clauses is called a propositional formula in *conjunctive normal form*, a CNF formula for short. A clause can be represented by the set of its literals and the empty clause correspond to 0 (*false*); a CNF formula can be represented by the set its clauses. The two representations are used interchangeably. We typically use meaningful terms typeset in sans serif font, for example edp or cmult, to highlight the fact that a propositional formula is a CNF formula of interest.

For a propositional formula Φ, we write $\Phi(p_1, \dots, p_n)$ to indicate that $\{p_1, \dots, p_n\} \subseteq \mathrm{vars}(\Phi)$. Propositions $p_1, \dots, p_n$ are designated as 'input' propositions in this case, and the intended meaning is that formula Φ encodes some property of $p_1, \dots, p_n$. Then the expression $\Phi(q_1, \dots, q_n)$ denotes the result of simultaneous replacement of every occurrence of p_i in Φ with q_i, for $1 \leq i \leq n$.

The semantics of propositional formulae is given by interpretations (also termed assignments). An interpretation I is a mapping $PV \to \{0, 1\}$ extended to literals, clauses, CNF formulae and propositional formulae in general in the usual way. For an assignment I and a formula Φ we say that I satisfies Φ (or I is a model of Φ) if $I(\Phi) = 1$. A formula Φ is satisfiable if there exists an assignment that satisfies it, and unsatisfiable otherwise.

Despite the NP-completeness of the satisfiability problem, the tremendous progress in recent years in the development of SAT solvers—computer programs capable to find a satisfying assignment for a given propositional formula—made it possible to solve many interesting hard problems by first expressing them as a propositional formula and then using a SAT solver for obtaining a solution [35]. In addition to returning a satisfying assignment if the input formula is satisfiable, some SAT solvers are also capable to return a proof (or certificate) of unsatisfiability.

Reverse Unit Propagation (RUP) proofs constitute a compact representation of the resolution refutation of the given formula [36] in the following sense. *Unit propagation* is a CNF formula transformation technique, which simplifies the formula by fixing the values of propositions occurring to its unit clauses to satisfy these clauses. That is, if the unit clause (p) occurs in the CNF formula then all occurrences of p are replaced by 1 and if the unit clause $(\neg p)$ occurs in the CNF formula, all occurrences of p are replaced by 0. Then the CNF formula is simplified in the obvious way. A clause $C = (l_1, \dots l_m)$ is a RUP inference from the input CNF formula Ψ if adding the unit clauses $(\neg l_1), \dots, (\neg l_m)$ to Ψ makes the whole formula refutable by unit propagation. A RUP unsatisfiability certificate is the sequence of clauses $C_1, \dots C_m$ such that for every $1 \leq i \leq m$ the clause C_i is a RUP inference from $\Psi \cup \{C_1, \dots, C_{i-1}\}$ and C_m is the empty clause. Every unsatisfiable CNF formula has a RUP unsatisfiability certificate [36].

Delete Reverse Unit Propagation (DRUP) proofs extend RUP proofs by including extra information about the proof search process, namely clauses that have been discarded by the solver. Eliminating this extra information from a DRUP proof converts it to a valid RUP proof. DRUP proofs are somewhat longer but they are significantly faster to verify than RUP proofs [37].

3 SAT Encoding of the discrepancy problem

In this section we present our SAT encoding of the EDP. We start with characterising C-boundedness of a sequence with the help of cardinality constraints. We then represent these constraints as a propositional formula in Section 3.2. In Section 3.3 we present an optimised clausal form for this encoding. In Section 3.4 we discuss a SAT encoding of multiplicativity. Finally, in Section 3.5 we put all the parts together.

3.1 C-Boundedness Expressed as Cardinality Constraints

In the context of propositional satisfiability, cardinality constraints [38] are expressions that impose restrictions on propositional interpretations by specifying numerical bounds on the number of propositions, from a fixed set of propositions, that can be assigned value 1. The *at-most* r constraint over the set of propositions $\{p_1, \dots, p_n\}$, written as $p_1 + \dots + p_n \leq r$, holds for an interpretation I if, and only if, at most r propositions among $p_1, \dots, p_n$ are true under I. Its counterpart, the *at-least* r constraint, written as $p_1 + \dots + p_n \geq r$, holds for an interpretation I if, and only if, at least r propositions among $p_1, \dots, p_n$ are true under I. A constraint is satisfiable if there exists an assignment in which the constraint holds. Cardinality constraints can be encoded by propositional formulae so that every interpretation satisfying the formula satisfies the cardinality constraint and vice versa; a number of such encodings can be found in the literature [38].

As every ± 1 sequence whose length does not exceed C is always C-bounded, in what follows we only consider cases of $l > C$. As a first step towards a SAT encoding of C-boundedness, we switch our consideration from finite ± 1 sequences of the form $x_1, \dots, x_l$ to their 'characteristic' representation by $0/1$ sequences of the form $a_1, \dots, a_l$ so that $x_i = 2a_i - 1$ (in other words, $+1$ corresponds to 1 and -1 corresponds to 0). We extend the definition of C-boundedness to $0/1$ sequences in the obvious way: a $0/1$ sequence $a_1, a_2, \dots, a_l$ is C-bounded if, and only if, the ± 1 sequence $2a_1 - 1, 2a_2 - 1, \dots, 2a_l - 1$ is C-bounded.

Notice that $|\sum_{i=1}^{j} x_i|$ from the definition of C-boundedness above can be characterised as the absolute value of the difference of the number of occurrences of $+1$ in $x_1, \dots, x_j$ and the number of occurrences of -1 in $x_1, \dots, x_j$. Under our correspondence, the number of occurrences of $+1$ in $x_1, \dots, x_j$ is the number of occurrences of 1 in $a_1, \dots, a_j$ is $\sum_{i=1}^{j} a_i$. As the total number of elements in $x_1, \dots x_j$ is j, the number of occurrences of -1 in $x_1, \dots, x_j$ is the number of occurrences of 0 in $a_1, \dots, a_j$ is $(j - \sum_{i=1}^{j} a_i)$. Then

$$\Big|\sum_{i=1}^{j} x_i\Big| = \Big|\sum_{i=1}^{j} a_i - \big(j - \sum_{i=1}^{j} a_i\big)\Big| = \Big|2\sum_{i=1}^{j} a_i - j\Big|.$$

Thus $a_1, \dots, a_l$ is C-bounded if, and only if,

$$-C \leq 2\sum_{i=1}^{j} a_i - j \leq C, \text{ for all } j : C < j \leq l,$$

which is equivalent to the two following systems of inequalities,

$$\sum_{i=1}^{j} a_i \leq \left\lfloor \frac{C+j}{2} \right\rfloor, \quad \text{for all } j : C < j \leq l \tag{4}$$

and

$$\sum_{i=1}^{j} a_i \geq \left\lceil \frac{-C+j}{2} \right\rceil, \quad \text{for all } \; j : C < j \leq l. \tag{5}$$

If we interpret now the numerical values 0 and 1 as Boolean constants *true* and *false*, respectively, the $0/1$ sequence $a_1, \ldots, a_l$ corresponds to the evaluation of a sequence of Boolean propositions $p_1, \ldots, p_l$ under some interpretation I, and C-boundedness of a sequence can be expressed as cardinality constraints in a natural way.

Theorem 3. *A* ± 1 *sequence* $x_1, \ldots, x_l$*, for some* $C > 0$ *and* $l > C$*, is* C*-bounded if, and only if, the union of the* at-most *cardinality constraints*

$$p_1 + \cdots + p_j \leq \left\lfloor \frac{C+j}{2} \right\rfloor, \textit{ for all } j : C < j \leq l, \tag{6}$$

and the at-least *cardinality constraints*

$$p_1 + \cdots + p_j \geq \left\lceil \frac{-C+j}{2} \right\rceil, \textit{ for all } j : C < j \leq l. \tag{7}$$

is satisfiable.

Proof. The proof consists of a trivial observation that, by definition of cardinality constraints, the *at-most* constraints (6) capture condition (4) while the *at-least* constraints (7) capture condition (5). □

Theorem 3 immediately yields a reduction of the question of the existence of a C-bounded sequence to a propositional satisfiability problem using any SAT representation of cardinality constraints [38]. However, if used in a "black-box" manner, constraints (6) and (7) are to be considered independently of each other for every $j : C < j \leq l$ leading to a proliferation of their SAT encodings. Instead we use the SAT encoding of cardinality constraints based on sequential counter circuits, which allows us to express C-boundedness of a sequence with a single propositional formula.

3.2 Sequential Counter-Based SAT Encoding of C-boundedness

A SAT encoding of cardinality constraints based on sequential counter circuits has been suggested by Sinz [39]. This encoding introduces auxiliary propositions s_j^k to represent unary counters storing the partial sums of prefixes of $p_1, \ldots, p_l$ so that whenever $\sum_{i=1}^{j} p_i \geq k$, for some $j \leq l$, we have $s_j^k = 1$.

We slightly modify the encoding in [39] as follows. Let $\Phi(p_1, \ldots, p_l)$ be the conjunction of

$$s_j^k \leftrightarrow (s_{j-1}^k \vee (s_{j-1}^{k-1} \wedge p_j)), \qquad \text{for all } 1 \leq k \leq l, 1 \leq j \leq l; \tag{8}$$

$$(\neg s_j^k), \qquad \text{for all } 0 \leq j < k \leq l; \tag{9}$$

$$(s_j^k), \qquad \text{for } k = 0 \text{ and all } 0 \leq j \leq l. \tag{10}$$

Recall that we write $\Phi(p_1, \dots, p_l)$ to highlight the fact that $p_1, \dots, p_l$ are designated 'input' propositions; the set of all propositions of $\Phi(p_1, \dots, p_l)$ is $\mathrm{vars}(\Phi(p_1, \dots, p_l)) = \{p_1, \dots, p_l\} \cup \bigcup_{k=1}^{l} \bigcup_{j=1}^{l} \{s_j^k\}$.

The proof of the fact that in every model I of $\Phi(p_1, \dots, p_l)$ we have $I(s_j^k) = 1$ if, and only if, $\sum_{i=1}^{j} I(p_i) \geq k$ can be extracted from [39]. It is based on the observation that the sum of the first j elements of the $0/1$ sequence $I(p_1), \dots, I(p_l)$ exceeds k if, and only if, either already the sum of the first $j-1$ elements exceeds k, or the sum of the first $j-1$ elements is k and the j-th element of the sequence is 1. Formulae (9) and (10) specify the border cases that the sum of the first j elements of any sequence cannot exceed j and that the sum of every initial subsequence is at least 0.

Notice that rather than include formulae (9) and (10) explicitly in the encoding, one can directly modify (8) by replacing all occurrences of s_j^k, for $0 \leq j < k \leq l$, with 0 (the truth value *false*) and all occurrences of s_j^k, for $k = 0$ and all $0 \leq j \leq l$, with 1 (the truth value *true*). Then, for example, for $k = j = 1$ formula (8) simplifies to $s_1^1 \leftrightarrow p_1$. We write (9) and (10) explicitly for the exposition purposes.

We give the formal proof of the following proposition in A for completeness of the presentation.

Proposition 4. *Let $\Phi(p_1, \dots, p_l)$ be as defined above. Then*

(i) For any assignment $I : \mathrm{vars}(\Phi(p_1, \dots, p_l)) \to \{0, 1\}$ such that I satisfies $\Phi(p_1, \dots, p_l)$, any $1 \leq j \leq l$ and $1 \leq k \leq l$ we have

$$I(s_j^k) = 1 \quad \textit{if, and only if,} \qquad \sum_{i=1}^{j} I(p_i) \geq k.$$

(ii) For any $0/1$-sequence $(a_1, \dots, a_l) \in \{0, 1\}^l$ there exists an assignment $I : \mathrm{vars}(\Phi(p_1, \dots, p_l)) \to \{0, 1\}$ such that I satisfies $\Phi(p_1, \dots, p_l)$ and $I(p_i) = a_i$, for all $1 \leq i \leq l$.

It follows from Proposition 4 that the formula $(\Phi(p_1, \dots, p_l) \wedge \neg s_l^{r+1})$ encodes the *at-most* r cardinality constraint $p_1 + \cdots + p_l \leq r$, while the formula $(\Phi(p_1, \dots, p_l) \wedge s_l^r)$ encodes the *at-least* r cardinality constraint $p_1 + \cdots + p_l \geq r$. Notice that the original encoding in [39] only captures the *at-most* constraint and uses the polarity-based optimisation based on Tseitin's [40] renaming techniques yielding $O(nr)$ clauses, which require $O(nr)$ auxiliary propositions.

We now use Proposition 4 to encode C-boundedness of sequences. Let propositional formula $\Psi^C(p_1, \dots, p_l)$ be the conjunction of $\Phi(p_1, \dots, p_l)$, with

$$(\neg s_j^{r+1}), \text{for all } j : C < j \leq l \text{ and } r = \left\lfloor \frac{C + j}{2} \right\rfloor, \tag{11}$$

and

$$(s_j^r), \text{for all } j : C < j \leq l \text{ and } r = \left\lceil \frac{-C + j}{2} \right\rceil. \tag{12}$$

Notice that, as in the case of $\Phi(p_1, \dots, p_l)$, we write (11) and (12) explicitly for the ease of explanation. Then we have the following.

Theorem 5. *For any $C > 0$, any $l > C$ and any assignment $I : \{p_1, \ldots, p_l\} \to \{0, 1\}$ the following holds: there exists an extension of I to $I' : \mathrm{vars}(\Psi^C(p_1, \ldots, p_l)) \to \{0, 1\}$ that is a model of $\Psi^C(p_1, \ldots, p_l)$ if, and only if, the sequence $I(p_1), \ldots, I(p_l)$ is C-bounded.*

Proof. Assume that for some assignment $I : \{p_1, \ldots, p_l\} \to \{0, 1\}$ the sequence $I(p_1), \ldots, I(p_l)$ is C-bounded. By item (*ii*) of Proposition 4, I can be extended to an assignment $I' : \mathrm{vars}(\Phi(p_1, \ldots, p_l) \to \{0, 1\}$ such that I' satisfies $\Phi(p_1, \ldots, p_l)$ and $I'(p_i) = I(p_i)$, for all $1 \leq i \leq l$. As I' is an extension of I the sequence $a_1, \ldots, a_l$, where $a_i = I'(p_1)$ for $i = 1, \ldots, l$, also is C-bounded, so conditions (4) and (5) hold true. By item (*i*) of Proposition 4, $I'(s_j^{r+1}) = 0$, for all $j : C < j \leq l$ and $r = \left\lfloor \frac{C+j}{2} \right\rfloor$, and $I'(s_j^r) = 1$, for all $j : C < j \leq l$ and $r = \left\lceil \frac{-C+j}{2} \right\rceil$. Thus, I' satisfies $\Psi^C(p_1, \ldots, p_l)$.

Conversely, consider an assignment $I : \{p_1, \ldots, p_l\} \to \{0, 1\}$ and assume that its extension to $I' : \mathrm{vars}(\Psi^C(p_1, \ldots, p_l)) \to \{0, 1\}$ is a model of $\Psi^C(p_1, \ldots, p_l)$. Since I' satisfies $\Psi^C(p_1, \ldots, p_l)$, we have $I'(s_j^{r+1}) = 0$, for all $j : C < j \leq l$ and $r = \left\lfloor \frac{C+j}{2} \right\rfloor$, and $I'(s_j^r) = 1$, for all $j : C < j \leq l$ and $r = \left\lceil \frac{-C+j}{2} \right\rceil$. As $\Phi(p_1, \ldots, p_l)$ is a conjunct of $\Psi^C(p_1, \ldots, p_l)$, by Proposition 4 item (*i*), $\sum_{i=1}^{j} I'(p_i) \leq \lfloor \frac{C+j}{2} \rfloor$, for all $j : C < j \leq l$, and $\sum_{i=1}^{j} I'(p_i) \geq \lfloor \frac{-C+j}{2} \rfloor$, for all $j : C < j \leq l$, so $I'(p_1), \ldots, I'(p_l)$ is C-bounded.

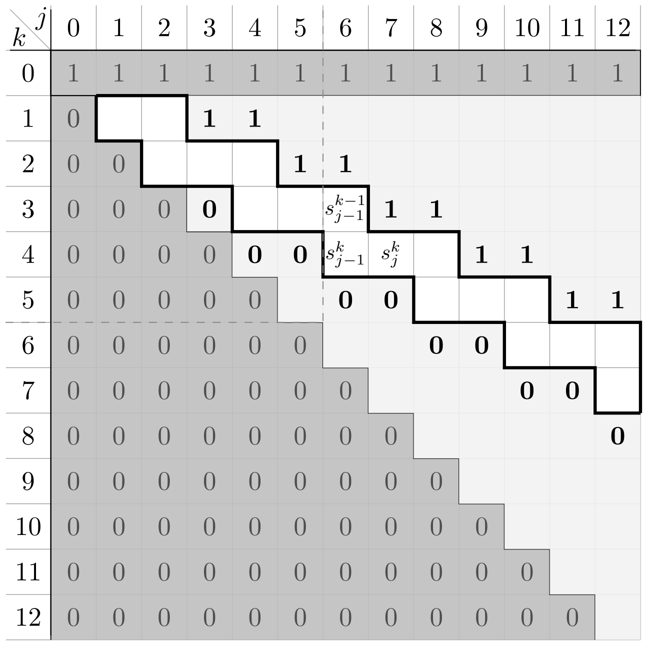
□

3.3 Clausal form

Straightforward clausification of the formula $\Psi^C(p_1, \ldots, p_l)$ yields $O(l^2)$ clauses; however, unit propagation of (9), (10), (11) and (12) into (8) reduces significantly the size of the resulting CNF.

Example 6. *We illustrate the effect of unit propagation on $\Psi^C(p_1, \ldots, p_l)$ in Figure 1 by presenting graphically values of s_j^k for $\{k, j\} \subseteq \{0, \ldots, l\}$ for the case of $C = 2$ and a relatively large $l = 12$. Notice that in every interpretation I satisfying $\Psi^C(p_1, \ldots, p_l)$, the unit clauses (9), (10), (11) and (12) are true so the truth values of auxiliary propositions s_j^k occurring in these clauses are fixed in any such interpretation. The dark gray areas in Figure 1 correspond to s_j^k whose values are being fixed by (9) and (10), while the boldface zeros and ones on the light gray background are due to (11) and (12). It is not hard to see that once these values are fixed, due to (8) all light gray cells above the unshaded region should contain 1 and all light gray cells below the unshaded region should contain 0. Thus, it is only the values of s_j^k in the unshaded region that are not uniquely determined in any interpretation satisfying $\Psi^3(p_1, \ldots, p_{12})$. In our example there are only 18 indeterminate values of s_j^k out of $13 \times 13 = 169$ combinations of j, k.*

The same reasoning as in the example above applies to other values of C and l.



$k \backslash j$	0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0			**1**	**1**								
2	0	0				**1**	**1**						
3	0	0	0	**0**			s_{j-1}^{k-1}	**1**	**1**				
4	0	0	0	0	**0**	**0**	s_{j-1}^{k}	s_j^k		**1**	**1**		
5	0	0	0	0	0		**0**	**0**				**1**	**1**
6	0	0	0	0	0	0			**0**	**0**			
7	0	0	0	0	0	0	0				**0**	**0**	
8	0	0	0	0	0	0	0	0					**0**
9	0	0	0	0	0	0	0	0	0				
10	0	0	0	0	0	0	0	0	0	0			
11	0	0	0	0	0	0	0	0	0	0	0		
12	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 1: Auxiliary propositions s_j^k in $\Psi^C(p_1, \ldots, p_l)$ for $C = 2$ and $l = 12$.

Proposition 7. *Let $C > 0$ and $l > C$. Then*

$$(s_j^k), \quad \textit{for } 1 \le k \le l \textit{ and } 2k - 1 + C \le j \le l \tag{13}$$

and

$$(\neg s_j^k), \quad \textit{for } 1 \le k \le l \textit{ and } 0 \le j \le \max\{2k - C - 1, l\} \tag{14}$$

are logical consequences of $\Psi^C(p_1, \ldots, p_l)$.

Proof. Suppose, to the contrary, that for some $C > 0$, some $l > C$, some I satisfying $\Psi^C(p_1, \ldots, p_l)$, some $k : 1 \le k \le l$ and some $j : 2k - 1 + C \le j \le l$ we have $I(s_j^k) = 0$. Then by Proposition 4, $\sum_{i=1}^{j} I(p_i) < k$. Since $j \ge 2k - 1 + C$ and $I(p_1), \ldots I(p_l)$ is a $0/1$ sequence,

$$\sum_{i=1}^{j} I(p_i) \ge \sum_{i=1}^{2k-1+C} I(p_i).$$

Thus,

$$\sum_{i=1}^{2k-1+C} I(p_i) < k.$$

On the other hand, as I satisfies $\Psi^C(p_1, \dots, p_l)$, the sequence $I(p_1), \dots, I(p_l)$ is C-bounded and, as $2k - 1 + C > C$, by (5)

$$\sum_{i=1}^{2k-1+C} I(p_i) \geq \left\lceil \frac{-C + 2k - 1 + C}{2} \right\rceil = \left\lceil \frac{2k-1}{2} \right\rceil = k.$$

So we derive a contradiction.

The proof of (14) is similar with the use of (5). □

As (8) is logically equivalent to the set of clauses

$$(\neg s_j^k \vee s_{j-1}^k \vee s_{j-1}^{k-1}) \tag{15}$$

$$(\neg s_j^k \vee s_{j-1}^k \vee p_j) \tag{16}$$

$$(\neg s_{j-1}^k \vee s_j^k) \tag{17}$$

$$(\neg s_{j-1}^{k-1} \vee \neg p_j \vee s_j^k), \tag{18}$$

where $1 \leq k \leq l$, $1 \leq j \leq l$, it can be seen that the formula $\Psi^C(p_1, \dots, p_l)$ is logically equivalent to the set of clauses S consisting of (15)–(18), (9), (10), (13) and (14). Let $\mathsf{CBound}^C(p_1, \dots, p_l)$ be the result of applying unit propagation to S in an exhaustive manner. One can see that the set $\mathsf{CBound}^C(p_1, \dots, p_l)$ contains less than $C \cdot l$ auxiliary propositions and less than $4C \cdot l$ clauses.

Example 8. *To save space, we limit our consideration to the top-left* 5×5 *region in Figure 1 and present* $\mathsf{CBound}^2(p_1, \dots, p_5)$*, a clausal representation of the statement that the sequence* $p_1, \dots, p_5$ *is* 2*-bounded. We also demonstrate how clauses (9), (10), (13) and (14) unit propagate into clauses (15)–(18).*

Notice that for $k = 1$ *every instance of clause (15) contains literal* s_{j-1}^0*, the only literal of the unit clause (10). Thus, every instance of clause (15) for* $k = 1$ *is redundant.*

For $k = 2$ *and* $j = 1$*, clause (15) contains* $\neg s_1^2$*, the only literal of the unit clause (9), so for* $k = 2$ *and* $j = 1$*, the instance of clause (15) is also redundant.*

For $k = 2$ *and* $j = 2$*, an instance of the unit clause (9), namely* $\neg s_1^2$*, unit propagates into (15) resulting in a 2-CNF clause* $(\neg s_2^2 \vee s_1^1)$*.*

For $k = 2$ *and* $j = 3$*, (15) instantiates to* $(\neg s_3^2 \vee s_2^2 \vee s_2^1)$*.*

Finally, for $k = 2$ *and for* $4 \leq j \leq 5$*, clause (15) contains* s_{j-1}^1*, the only literal of the unit clause (13). Thus, for* $k = 2$ *and* $4 \leq j \leq 5$ *the instances of clause (15) are redundant.*

By a further consideration of cases, one can see that the set of all non-redundant simplified instance of clause (15), for $1 \leq k \leq 5$ *and* $1 \leq j \leq 5$*, consists of*

$$\begin{array}{ll} (\neg s_2^2 \vee s_1^1) & (\neg s_4^3 \vee s_3^2) \\ (\neg s_3^2 \vee s_2^2 \vee s_2^1) & (\neg s_5^3 \vee s_4^3 \vee s_4^2). \end{array} \tag{19}$$

Similarly, instances of clause (16), for $1 \leq k \leq 5$ *and* $1 \leq j \leq 5$ *are simplified* *with the help of unit clause (9), (10), (13) and (14) to*

$$\begin{array}{lll}
(\neg s_1^1 \vee p_1) & (\neg s_2^2 \vee p_2) & (\neg s_4^3 \vee p_4) \\
(\neg s_2^1 \vee s_1^1 \vee p_2) & (\neg s_3^2 \vee s_2^2 \vee p_3) & (\neg s_5^3 \vee s_4^3 \vee p_5). \\
(s_2^1 \vee p_3) & (\neg s_4^2 \vee s_3^2 \vee p_4) & \\
 & (s_3^2 \vee p_5) &
\end{array} \tag{20}$$

The set of all non-redundant simplified instances of clause (17) consists of

$$\begin{array}{ccc}
(\neg s_1^1 \vee s_2^1) & (\neg s_2^2 \vee s_3^2) & (\neg s_4^3 \vee s_5^3) \\
 & (\neg s_3^2 \vee s_4^2) &
\end{array} \tag{21}$$

and set of all non-redundant simplified instances of clause (18) consists of

$$\begin{array}{llll}
(\neg p_1 \vee s_1^1) & (\neg s_1^1 \vee \neg p_2 \vee s_2^2) & (\neg s_2^2 \vee \neg p_3) & (\neg s_4^3 \vee \neg p_5). \\
(\neg p_2 \vee s_2^1) & (\neg s_2^1 \vee \neg p_3 \vee s_3^2) & (\neg s_3^2 \vee \neg p_4 \vee s_4^3) & \\
 & (\neg p_4 \vee s_4^2) & (\neg s_4^2 \vee \neg p_5 \vee s_5^3) &
\end{array} \tag{22}$$

Thus, the set $\mathsf{CBound}^2(p_1, \ldots, p_5)$ *consists of 26 clauses grouped in (19)–(22) above.*

3.4 SAT Encoding of Multiplicativity

Multiplicativity and complete multiplicativity of ± 1 sequences can be encoded in SAT in a rather straightforward way. Assuming that a Boolean sequence $p_1, \ldots p_n$ encodes a ± 1 sequence $x_1, \ldots, x_n$ so that the logical value 1 encodes the numerical value $+1$ and the logical value 0 encodes the numerical value -1, a SAT encoding of the fact that $x_{j \cdot k} = x_j \cdot x_k$ is captured by the following clauses, which enumerate all four combinations of values of x_j and x_k:

$$\begin{aligned}
\mathsf{prod}_{j,k} = \{&(\neg p_j \vee \neg p_k \vee p_{j \cdot k}), (p_j \vee p_k \vee p_{j \cdot k}), \\
&(\neg p_j \vee p_k \vee \neg p_{j \cdot k}), (p_j \vee \neg p_k \vee \neg p_{j \cdot k})\}
\end{aligned} \tag{23}$$

Then multiplicativity of $x_1, \ldots x_n$ is captured by instances of (23) for all coprime pairs $i < j$; and, by (2), complete multiplicativity of the sequence $x_1, \ldots, x_n$ is captured by instances of (23) for j and k such that every product $j \cdot k$ is generated only once.

For complete multiplicativity further optimisation is possible due to the fact that in any such sequence $x_{j^2} = +1$ for any $j \in \mathbb{N}^+$. It can be seen that the complete multiplicativity condition can be expressed by the union of the sets of clauses cmult_i defined below for every $i : 1 \leq i \leq n$.

$$\mathsf{cmult}_i = \begin{cases} \emptyset & \text{if } i \text{ is prime} \\ \{(p_i)\} & \text{if } i = j^2\text{, for some } j \geq 1 \\ \mathsf{prod}_{j,k} & \text{if none of the cases above applies} \\ & \text{and } j < k \text{ are some non-trivial divisors of } i. \end{cases}$$

3.5 Putting It All Together

We now have all the ingredients we need to define the CNF formulae used in our experiments. First we define the CNF encoding of finite sequences of length n having discrepancy bounded by C. It is defined as the conjunction of formulae expressing C-boundedness of its relevant subsequences (recall that every ± 1 sequence whose length does not exceed C is always C-bounded).

$$\mathsf{edp}(C,n) = \bigwedge_{d=1}^{\lfloor \frac{n}{C+1} \rfloor} \mathsf{CBound}^C(p_d, p_{2d}, \ldots, p_{\lfloor n/d \rfloor \cdot d}). \tag{24}$$

We assume here that for the different values of d sets $\mathsf{CBound}^C(p_d, x_{2d}, \ldots, p_{\lfloor n/d \rfloor \cdot d})$ share the same input propositions $p_1, \ldots, p_n$ but use different auxiliary propositions s_j^k. Then the following theorem is a direct consequence of Theorem 5.

Theorem 9. *For any assignment $I : \{p_1, \ldots, p_n\} \to \{0,1\}$ the following holds: there exists an extension of I to $I' : \mathrm{vars}(\mathsf{edp}(C,n)) \to \{0,1\}$ that is a model of $\mathsf{edp}(C,n)$ if, and only if, $I(p_1), \ldots, I(p_n)$ encodes a ± 1 sequence $x_1, \ldots, x_n$ of length n and discrepancy at most C.*

In our experiments we use two optimisations, which we present in the form of propositions. Both reduce significantly the size of the unsatisfiability certificate and have some noticeable effect on the running time. The first optimisation allows one to remove the 'don't care' propositions, which do not affect the satisfiability of the problem. The second optimisation breaks the symmetry in the problem.

Proposition 10. *Suppose that a ± 1 sequence $a_1, \ldots, a_n$ is C-bounded and either n is odd and C is even or n is even and C is odd. Then for an arbitrary value $b \in \{+1, -1\}$ the sequence $a_1, \ldots, a_n, b$ is C-bounded.*

Proof. It suffices to notice that $|\sum_{i=1}^{j} a_i|$ is odd if, and only if, j is odd. Thus, under the conditions of the proposition, $|\sum_{i=1}^{j} a_i| \leq C-1$, and the sequence can be extended arbitrarily. $\square$

Symmetry breaking [41] is a well-known technique to reduce search in combinatorial problems. In the context of propositional satisfiability, a *solution symmetry* [42] can be defined as a bijection on the set of assignments of truth values to a set of *solution variables* [43, 44]. As the discrepancy of a ± 1 sequence $x_1, \ldots, x_n$ is bounded by C if, and only if, the discrepancy of $-x_1, \ldots, -x_n$ is bounded by C, the permutation $l \mapsto \neg l$, where l is a literal, is a solution symmetry for the SAT encoding of the EDP. This symmetry induces an equivalence relation on the set of all assignments. Notice that either all assignments in an equivalence class generated by this symmetry equivalence relation satisfies the formula, or the class contains no satisfying assignment. A *symmetry breaking predicate* [41] is a propositional formula, which is true on at least one assignment in every equivalence class generated by the symmetry equivalence relation. Conjoining the symmetry breaking predicate with the formula ensures that the SAT solver finds few representative assignment for every equivalence class. It should be clear that p_l is an equivalence breaking predicate for the encoding of the EDP for every $l : 1 \leq 1 \leq n$. We summarise this argument as a proposition.

Proposition 11 (Symmetry breaking)**.** *For every $n > 0$ and $C > 0$, the formula $\mathsf{edp}(C,n)$ is satisfiable if, and only if, the formula $\mathsf{edp}(C,n) \wedge (p_l)$ is satisfiable, for some arbitrary but fixed value of l, $1 \leq l \leq n$.*

Proposition 11 introduces a rather simple form of symmetry breaking. It is an interesting problem to identify some other forms of symmetry in the encoding of the EDP, for example, symmetry *within* solutions [45, 46] and investigate their effect on the performance of SAT solvers.

From the propositional satisfiability point of view, the study of multiplicative and completely multiplicative sequences of bounded discrepancy can be seen an example of streamlining [47] or tunnelling [48]. Notice however, that multiplicative and completely multiplicative sequences of bounded discrepancy are interesting in their own right and, as mentioned in Section 2.1, the question whether the discrepancy of unrestricted ± 1 sequences is unbounded is equivalent to the question whether the discrepancy of completely multiplicative ± 1 sequences is unbounded.

We define two sets of clauses

$$\mathsf{edp_m}(C,n) = \mathsf{edp}(C,n) \cup \bigcup_{\substack{1<j<k\leq n \\ j,k \text{ are coprime} \\ j\cdot k \leq n}} \mathsf{prod}_{j,k}$$

and

$$\mathsf{edp_{cm}}(C,n) = \mathsf{edp}(C,n) \cup \bigcup_{i=1}^{n} \mathsf{cmult}_i .$$

The following statement is a direct consequence of Theorem 9.

Theorem 12. *For any assignment $I : \{p_1, \ldots, p_n\} \to \{0,1\}$ the following holds: there exists an extension of I to $I' : \mathrm{vars}(\mathsf{edp_m}(C,n)) \to \{0,1\}$ (or an extension of I to $I' : \mathrm{vars}(\mathsf{edp_{cm}}(C,n)) \to \{0,1\}$), which is a model of $\mathsf{edp_m}(C,n)$ (or $\mathsf{edp_{cm}}(C,n)$, respectively) if, and only if, $I(p_1), \ldots, I(p_n)$ encodes a multiplicative (or completely multiplicative, respectively) ± 1 sequence $x_1, \ldots, x_n$ of length n and discrepancy at most C.*

The completely multiplicative case can be further optimised based on the following observation.

Proposition 13. *The discrepancy of a completely multiplicative ± 1 sequence $x_1, \ldots, x_n$ is bounded by C, for some $C > 0$, if, and only if, $x_1, \ldots, x_n$ is C-bounded.*

Proof. The necessary condition is trivial by definition of discrepancy. For the sufficient condition we show that for any C-bounded sequence $x_1, \ldots, x_n$ and any $d > 1$ the subsequence $x_d, x_{2d}, \ldots, x_{\lfloor n/d \rfloor \cdot d}$ is C-bounded. Let $1 \leq j \leq \lfloor n/d \rfloor$. Then $|\sum_{i=1}^{j} x_{i\cdot d}| = |\sum_{i=1}^{j} (x_i \cdot x_d)| = |x_d \cdot \sum_{i=1}^{j} x_i| = |\sum_{i=1}^{j} x_i| \leq C$. $\square$

Finally notice that the fact that $x_1, \ldots, x_n$ is multiplicative does not imply that $-x_1, \ldots, -x_n$ is, so symmetry breaking described in Proposition 11 is not applicable for multiplicative and completely multiplicative sequences.

4 Results

Experimental setting In our experiments we use `Treengeling`, a parallel cube-and-conquer flavour of the `Lingeling` SAT solver [49] version aqw, the winner of the application *SAT-UNSAT* category of the SAT'13 competition [50], and the `Glucose` solver [51] version 3.0, the winner of the application *certified UNSAT* category of the SAT'13 competition [50]. All experiments were conducted on PCs equipped with an Intel Core i5-2500K CPU running at 3.30GHz and 16GB of RAM. Both solvers are used in a black-box manner with default parameters with the only exception of the activity heuristics [52], which contributes to the selection of variables for branching, being tuned to reduce the size of the unsatisfiability certificate for the case of $C = 2$ described below.

In our first series of experiments we investigate the discrepancy of unrestricted ± 1 sequences. We encode[3] the existence of a ± 1 discrepancy C sequence of length n into SAT as described in Section 3. We deploy both optimisations described in Proposition 10 and Proposition 11. We choose as l, for which we fix x_l to be $+1$, a colossally abundant number [53], which has many divisors and thus contribute to many homogeneous sequences. Specifically for $C = 2$, the choice of $l = 120$ is more beneficial for satisfiable instances; however, $l = 60$ results in a better reduction of the size of the unsatisfiability proof described below. For consistency of presentation, we use $l = 60$ in all our experiments for $C = 2$.

We establish that the maximal length of a ± 1 sequence of discrepancy 2 is 1160. The CNF formula $\mathsf{edp}(2, 1160)$ contains 11824 propositions and 41 884 clauses. It takes the `Treengeling` system about 430 seconds to find a satisfying assignment on our hardware configuration. One of the sequences of length 1160 of discrepancy 2 can be found in B. When applied to the CNF formula $\mathsf{edp}(2, 1161)$, which contains 11847 propositions and 41 970 clauses, `Treengeling` reports unsatisfiability. In order to corroborate this statement, we also use `Glucose`. It takes the solver about 800 seconds to generate a DRUP certificate of unsatisfiability. The correctness of the generated unsatisfiability certificate has been independently verified with the `drup-trim` tool [37].

The size of the certificate is about 1.88 GB. An experimental exploration of the effect of different solver options on the size of the certificate revealed that setting the `var-decay` option of `Glucose`, which controls the solver activity heuristic, to 0.995 reduces the unsatisfiability certificate roughly by 12.5% to 1.67 GB. Interestingly, deviating from the default options in other unsatisfiability cases reported below had a detrimental effect on the solver performance and, therefore, the default values were used in all other experiments.

The time needed to verify the certificate is comparable with the time needed to generate it. The RUP unsatisfiability certificate, that is the DRUP certificate with all information on the deleted clauses stripped, is 850.2MB; it takes the `drup-trim` tool about five and a half hours to verify it. Combined with Theorem 9, these two experiments yield a computer proof of the following statement.

[3]The problem generator and results can be found at `http://www.csc.liv.ac.uk/~konev/edp/`

Theorem 14. *The length of a maximal* ± 1 *sequence of discrepancy* 2 *is* 1160*.*

Thus we prove that the Erdős discrepancy conjecture holds true for $C = 2$.

When applied to $\mathsf{edp}(3, n)$ for increasing values of n our method could only produce sequences of discrepancy 3 of length in the region of 14 000, even though solvers were allowed to run for weeks. Since both multiplicativity and complete multiplicativity restrictions reduce severely the search space, in hope for better performance, we perform the second series of experiments to investigate the discrepancy bound for multiplicative and completely multiplicative sequences. Notice that the optimisation described in Proposition 11 is not applicable in this case as the fact that $x_1, \ldots, x_n$ is multiplicative does not imply that $-x_1, \ldots, -x_n$ is.

We saw in Example 2 that multiplicative sequences of discrepancy 1 are longer than completely multiplicative sequences. The longest completely multiplicative sequence of discrepancy 2 is known to contain 246 elements [54]; tests with $\mathsf{edp_m}$ show that the longest multiplicative sequence of discrepancy 2 has 344 elements. Thus it wouldn't be unreasonable to expect that the longest multiplicative discrepancy 3 sequence is longer than the longest completely multiplicative one, but is probably harder to find. It turns out that this expectation is wrong on both accounts.

We establish that the length of a maximal ± 1 completely multiplicative discrepancy 3 sequence coincides with the length of a maximal ± 1 multiplicative discrepancy 3 sequence and is equal to 127 645. It takes `Treengeling` about one hour and fifty minutes to find a satisfying assignment to $\mathsf{edp_{cm}}(3, 127\,645)$, which contains 3 484 084 propositions and 13 759 785 clauses, and about one hour and thirty five minutes to find a satisfying assignment to $\mathsf{edp_m}(3, 127\,645)$, which also contains 3 484 084 propositions but 14 813 052 clauses.

It takes the `Glucose` solver just under eight hours to generate an approximately 1.28 GB DRUP proof of unsatisfiability for $\mathsf{edp_{cm}}(3, 127\,646)$, which contains 3 484 084 propositions and 13 759 809 clauses, and about nine and a half hours to generate an approximately 1.56 GB DRUP proof of unsatisfiability for $\mathsf{edp_m}(3, 127\,646)$, which again contains the same number of propositions but 14 813 076 clauses.

The optimisation of Proposition 13 leads to a reduction in the problem size for the completely multiplicative case (446 753 propositions and 1 738 125 clauses for length 127 645 and 446 759 propositions and 1 738 149 clauses for length 127 646) and a significant reduction both in the `Treengeling` running time (about 20 and 30 minutes, respectively) and in the size of the DRUP certificate, which is about 0.84Gb.

So we get a computer-aided proof of another sharp bound on the sizes of maximal sequences of bounded discrepancy.

Theorem 15. *The length of a maximal multiplicative* ± 1 *sequence of discrepancy* 3 *equals the length of a maximal completely multiplicative* ± 1 *sequence of discrepancy* 3 *and is* $127\,645$*.*

Unrestricted sequences of discrepancy 3 can still be longer than 127 646: by requiring that only first 127 600 elements of a sequence are completely multiplicative, we generate a 130,000 long EDP3 sequence in about one hour and fifty minutes thus establishing a slightly better lower bound on the length of ± 1 sequences of discrep-

Discrepancy bound	Completely multiplicative	Multiplicative	Unconstrained
C = 1	9	**11**	**11**
C = 2	246	344	1160
C = 3	**127 645**	**127 645**	>130 000

Table 1: Maximal length of ± 1 sequences of bounded discrepancy

ancy 3 than the one from Theorem 15. The solvers struggle to expand it much further. Notice that the optimisation of Proposition 13 is not applicable here.

We summarise known facts about discrepancy of unrestricted, multiplicative and completely multiplicative sequences in Table 1. We highlight in boldface cases where the lengths of maximal sequences of different kinds are equal.

5 Conclusions

We have demonstrated that SAT-based methods can be used to tackle the longstanding mathematical questions related to discrepancy of ± 1 sequences. To the best of our knowledge, this is the first use of automatically generated unsatisfiability certificates as formal proofs of non-trivial mathematical statements. As a result, we able to identify the exact boundary between satisfiability and unsatisfiability for the encoding of the EDP for $C = 2$, thus identifying the longest sequences of discrepancy 2. We have established the surprising fact that the lengths of the longest multiplicative and completely multiplicative sequences of discrepancy 3 coincide. The latter result helps to establish a novel lower bound on the length of the longest discrepancy 3 sequence.

The general question of the existence of a finite bound on the length of ± 1 sequences of discrepancy 3 (that is, the Erdős conjecture for $C = 3$) remains open. Considering the tenfold gap between the sizes of unrestricted discrepancy 3 sequences that solvers can find and maximal (completely) multiplicative discrepancy 3 sequences it would seem that without fresh ideas the conjecture is unlikely to be settled by a brute-force analysis, even helped by the modern SAT solver technology.

There is a noticeable asymmetry in our findings. The fact that a sequence of length 1160 has discrepancy 2 can be relatively easily checked manually. It is harder but not impossible to verify the correctness of the discrepancy bound for 127 645-long sequences. On the other hand, even though improvements to our method shortened the Wikipedia-size 13 GB proof reported in [30] more than tenfold, passing the psychological barrier of 1 GB, it still probably is one of the longest proofs of a non-trivial mathematical result. It is equally improbable that a mathematician would verify by hand ten billion or half a billion of automatically generated proof lines. Having said that, the reduction of proof size will be useful for any future analysis in an attempt to identify patterns and lemmas and produce a compact proof more amenable for human comprehension.

Until such a human-comprehensible proof is found, the epistemic status of our results remains rather peculiar: we *know* that a proof exists, we even *have* it, we can handle it, check it by a third-party tool, analyse it, transform it, but we cannot understand it. To what extent this proof can be recognised as a proof is then a subject of foundational debate on the future of computer mathematics and goes beyond the scope of this article.

Acknowledgements

The authors would like to thank Armin Biere, Marijn Heule, Pascal Fontaine, Donald E. Knuth, Laurent Simon and Laurent Théry for helpful discussions, comments and ideas.

References

[1] W. McCune, Solution of the Robbins problem, J. Autom. Reasoning 19 (3) (1997) 263–276. doi:10.1023/A:1005843212881.

[2] K. Appel, W. Haken, Every map is four colourable, Bulletin of the American Mathematical Society 82 (1976) 711–712.

[3] T. C. Hales, A proof of the Kepler conjecture, Annals of Mathematics 162 (3) (2005) 1065–1185. doi:10.4007/annals.2005.162.1065.

[4] A. Bundy, Automated theorem provers: a practical tool for the working mathematician?, Ann. Math. Artif. Intell. 61 (1) (2011) 3–14. doi:10.1007/s10472-011-9248-8.

[5] J. Avigad, J. Harrison, Formally verified mathematics, Commun. ACM 57 (4) (2014) 66–75. doi:10.1145/2591012.

[6] C. W. H. Lam, L. Thiel, S. Swiercz, The nonexistence of finite projective planes of order 10, Canadian Journal of Mathematics 41 (6) (1989) 1117–1123. doi:10.4153/CJM-1989-049-4.

[7] T. Tymoczko, The four-color problem and its philosophical significance, The Journal of Philosophy 6 (2) (1979) 57–83.

[8] N. Robertson, D. Sanders, P. Seymour, R. Thomas, The four-colour theorem, Journal of Combinatorial Theory, Series B 70 (1) (1997) 2–44.

[9] G. Gonthier, Formal proof—the four-color theorem, Notices Amer. Math. Soc. 55 (11) (2008) 1382–1393.

[10] T. C. Hales, J. Harrison, S. McLaughlin, T. Nipkow, S. Obua, R. Zumkeller, A revision of the proof of the Kepler conjecture, Discrete & Computational Geometry 44 (1) (2010) 1–34. doi:10.1007/978-1-4614-1129-1_9.

[11] G. Gonthier, A. Asperti, J. Avigad, Y. Bertot, C. Cohen, F. Garillot, S. L. Roux, A. Mahboubi, R. O'Connor, S. O. Biha, I. Pasca, L. Rideau, A. Solovyev, E. Tassi, L. Théry, A machine-checked proof of the odd order theorem, in: Proceedings of the 4th International Conference on Interactive Theorem Proving, Vol. 7998 of Lecture Notes in Computer Science, 2013, pp. 163–179. `doi:10.1007/978-3-642-39634-2_14`.

[12] J. Beck, W. W. L. Chen, Irregularities of Distribution, Cambridge University Press, Cambridge, 1987.

[13] B. Chazelle, The Discrepancy Method: Randomness and Complexity, Cambridge University Press, New York, 2000.

[14] J. Matoušek, Geometric Discrepancy: An Illustrated Guide, Vol. 18 of Algorithms and combinatorics, Springer, 1999.

[15] J. Beck, V. T. Sós, Discrepancy theory, in: R. L. Graham, M. Grötschel, L. Lovász (Eds.), Handbook of combinatorics, Vol. 2, Elsivier, Amsterdam, 1995, pp. 1405–1446.

[16] N. Alon, Transmitting in the *n*-dimensional cube, Discrete Applied Mathematics 37-38 (1992) 9–11. `doi:10.1016/0166-218X(92)90121-P`.

[17] S. Muthukrishnan, A. Nikolov, Optimal private halfspace counting via discrepancy, in: Proceedings of the 44th Symposium on Theory of Computing, STOC '12, ACM, New York, NY, USA, 2012, pp. 1285–1292. `doi:10.1145/2213977.2214090`.

[18] W. Chen, A. Srivastav, G. Travaglini, A Panorama of Discrepancy Theory, Vol. 2107 of Lecture Notes in Mathematics, Springer International Publishing, 2014.

[19] J. Matoušek, J. Spencer, Discrepancy in arithmetic progressions, Journal of the American Mathematical Society 9 (1) (1996) 195–204. `doi:10.1090/S0894-0347-96-00175-0`.

[20] K. F. Roth, Remark concerning integer sequence, Acta Arithmetica 9 (1964) 257–260.

[21] D. Bilyk, Roth's orthogonal function method in discrepancy theory and some new connections, in: W. Chen, A. Srivastav, G. Travaglini (Eds.), A Panorama of Discrepancy Theory, Vol. 2107 of Lecture Notes in Mathematics, Springer, 2014, pp. 71–158.

[22] P. Erdős, Some unsolved problems, The Michigan Mathematical Journal 4 (3) (1957) 291–300.

[23] N. Čudakov, Theory of the characters of number semigroups, Journal of Indian Mathematical Society 20 (1956) 11–15.

[24] A. Nikolov, K. Talwar, On the hereditary discrepancy of homogeneous arithmetic progressions, CoRR abs/1309.6034v1.

[25] T. Gowers, Erdős and arithmetic progressoins, in: Erdős Centennial conference, 2013, http://www.renyi.hu/conferences/erdos100/program.html. Last accessed April, 10 2014.

[26] P. Borwein, S. K. K. Choi, M. Coons, Completely multiplicative functions taking values in $\{1, -1\}$, Transactions of the American Mathematical Society 362 (12) (2010) 6279–6291. doi:10.1090/S0002-9947-2010-05235-3.

[27] A. R. D. Mathias, On a conjecture of Erdős and Čudakov, in: B. Bollobás, A. Thomason (Eds.), Combinatorics, geometry and probability: Proceedings of the conference dedicated to Paul Erdős on the occasion of his 80th birthday, Cambridge University Press, 1993.

[28] D. Polymath, Erdős discrepancy problem: Polymath wiki, http://michaelnielsen.org/polymath1/index.php?title=The_Erdős_discrepancy_problem. Last accessed April, 10 2014 (2010).

[29] T. Gowers, Is massively collaborative mathematics possible?, http://gowers.wordpress.com/2009/01/27/is-massively-collaborative-mathematics-possible/. Last accessed April, 10 2014 (2009).

[30] B. Konev, A. Lisitsa, A SAT attack on the Erdős discrepancy conjecture, CoRR abs/1402.2184.

[31] B. Konev, A. Lisitsa, A SAT attack on the Erdős discrepancy conjecture, in: Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing, SAT 2014, 2014, to appear.

[32] R. Le Bras, C. P. Gomes, B. Selman, On the Erdős discrepancy problem, in: Proceeding of the 20th International Conference on Principles and Practice of Constraint Programming, CP 2014, Vol. 8656 of Lecture Notes in Computer Science, Springer, 2014, pp. 440–448.

[33] T. M. Apostol, Introduction to analytic number theory, Undergraduate Texts in Mathematics, Springer, New York, NY, USA, 1976.

[34] W. Rautenberg, A Concise Introduction to Mathematical Logic, 3rd Edition, Springer, New York, 2010.

[35] A. Biere, M. Heule, H. van Maaren, T. Walsh (Eds.), Handbook of Satisfiability, Vol. 185 of Fronteers in Artificial Intelligence and Applications, IOS Press, Amsterdam, 2009.

[36] E. I. Goldberg, Y. Novikov, Verification of proofs of unsatisfiability for CNF formulas, in: Proceedings of Design, Automation and Test in Europe Conference and Exposition (DATE 2003), 3-7 March 2003, Munich, Germany, 2003, pp. 10886–10891.

[37] M. Heule, W. A. H. Jr., N. Wetzler, Trimming while checking clausal proofs, in: Proceedings of Formal Methods in Computer-Aided Design, FMCAD 2013, IEEE, 2013, pp. 181–188.

[38] O. Roussel, V. M. Manquinho, Pseudo-boolean and cardinality constraints, in: A. Biere, M. Heule, H. van Maaren, T. Walsh (Eds.), Handbook of Satisfiability, Vol. 185 of Fronteers in Artificial Intelligence and Applications, IOS Press, Amsterdam, 2009, pp. 695–733.

[39] C. Sinz, Towards an optimal CNF encoding of boolean cardinality constraints, in: Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings, Vol. 3709 of Lecture Notes in Computer Science, Springer, 2005, pp. 827–831. `doi:10.1007/11564751_73`.

[40] G. S. Tseitin, On the complexity of derivation in propositional calculus, in: A. O. Slisenko (Ed.), Studies in Constructive Mathematics and Mathematical Logic, Part 2, Consultants Bureau, New York, 1970, pp. 115–125.

[41] J. M. Crawford, M. L. Ginsberg, E. M. Luks, A. Roy, Symmetry-breaking predicates for search problems, in: Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96), Morgan Kaufmann, 1996, pp. 148–159.

[42] D. A. Cohen, P. Jeavons, C. Jefferson, K. E. Petrie, B. M. Smith, Symmetry definitions for constraint satisfaction problems, Constraints 11 (2-3) (2006) 115–137.

[43] R. Backofen, S. Will, Excluding symmetries in constraint-based search, in: Proceedings of the 5th International conference on Principles and Practice of Constraint Programming, CP'99, Vol. 1713 of Lecture Notes in Computer Science, Springer, 1999, pp. 73–87.

[44] I. P. Gent, K. E. Petrie, J.-F. Puget, Symmetry in constraint programming, in: F. Rossi, P. van Beek, T. Walsh (Eds.), Handbook of Constraint Programming, Elsivier, 2006, pp. 329–376.

[45] M. Heule, T. Walsh, Symmetry in solutions, in: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, AAAI Press, 2010.

[46] C. Hartman, M. Heule, K. Kwekkeboom, A. Noels, Symmetry in gardens of eden, Electr. J. Comb. 20 (3) (2013) P16.

[47] C. P. Gomes, M. Sellmann, Streamlined constraint reasoning, in: Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming, CP 2004, Vol. 3258 of Lecture Notes in Computer Science, Springer, 2004, pp. 274–289.

[48] M. Kouril, J. V. Franco, Resolution tunnels for improved SAT solver performance, in: Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing, SAT 2005, Vol. 3569 of Lecture Notes in Computer Science, Springer, 2005, pp. 143–157.

[49] A. Biere, Lingeling, Plingeling and Treengeling entering the SAT Competition 2013, in: Proceedings of SAT Competition 2013, University of Helsinki, Helsinki, 2013, pp. 51–52.

[50] A. Balint, A. Belov, M. J. H. Heule, M. Järvisalo (Eds.), Proceedings of SAT competition 2013, University of Helsinki, 2013.

[51] G. Audemard, L. Simon, Glucose 2.3 in the SAT 2013 Competition, in: Proceedings of SAT Competition 2013, University of Helsinki, Helsinki, 2013, pp. 42–43.

[52] N. Eén, N. Sörensson, An extensible sat-solver, in: E. Giunchiglia, A. Tacchella (Eds.), Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing, SAT 2003, Vol. 2919 of Lecture Notes in Computer Science, Springer, 2003, pp. 502–518.

[53] L. Alaoglu, P. Erdős, On highly composite and similar numbers, Transactions of the American Mathematical Society 56 (3) (1944) 448–469.

[54] D. Polymath, Human proof that completely multiplicative sequences have discrepancy greater than 2, `http://michaelnielsen.org/polymath1/index.php?title=Human_proof_that_completely_multiplicative_sequences_have_discrepancy_at_least_2`. Last accessed April, 10 2014 (2011).

A Proof of Proposition 4

In this section we give proofs of the technical result used in the main text. We re-state the proposition for the reader's convenience.

PROPOSITION 4. *Let $\Phi(p_1, \ldots, p_n)$ be as defined above. Then*

(i) For any assignment $I : \mathrm{vars}(\Phi(p_1, \ldots, p_n)) \to \{0, 1\}$ such that I satisfies $\Phi(p_1, \ldots, p_n)$, any $1 \leq j \leq n$ and $1 \leq k \leq n$ we have

$$I(s_j^k) = 1 \quad \textit{if, and only if,} \qquad \sum_{i=1}^{j} I(p_i) \geq k.$$

(ii) For any 0/1-sequence $(a_1, \ldots, a_n) \in \{0, 1\}^n$ there exists an assignment $I : \mathrm{vars}(\Phi(p_1, \ldots, p_n)) \to \{0, 1\}$ such that $I(p_i) = a_i$, for $1 \leq i \leq n$; I satisfies $\Phi(p_1, \ldots, p_n)$; and for any $r \leq n$ and $j \leq n$ if $\sum_{i=1}^{j} a_i \leq r$ then $I(s_j^k) = 0$, for $r < k \leq n$.

Proof. (i) The proof proceeds by induction on the lexicographical partial order $\prec$ on pairs of non-negative integers: $(j, k) \prec (j', k')$ iff $j < j' \vee ((j = j') \wedge (k < k'))$. Fix some $n \geq 1$.

Consider cases:

- Suppose that $j = k = 1$. Then formula (8), one of the conjuncts of $\Phi^n(p_1, \ldots, p_n)$, instantiates to $s_1^1 \leftrightarrow (s_0^1 \vee (s_0^0 \wedge p_1)$. Therefore, for an assignment I such that I satisfies $\Phi^n(p_1, \ldots, p_n)$ we have $I(s_1^1 \leftrightarrow (s_0^1 \vee (s_0^0 \wedge p_1)) = 1$. Furthermore, for the satisfying assignment we have $I(s_0^1) = 0$ and $I(s_0^0) = 1$. It follows then that $I(s_1^1) = I(p_1)$, which is equivalent to the statement of the proposition for the case $k = j = 1$.
- Suppose that $j = 1$ and $k > 1$. For a satisfying assignment I we have $I(s_1^k) = 0$ (as (9) is a conjunct of $\Phi^n(p_1, \ldots, p_n)$). On the other hand for $k > 1$ we have $\sum_{i=1}^{1} I(p_i) < k$. Thus the statement of the proposition holds true in this case.
- Suppose that $j > 1$, $k \geq 1$. For a satisfying assignment I we have $I(s_j^k) = 1$ if and only if $I(s_{j-1}^k) = 1$ or $I(s_{j-1}^{k-1} \wedge p_j) = 1$ (by satisfaction of (8)). By induction hypothesis the later is equivalent to $\sum_{i=1}^{j-1} I(p_i) \geq k$ or $\sum_{i=1}^{j-1} I(p_i) \geq k - 1$ and $I(p_j) = 1$, which in turn is equivalent to $\sum_{i=1}^{j} I(p_i) \geq k$.

(ii) First notice that any assignment $I_p : \{p_1, \ldots, p_n) \to \{0, 1\}$ can be extended in a unique way to the assignment $I : \mathrm{vars}(\Phi(p_1, \ldots, p_n)) \to \{0, 1\}$. Indeed, satisfaction of (9) and (10) defines uniquely the values of satisfying assignment I on s_j^k for the cases $0 \leq j < k \leq n$ and $k = 0; 0 \leq j \leq n$, respectively. Further, using satisfaction condition for (8) the values of I on the remaining variables s_j^k with $1 \leq k \leq n, 1 \leq j \leq n$ are defined uniquely by induction on $\prec$. The remaining condition, that is for any $r \leq n$ and $j \leq n$ if $\sum_{i=1}^{j} a_i \leq r$ then $I(s_j^k) = 0$, for $r < k \leq n$, now follows from item (*i*) above.

$\square$

B A sequence of length 1160 and discrepancy 2

We give a graphical representation of one of the sequences of length 1160 obtained from the satisfying assignment computed with the `Treengeling` solver. Here $+$ stands for $+1$ and $-$ for -1, respectively.

```
− + + − + − − + + − + + − + − − + − − + + − + − − + − − +
+ − + − − + + − + + − + − + + − − + + − + − − − + − + + −
+ − − + − − + + + + − − + − − + + − + − − + + − + + − − −
− + + − + + − + − + + − − + + − + − + − − − + + − + − − +
+ − + + − + − − + + − + − − + − − − + − + + − + − − + + −
+ + − + − − + − − + + − + + − + − − + + − + − − + + + − +
− + − − − − + + + − + − − + − − + + + − − − + + − + + − +
− − + − − + + + − − + − + − + − − + − + + + − + + − + − −
+ − − + + − + − − + + − + + − + − − + − − + + − − + + + −
− − + + + − + − − − + + − + − − + + − − + − + − − + − + +
+ − + − − + + − + + − + − − + + − + − − + − − + + − + − −
+ + − − + − + + − + − + − − + − + − + + − + − − + + − + −
− + − − + + − + − + − + + − + − + − + + − − − + − + − − +
+ + + − − + − − − + + − + − + + − + − − + + − + − − + − −
+ + − + − − + + + + − − + − − − + − + + + + − − + − − + +
− + + − + − − + + − + − − + − − + + − + − − + + − + + − +
− − + + − + − − + − − + + − + + − + − − − − + + + − + − −
+ + − − + + + − − − + − + + − + − − + − + + − − − + − + +
− + + − + − − + − − + + − − + + + + − + − − + − − + − − +
+ + + − − + − − + + + − − − + + − + + − + − − + + − − + −
+ − − + − − + + − + + − + − − + − − + − + + + − + + − + −
− + − − + + − − + − + + − + + − + − − + − − + − − + + − +
+ − + − − + + + − − − + + − + − − + + − + + − − − + + + −
− − + + − + + − − − − + + + − − + − + + − + − − + − − + +
− + − − + + − + + − + − + + − − + + − − + + − − − − + + +
− + + − − + + − − − − + + − + + + − − + + − − − + + + − −
− − + − + − + + − + + − + + − + − + − − − − + + + − − + +
− + − − + + − + + − + − − + − − + − − + + − + − − + + − +
+ − + − − + + − − + − + − − + − + − + − + + + + − − − + −
+ − + + − − + − − + − + − + − + + − + − + + + − − + − + −
− + − − + − + + + − − + − + + + − − − + + − + − − + − − +
+ − + + − − + + − − − + + − + − + + − − + + − + − − − + −
+ + − + − − + − + + − − + + − + − − + + − + − − + − + + +
− + − − + + − − + − + − + + + − − + − + − − + + − + + − +
− − + − − + − + + − − − + − + + − + − + + − − + + − + − −
+ + + − + − − − − + + − − + − + + − + − + + − − + + − + −
− + + − + − + + − − + + − + − − − + − + + − + − − + + + −
− − − + − + − + + − − + + − + − − + + − + + − + + − + − −
+ − − + − − + + + + − − − + + − − − + − + − + + − + − + +
+ − − + − + + − − + − + − − + − + − + + − − − + + + − + +
```