Near-Optimal Encodings of Cardinality Constraints

Andrew Krapivin
Carnegie Mellon University, Pittsburgh, PA, USA

Benjamin Przybocki
Carnegie Mellon University, Pittsburgh, PA, USA

Bernardo Subercaseaux
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

We present several novel encodings for cardinality constraints, which use fewer clauses than previous encodings and, more importantly, introduce new generally applicable techniques for constructing compact encodings. First, we present a CNF encoding for the $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ constraint using $2n + 2\sqrt{2n} + O(\sqrt[3]{n})$ clauses, thus refuting the conjectured optimality of Chen's product encoding. Our construction also yields a smaller monotone circuit for the threshold-2 function, improving on a 50-year-old construction of Adleman and incidentally solving a long-standing open problem in circuit complexity. On the other hand, we show that any encoding for this constraint requires at least $2n + \sqrt{n+1} - 2$ clauses, which is the first nontrivial unconditional lower bound for this constraint and answers a question of Kučera, Savický, and Vorel. We then turn our attention to encodings of $\mathsf{AtMost}_k(x_1, \ldots, x_n)$, where we introduce *grid compression*, a technique inspired by hash tables, to give encodings using $2n + o(n)$ clauses as long as $k = o(\sqrt[3]{n})$ and $4n + o(n)$ clauses as long as $k = o(n)$. Previously, the smallest known encodings were of size $(k+1)n + o(n)$ for $k \leq 5$ and $7n - o(n)$ for $k \geq 6$.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming; Theory of computation → Automated reasoning; Theory of computation → Circuit complexity

Keywords and phrases CNF encodings, cardinality constraints, circuit complexity

Digital Object Identifier 10.4230/LIPIcs.SAT.2026.23

Related Version *Full Version*: `https://arxiv.org/abs/2603.28954` [26]

Funding *Andrew Krapivin*: Krapivin was supported by the NSF grant CNS2504471, Packard Foundation grant 2020-71730, and the Jeanne B. and Richard F. Berdik ARCS Pittsburgh Endowed Scholar Award.
Benjamin Przybocki: Przybocki was supported by NSF grant DMS-2434625 and the NSF Graduate Research Fellowship Program under Grant No. DGE-2140739.
Bernardo Subercaseaux: Subercaseaux was supported by NSF grant DMS-2434625.

Acknowledgements We are grateful to Petr Savický for finding an error in an earlier version of this paper. We also thank Marijn Heule for his suggestions on improving the presentation of this paper.

1 Introduction

Cardinality constraints are a fundamental building block used to encode problems into conjunctive normal form (CNF). Due to their ubiquitous nature and importance to SAT solving, cardinality constraints have been extensively studied by the SAT community from both theoretical and experimental perspectives (see, e.g., [5–7, 11, 15, 17, 29, 31, 33, 34, 36]).

The most basic cardinality constraint is $\mathsf{AtMostOne}(x_1, \ldots, x_n)$, which asserts that at most one boolean variable x_i is true. While this constraint can be encoded using $\binom{n}{2}$ clauses:

$$\mathsf{AtMostOne}(x_1, \ldots, x_n) \iff \bigwedge_{1 \leq i < j \leq n} (\overline{x_i} \vee \overline{x_j}),$$

this quadratic blowup in the number of clauses is problematic when n is large. Fortunately, by introducing auxiliary variables, there are several encodings for $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ using only $O(n)$ clauses, such as [4, 19, 36, 37]. Empirically, SAT solvers perform much better when using linear encodings rather than the quadratic one [33]. Which cardinality constraint encoding is the most performant in practice depends on a number of factors, including the specific solver used and the encoding of the remaining constraints, but we focus on three factors that are amenable to theoretical analysis: the number of clauses in the encoding, the number of auxiliary variables, and whether the encoding is propagation complete (also known as arc consistent [18]), a notion defined in Section 2.

Prior to the present work, the smallest known encoding for $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ was Chen's product encoding [11], which uses $2n + 4\sqrt{n} + O(\sqrt[4]{n})$ clauses and $2\sqrt{n} + O(\sqrt[4]{n})$ auxiliary variables while also being propagation complete. Chen conjectured his encoding to be optimal with respect to the number of clauses. Our first contribution is to refute this conjecture:

▶ **Theorem 1** (Multipartite Encoding). *There is a propagation-complete encoding of the $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ constraint using $2n + 2\sqrt{2n} + O(\sqrt[3]{n})$ clauses and $\sqrt{2n} + O(\sqrt[3]{n})$ auxiliary variables.*

Not only does our encoding improve upon the product encoding with respect to both the number of clauses and the number of auxiliary variables, but it also has a few conceptually interesting aspects. Perhaps surprisingly, our encoding includes an AtMost_2 constraint within its definition, which makes it 3-CNF despite the fact that $\mathsf{AtMostOne}$ is a 2-CNF function. While all previous linear encodings for this constraint in the literature are 2-CNF, Kučera, Savický, and Vorel [29] asked whether optimal propagation-complete encodings of $\mathsf{AtMostOne}$ are 2-CNF; our result gives evidence to the contrary. Moreover, as we show in Section 3.4, our construction yields a smaller monotone boolean circuit for the *threshold-2* function, the negation of $\mathsf{AtMostOne}$, which improves on a 50-year-old construction of Adleman,[1] answers a 47-year-old open question from Bloniarz [8, p. 158], and has implications for the so-called single-level conjecture [3, 10, 23, 30, 32].

Our encoding originates from reinterpreting Chen's product encoding, traditionally presented in terms of a grid, as involving a complete bipartite graph. With this change of perspective, any graph with n edges can form the basis of an encoding of $\mathsf{AtMostOne}$, with $2n$ clauses that relate each edge to a pair of auxiliary variables corresponding to its endpoints and additional constraints depending on the graph structure. Our improvement in the second-order term comes from using a complete multipartite graph with $\omega(1)$ parts, which has edge density 1 rather than $1/2$ as in a complete bipartite graph; this allows for fewer vertices (i.e., auxiliary variables) for the same number of edges.

With regard to lower bounds, Kučera, Savický, and Vorel [29] proved that every propagation-complete encoding of $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ requires $2n + \sqrt{n} - 2$ clauses for $n \geq 7$, so Theorem 1 is close to optimal. On the other hand, prior to the present work, no nontrivial lower bound was known without assuming propagation completeness.[2] Our second contribution provides such a bound and answers a question from [29]:

▶ **Theorem 2.** *Every encoding of the $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ constraint has at least $2n + \sqrt{n+1} - 2$ clauses for $n \geq 8$.*

Together with Theorem 1 (or Chen's result), this implies that the minimum number of clauses in an encoding of $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ is $2n + \Theta(\sqrt{n})$.

[1] Adleman's construction was first mentioned in print by Bloniarz [8].

[2] Sinz [36] claimed that every encoding of $\mathsf{AtMost}_k(x_1, \ldots, x_n)$ has at least n clauses for all $k \in [n-2]$, but his proof has an irreparable gap. Indeed, the claim is false in general: Ben-Haim, Ivrii, Margalit, and Matsliah [7] showed that $\mathsf{AtMost}_k(x_1, \ldots, x_n)$ can be encoded using $O(\log n)$ clauses when $k = n - O(1)$.

Next, we turn to the constraint $\mathsf{AtMost}_k(x_1, \ldots, x_n)$, which asserts that at most k of the boolean variables $x_1, \ldots, x_n$ are true. Sinz [36] gave an encoding for this using $7n-3\lfloor \log n \rfloor -6$ clauses and $2n-2$ auxiliary variables. When k is large relative to n, this is the smallest known encoding for $\mathsf{AtMost}_k(x_1, \ldots, x_n)$ as measured by the number of clauses. Our third contribution is to present smaller encodings when k is small relative to n:

▶ **Theorem 3** (Grid Compression). *There is an encoding of the $\mathsf{AtMost}_k(x_1, \ldots, x_n)$ constraint using $4n + O\left(\sqrt[3]{kn^2}\right)$ clauses and $O\left(\sqrt[3]{kn^2}\right)$ auxiliary variables.*

▶ **Theorem 4** (Disjunctive Grid Compression). *There is an encoding of the $\mathsf{AtMost}_k(x_1, \ldots, x_n)$ constraint using $2n + O\left(\sqrt{nk^3 \log_k^3 n}\right)$ clauses and $O\left(\sqrt{nk^3 \log_k^3 n}\right)$ auxiliary variables.*

In particular, for $k = o(n)$, Theorem 3 gives an encoding using $4n + o(n)$ clauses and $o(n)$ auxiliary variables; for $k = o(\sqrt[3]{n})$, Theorem 4 gives an encoding using $2n + o(n)$ clauses and $o(n)$ auxiliary variables. We show in Corollary 10 that encoding $\mathsf{AtMost}_k(x_1, \ldots, x_n)$ requires $2(n-k) + \Omega(\sqrt{n-k})$ clauses, so Theorem 4 notably implies that the minimum number of clauses in an encoding of $\mathsf{AtMost}_k(x_1, \ldots, x_n)$ is $2n + \widetilde{\Theta}(\sqrt{n})$ for any $k = \log^{O(1)} n$.

Unfortunately, neither our encodings for AtMost_k nor Sinz's encoding are propagation complete. The smallest known propagation-complete encoding when k is a small constant is the *generalized product encoding* [17], which uses $(k+1)n + O(k^2 n^{k/(k+1)})$ clauses and $O(kn^{k/(k+1)})$ auxiliary variables for $k = o(\log n / \log\log n)$. For large k, the smallest known propagation-complete encoding combines the AKS sorting network [2] and a CNF encoding of sorting networks [5, 14], and it uses $O(n \log k)$ clauses and auxiliary variables. Nonetheless, the disjunctive grid compression encoding turns out to be competitive in practice for some benchmarks. While our motivation for this work was primarily theoretical, these preliminary empirical results highlight that our techniques can be practical, and they cast doubt on the conventional wisdom that propagation completeness is essential for performance. We discuss this further in Section 6 and in the full version of this paper [26, Appendix G].

The encodings from Theorem 3 and Theorem 4 are based on a new technique that we call *grid compression*. The technique consists of arranging the input variables into a grid M, which is then "compressed" into a smaller grid L. Then, we enforce that at most k variables in L are true, which is cheaper since L is smaller. The techniques to map M into L are inspired by those used in the construction of hash tables [12, Chapter 11].

In addition to grid compression, Theorem 4 leverages a novel paradigm we call *disjunctive switching*, which aims to address a common problem that arises when translating algorithmic ideas into CNF encodings. When an algorithm can take different branches according to some condition (i.e., a *switch*), its runtime will only depend on the taken branch, whereas naïve CNF encodings include clauses for each possible branch, and thus their size is based on the sum of all branches. In a nutshell, disjunctive switching introduces wide clauses stating that *some* branch will be taken, and then succinctly enforces that branches whose condition is not met cannot be taken, thus making the taken branch the only way to satisfy the disjunction. Besides applying disjunctive switching to the disjunctive grid compression encoding, we illustrate its utility by showing how the generalized product encoding for $\mathsf{AtMost}_k(x_1, \ldots, x_n)$ [17], which uses $(k+1)n + O(k^2 n^{k/(k+1)})$ clauses for $k = o(\log n / \log\log n)$, can be "disjunctivized" into an encoding with only $2n + O(kn^{k/(k+1)})$ clauses (see Theorem 11). If not for Theorem 4, this would be the smallest known encoding of this constraint for $k = o(\log n / \log\log n)$.

Due to space constraints, most proofs are omitted and can be found in the full version of this paper [26].

2 Preliminaries

A *CNF formula* is a set of clauses, each of which is a set of literals, and the *size* of such a formula is the number of clauses it contains. Given a set of variables X, we denote by $\mathrm{lit}(X) := X \cup \{\overline{x} \mid x \in X\}$ the set of literals over those variables. For a partial assignment $\tau : \mathcal{V} \to \{\bot, \top\}$ and a formula φ, we denote by $\varphi|_\tau$ the formula obtained by eliminating from φ each clause satisfied by τ, and then from each remaining clause eliminating every literal ℓ such that $\tau \models \overline{\ell}$. We will write $\mathsf{SAT}(\varphi)$ to say that $\tau \models \varphi$ for some assignment τ.

We now present the two main definitions used in this paper.

▶ **Definition 5** (CNF Encoding). *Given a boolean function $f\colon \{\bot,\top\}^n \to \{\bot,\top\}$, and a sequence of propositional variables $X := (x_1, \ldots, x_n)$, we say that a CNF formula φ over variables $X \sqcup Y$* encodes *f if, for every assignment τ of X,*

$$f(\tau(x_1), \ldots, \tau(x_n)) = \top \iff \mathsf{SAT}(\varphi|_\tau).$$

The variables in Y are called auxiliary *variables, whereas those in X are called* input *variables.*

▶ **Definition 6** (Propagation Completeness [29]). *Let $f\colon \{\bot,\top\}^n \to \{\bot,\top\}$ be a boolean function and φ a CNF encoding for f over input variables X and auxiliary variables Y. We say that φ is* propagation complete *if for every $\ell_1, \ldots, \ell_m \in \mathrm{lit}(X)$*

$$\left(\varphi \wedge \bigwedge_{i=1}^{m-1} \ell_i\right) \models \ell_m \implies \left[\left(\varphi \wedge \bigwedge_{i=1}^{m-1} \ell_i\right) \vdash_1 \ell_m \;\; \text{or} \;\; \left(\varphi \wedge \bigwedge_{i=1}^{m-1} \ell_i\right) \vdash_1 \bot\right],$$

where $\psi \vdash_1 \ell$ means that ℓ can be derived from ψ by unit propagation.

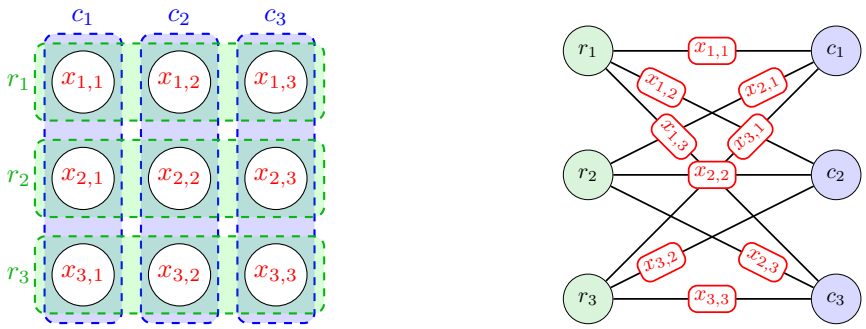
For example, to say that an encoding φ of $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ is propagation complete means that, whenever a variable x_i is assigned to true, all other variables x_j will be assigned to false by unit propagation, and thus if two input variables are ever assigned to true, a contradiction is detected by unit propagation alone. Given that SAT solvers perform unit propagation very efficiently, propagation-complete encodings guarantee that entailments involving the encoded function can be derived efficiently by a solver. We remark that there are several variations of propagation completeness considered in the literature; for example, Bordeaux and Marques-Silva proposed earlier a definition that does not distinguish between input and auxiliary variables [9], which makes for a stronger condition. A weaker condition, *unit-refutation completeness*, requires that "refutations" (i.e., derivations of $\bot$) can be obtained by unit propagation [28].

3 Encodings for AtMostOne

In this section, we prove Theorems 1 and 2. We start by presenting Chen's product encoding for AtMostOne and giving a new graph-theoretic perspective on it, from which our improved encoding will seem more natural.

3.1 Two perspectives on the product encoding

The traditional way to present Chen's product encoding is by arranging the input variables into a grid as depicted in Figure 1a. Then, the key insight underlying the encoding is that at most one input variable is true if and only if (a) at most one row contains a true variable and (b) at most one column contains a true variable. This is encoded as follows. For each



(a) Grid interpretation. **(b)** Bipartite interpretation.

Figure 1 Illustration of our proposed change of perspective on Chen's product encoding.

row, introduce an auxiliary variable r_i that is implied by each variable in that row, and do similarly for the columns with auxiliary variables c_j. Then, one recursively encodes the AtMostOne constraints for $\{r_1, r_2, \dots\}$ and $\{c_1, c_2, \dots\}$.

More formally, rename the input variables $x_1, \dots, x_n$ to be of the form $x_{i,j}$ with $i, j \in [p]$, where $p = \lceil \sqrt{n} \rceil$. Let E be the set of ordered pairs (i, j) to which some variable is assigned. Then, the product encoding is as follows:

$$\mathsf{PE}(\{x_{i,j} \mid (i,j) \in E\}) := \left(\bigwedge_{(i,j)\in E} (\overline{x_{i,j}} \vee r_i) \wedge (\overline{x_{i,j}} \vee c_j) \right) \wedge \mathsf{PE}(r_1, \dots, r_p) \wedge \mathsf{PE}(c_1, \dots, c_p).$$

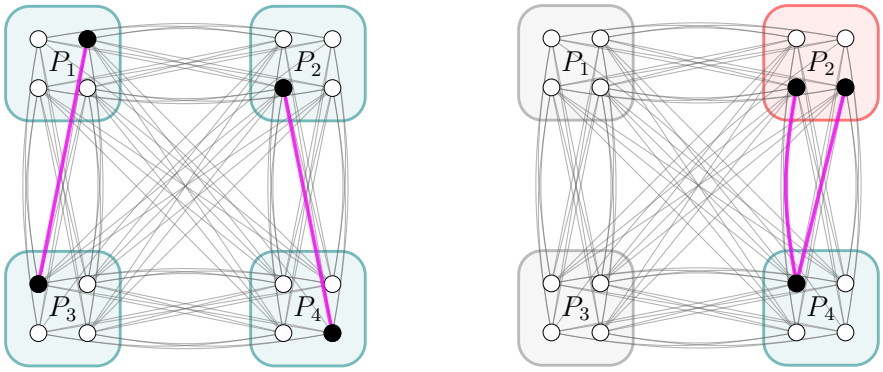
For the base case (say, when $n \leq 4$), we use the direct encoding: $\bigwedge_{1\leq i<j\leq n} (\overline{x_i} \vee \overline{x_j})$.

But there is also another interpretation of the product encoding, illustrated in Figure 1b. Here, we identify the input variables with the edges of a bipartite graph. We say that an edge is *selected* if the corresponding input variable is true, and we say that a vertex is *selected* if it is incident to a selected edge. Now, the key insight can be rephrased as follows: at most one edge is selected if and only if (a) at most one vertex in the left part is selected and (b) at most one vertex in the right part is selected. Of course, the grid interpretation and bipartite interpretation are equivalent, but the bipartite interpretation provides a nice conceptual lens for designing new encodings for AtMostOne.

Given a graph G, our goal is to encode $\mathsf{AtMostOne}(E(G))$, the constraint asserting that at most one edge is selected. Let the input variables be $x_{\{i,j\}}$ for each $\{i, j\} \in E(G)$. As in the product encoding, we introduce an auxiliary variable v_i for each vertex $i \in V(G)$, and we spend $2|E|$ clauses to make each edge imply its endpoints: $\overline{x_{i,j}} \vee v_i$ and $\overline{x_{i,j}} \vee v_j$. Then, it remains to use the auxiliary variables associated with the vertices to encode that at most one edge is selected; how we do this depends on the choice of G. The efficiency of the encoding (i.e., how many clauses it has as a function of $|E|$) depends on the edge density of the graph and how succinctly we can use the vertex variables to encode that at most one edge is selected.

3.2 The multipartite encoding

We now describe an encoding for $\mathsf{AtMostOne}(x_1, \dots, x_n)$ with $2n + 2\sqrt{2n} + O(\sqrt[3]{n})$ clauses and $\sqrt{2n} + O(\sqrt[3]{n})$ auxiliary variables, thus proving Theorem 1. We use the graph-theoretic strategy just described, taking G to be a complete multipartite graph with $\Theta(\sqrt[6]{n})$ parts and $\Theta(\sqrt[3]{n})$ vertices within each part; we therefore call our encoding the *multipartite encoding*.



(a) AtMostTwo-parts constraint is violated. **(b)** AtMostOne′(_; z_2)-vertex constraint is violated.

Figure 2 Illustration of the multipartite encoding. Parts violating the AtMostOne-vertex constraint are shaded red. Parts for which z_k is true are shaded teal.

The constants are chosen so that G has $\sim\sqrt{2n}$ vertices and $\sim n$ edges. These parameters turn out to be a good choice for two reasons. First, G has a high edge density, which allows us to assign more input variables to the edges of G. Second, we have a succinct way to use the vertex variables to encode that at most one edge is selected:

> **Key insight:** At most one edge is selected if and only if (a) at most one vertex from each part is selected and (b) at most two parts contain a selected vertex (see Figure 2).

To materialize this encoding, we first require an intermediate construction, which is a simple modification of Chen's product encoding. Let $\mathsf{AtMostOne}'(x_1, \ldots, x_n; z)$ be the constraint asserting that at most one of the variables $x_1, \ldots, x_n$ is true and that x_i implies z for each $i \in [n]$; that is $\mathsf{AtMostOne}'(x_1, \ldots, x_n; z) \iff \mathsf{AtMostOne}(x_1, \ldots, x_n) \wedge \bigwedge_{i\in[n]}(\overline{x_i} \vee z)$.

▶ **Lemma 7.** *There is a propagation-complete encoding of the* $\mathsf{AtMostOne}'(x_1, \ldots, x_n; z)$ *constraint using* $2n + O(\sqrt{n})$ *clauses and* $O(\sqrt{n})$ *auxiliary variables.*

Proof sketch. It suffices to use Chen's product encoding and add one clause $(\overline{r_i} \vee z)$ per row r_i. Then, each input variable implies z indirectly through its row. ◀

Now we can prove Theorem 1.

Proof of Theorem 1. Let G be the complete p-partite graph with q vertices within each part for $p := \lceil \sqrt[6]{n} \rceil + 1$ and $q := \lceil \sqrt{2} \cdot \sqrt[3]{n} \rceil$. This way, $|E(G)| = \binom{p}{2} q^2 \geq n$. Let $P_1, \ldots, P_p$ be the parts of G. Assign the variables $x_1, \ldots, x_n$ to distinct edges of G, renaming the variables so that $x_{\{i,j\}}$ is the variable assigned to the edge $\{i, j\}$. Let E be the set of edges of G to which some variable is assigned. Discard any vertices of G not incident to an edge from E. Introduce auxiliary variables v_i for each $i \in V(G)$ and z_k for each $k \in [p]$. Our encoding is as follows:

$$\mathsf{ME}(\{x_{\{i,j\}} \mid \{i,j\} \in E\}) := \left(\bigwedge_{\{i,j\}\in E} (\overline{x_{\{i,j\}}} \vee v_i) \wedge (\overline{x_{\{i,j\}}} \vee v_j) \right) \wedge \left(\bigwedge_{k\in[p]} \mathsf{AtMostOne}'(\{v_i \mid i \in P_k\}; z_k) \right) \wedge \mathsf{AtMost}_2(z_1, \ldots, z_p),$$

where we use the generalized product encoding for $\mathsf{AtMost}_2(z_1, \ldots, z_p)$ – described as well in Section 4 – which is propagation complete and uses $3p + O(p^{2/3})$ clauses and $O(p^{2/3})$ auxiliary variables [17].

First, we argue that the encoding is correct and propagation complete. Suppose that at most one input variable is true. If zero input variables are true, then $\mathsf{ME}(\{x_{\{i,j\}} \mid \{i,j\} \in E\})$ is satisfiable by setting all of the v and z auxiliary variables to false. Otherwise, exactly one input variable $x_{\{i,j\}}$ is true. Let k and ℓ be such that $i \in P_k$ and $j \in P_\ell$. Then, $\mathsf{ME}(\{x_{\{i,j\}} \mid \{i,j\} \in E\})$ is satisfiable by setting v_i, v_j, z_k, and z_ℓ to true and all of the other v and z auxiliary variables to false.

It remains to show that $\mathsf{ME}(\{x_{\{i,j\}} \mid \{i,j\} \in E\}) \wedge x_{\{i,j\}} \vdash_1 \overline{x_{\{i',j'\}}}$ for all $\{i',j'\} \in E \setminus \{\{i,j\}\}$. If $x_{\{i,j\}}$ is true, then v_i and v_j can be derived by one step of unit propagation, and since our encodings of $\mathsf{AtMostOne}'(\{v_i \mid i \in P_k\}; z_k)$ and $\mathsf{AtMostOne}'(\{v_i \mid i \in P_\ell\}; z_\ell)$ are propagation complete by Lemma 7, we can derive z_k and z_ℓ by unit propagation (for k and ℓ defined as above), as well as $\overline{v_{i'}}$ for all $i' \in P_k \cup P_\ell \setminus \{i,j\}$. Thus, we can derive $\overline{x_{\{i',j'\}}}$ by unit propagation for every edge $\{i',j'\}$ between P_k and P_ℓ other than $\{i,j\}$. Since our encoding of $\mathsf{AtMost}_2(z_1, \ldots, z_p)$ is propagation complete, $\overline{z_{k'}}$ is derivable by unit propagation for all $k' \notin \{k,\ell\}$, and therefore $\overline{v_{i'}}$ is derivable by unit propagation for all $i' \in P_{k'}$ and all $k' \notin \{k,\ell\}$. Thus, we can derive $\overline{x_{\{i',j'\}}}$ by unit propagation for every edge $\{i',j'\}$ that is not between P_k and P_ℓ. We conclude that the encoding is correct and propagation complete.

Finally, the number of clauses is $2n + p \cdot (2q + O(\sqrt{q})) + (3p + O(p^{2/3})) = 2n + 2\sqrt{2n} + O(\sqrt[3]{n})$, and the number of auxiliary variables is $pq + p \cdot O(\sqrt{q}) + O(p^{2/3}) = \sqrt{2n} + O(\sqrt[3]{n})$. ◀

3.3 The clique encoding

The analysis in the proof of Theorem 1 generalizes as follows: let $f_1(n)$ and $f_2(n)$ be the minimum size of encodings for $\mathsf{AtMostOne}$ and AtMost_2 on n variables, respectively. Then, taking G to be a complete p-partite graph with q vertices within each part yields

$$f_1(n) \leq 2n + p \cdot f_1(q) + f_2(p).$$

For $p, q = \omega(1)$, the best choice is that of Theorem 1. When $p = 2$, we have $f_2(p) = 0$, and since G is a complete bipartite graph, we recover the product encoding. The other extreme is to take $q = 1$, in which case $f_1(q) = 0$, and G is a complete graph; then, the size of the encoding is $2n + f_2(p)$. Using our disjunctive grid compression encoding (Theorem 4) for AtMost_2, we can get an encoding for $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ with slightly fewer clauses than in Theorem 1, at the cost of losing propagation completeness. This is the smallest encoding for $\mathsf{AtMostOne}$ that we know of.

▶ **Theorem 8** (Clique Encoding). *There is an encoding of the $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ constraint using $2n + 2\sqrt{2n} + \widetilde{O}(\sqrt[4]{n})$ clauses and $\sqrt{2n} + \widetilde{O}(\sqrt[4]{n})$ auxiliary variables.*

3.4 A smaller circuit for threshold-2

The search for *small* circuits (i.e., using few gates or wires) for cardinality constraints has been a cornerstone of circuit complexity (see, e.g., [8, 13, 20, 35, 38]). While the product encoding was only proposed in the context of SAT in 2010 [11], the same idea was independently discovered by Adleman in 1976 and first mentioned in print by Bloniarz [8] a few years later. The *threshold-2* function, denoted $T_2(x_1, \ldots, x_n)$, is the negation of $\mathsf{AtMostOne}(x_1, \ldots, x_n)$. Adleman showed that there is a monotone boolean circuit for $T_2(x_1, \ldots, x_n)$ with $2n + 2\sqrt{n} + O(\sqrt[4]{n})$ gates. Our construction from Theorem 1 can naturally be adapted to circuits, yielding the first improvement to Adleman's result:

▶ **Theorem 9.** *There is a monotone boolean circuit for* $T_2(x_1, \dots, x_n)$ *with* $2n + \sqrt{2n} + O(\sqrt[3]{n})$ *gates.*

Sergeev [35] proved that every monotone boolean circuit for $T_2(x_1, \dots, x_n)$ has at least $2n + \sqrt{(2n-4)/3} - 19/6$ gates, so Theorem 9 is almost optimal.

A monotone boolean circuit is said to be *single level* if every path from an input to the output goes through at most one $\wedge$ gate. Interestingly, Sergeev showed that every *single-level* monotone boolean circuit for $T_2(x_1, \dots, x_n)$ has at least $2n + 2\sqrt{n+11} - 10$ gates.[3] Thus, Adleman's construction is essentially optimal for single-level circuits, and a corollary of Theorem 9 is that the smallest monotone boolean circuits for $T_2(x_1, \dots, x_n)$ are not single level. This answers a 47-year-old open question from Bloniarz [8, p. 158], and it should be contrasted with a result of Krichevskii [27] stating that single-level monotone boolean *formulas* are optimal for $T_2(x_1, \dots, x_n)$. It was a long-standing open problem whether there exists a quadratic boolean function (i.e., a disjunction of cubes of the form $x_i \wedge x_j$) whose single-level monotone circuit complexity is strictly greater than its monotone circuit complexity; the negation of this statement was sometimes called the *single-level conjecture*. The problem appears to originate with Bloniarz [8, p. 158] and was further studied by Lenz and Wegener [30] and several other authors (see, e.g., [3, 10, 32]). The conjecture was finally disproved by Jukna [23] using a carefully constructed quadratic boolean function. Thus, our results show that, surprisingly, the conjecture already fails for $T_2(x_1, \dots, x_n)$, the simplest quadratic boolean function of all.

3.5 An unconditional lower bound for AtMostOne

Unconditional lower bounds on CNF encodings are rare (directly related to circuit lower bounds and thus $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ [1]); the only example we are aware of is for the *parity* function, for which Emdin, Kulikov, Mihajlin, and Slezkin [15] showed a $3n - 9$ lower bound, to be compared with the $4n - 6$ known upper bound.

Kučera, Savický, and Vorel [29] proved that every propagation-complete encoding of the $\mathsf{AtMostOne}(x_1, \dots, x_n)$ constraint requires $2n + \sqrt{n} - 2$ clauses for $n \geq 7$, and that this can be improved to $2n + 2\sqrt{n} - 3$ clauses for $n \geq 9$ for 2-CNF encodings. They asked whether a $2n + \Omega(\sqrt{n})$ lower bound holds for *unit refutation complete* encodings of $\mathsf{AtMostOne}(x_1, \dots, x_n)$, where unit refutation completeness is a weaker version of propagation completeness. Theorem 2 provides such a bound with no propagation assumptions. Our lower bound proof is heavily inspired by Kučera, Savický, and Vorel's proof. They define a *regular form* for encodings of $\mathsf{AtMostOne}(x_1, \dots, x_n)$, which is well-suited for theoretical analysis. The proof strategy is two-pronged. On one hand, if the smallest encoding φ of $\mathsf{AtMostOne}(x_1, \dots, x_n)$ is not in regular form, then they show that there is an encoding φ' of $\mathsf{AtMostOne}(x_1, \dots, x_{n-1})$ with three fewer clauses, which allows one to conclude by induction. On the other hand, if an encoding φ is in regular form, then they argue directly that the encoding satisfies their claimed bound. Both prongs of their argument use the propagation completeness assumption. Our contribution is to show that the assumption can be eliminated in both cases. The analysis of encodings in regular form uses a graph-theoretic argument that is inspired by the perspective in Section 3.1.

The lower bound for $\mathsf{AtMostOne}$ easily extends to AtMost_k:

▶ **Corollary 10.** *Every encoding of the* $\mathsf{AtMost}_k(x_1, \dots, x_n)$ *constraint has at least* $2(n-k) + \sqrt{n-k+2}$ *clauses for* $n \geq k + 7$.

[3] In [35], the bound is stated as $2n + 2\sqrt{n+27} - 14$. Sergeev told us that the proof contains a mistake that, when corrected, yields the (improved) bound $2n + 2\sqrt{n+11} - 10$.

Proof. Let φ be an encoding of $\mathsf{AtMost}_k(x_1, \ldots, x_n)$, and let τ be the partial assignment setting $x_1, \ldots, x_{k-1}$ to $\top$. Then, note that $\varphi|_\tau$ encodes $\mathsf{AtMostOne}(x_k, \ldots, x_n)$, and thus by Theorem 2, it must have at least $2(n-k+1) + \sqrt{n-k+2} - 2$ clauses. Since φ has at least as many clauses as $\varphi|_\tau$, this concludes the proof. ◀

4 Disjunctive Switching Yields a $2n + o_k(n)$ Encoding for AtMost_k

Before the present work, the smallest known encodings for $\mathsf{AtMost}_k(x_1, \ldots, x_n)$ were of size $(k+1)n + o(n)$ for $k \leq 5$ [17] and $7n - o(n)$ for $k \geq 6$ [36], and Corollary 10 gives a lower bound of $2n$ for $k = o(n)$. The encoding of size $(k+1)n + o(n)$ is based on a generalization of Chen's product encoding. In this section, we show that the generalized product encoding can be "disjunctivized", yielding the following:

▶ **Theorem 11.** *There is an encoding of the $\mathsf{AtMost}_k(x_1, \ldots, x_n)$ constraint using $2n + O(kn^{k/(k+1)})$ clauses and $O(kn^{k/(k+1)})$ auxiliary variables.*

For $k = o(\log n / \log\log n)$, this encoding uses $2n + o(n)$ clauses and $o(n)$ auxiliary variables.

We begin by introducing the concept of disjunctive switching through an example. Then, we present the generalized product encoding of Frisch and Giannaros [17] and sketch how disjunctive switching allows us to obtain Theorem 11 from it.

4.1 Disjunctive switching

Let us illustrate the main idea behind disjunctive switching through an example. A more abstract perspective is provided in the full version of this paper [26, Appendix F].

Recall that in the grid presentation of Chen's product encoding (Figure 1a), having two input variables $x_{i,j}$ and $x_{i',j'}$ assigned to true will either contradict that at most one row is selected or that at most one column is selected. Based on which of these two constraints is violated, one might in hindsight feel as if one of the two implications (a) $x_{i,j} \to r_i$ (b) $x_{i,j} \to c_j$ was "wasted". Disjunctive switching will use *disjunctive implications* $x_{i,j} \to (r_i \vee c_j)$ and then incorporate some *switching* mechanism to negate whichever variable (i.e., r_i or c_j) would have been "wasted".

We illustrate the technique with the following problem: we have a 3×3 grid of variables $x_{1,1}, \ldots, x_{3,3}$, as illustrated in Figure 3, and a "direction" variable d. We wish to encode that if d is true, then at most one column of the grid contains a true x variable, and that if d is false, then at most one row of the grid contains a true x variable. In other words, variable d corresponds to which direction (horizontal or vertical) will be constrained.

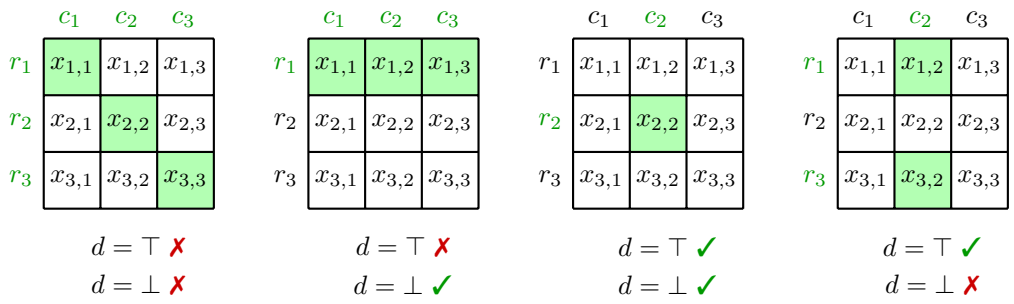


Figure 3 Example for the disjunctive switching technique. True input variables are highlighted in green. The ✓ and ✗ symbols indicate whether the encoding should be satisfiable for the given d.

A naïve encoding would resemble Chen's product encoding; we start with 2 clauses per input variable

$$\bigwedge_{i=1}^{3}\bigwedge_{j=1}^{3}(x_{i,j}\to r_i)\wedge(x_{i,j}\to c_j)\,, \tag{1}$$

and then we enforce $d\to\mathsf{AtMostOne}(c_1,c_2,c_3)$ and $\overline{d}\to\mathsf{AtMostOne}(r_1,r_2,r_3)$, both of which consist of 3 clauses. Thus, this encoding uses a total of $2\cdot 9+2\cdot 3=24$ clauses.

The disjunctive switching paradigm allows us to reduce the number of clauses. We start with the 9 disjunctive clauses

$$\bigwedge_{i=1}^{3}\bigwedge_{j=1}^{3}(x_{i,j}\to(r_i\vee c_j))\equiv\bigwedge_{i=1}^{3}\bigwedge_{j=1}^{3}(\overline{x_{i,j}}\vee r_i\vee c_j)\,, \tag{2}$$

instead of the 18 clauses from constraint (1). Now, we add the AtMostOne constraints

$$(\overline{r_1}\vee\overline{r_2})\wedge(\overline{r_1}\vee\overline{r_3})\wedge(\overline{r_2}\vee\overline{r_3})\wedge(\overline{c_1}\vee\overline{c_2})\wedge(\overline{c_1}\vee\overline{c_3})\wedge(\overline{c_2}\vee\overline{c_3}), \tag{3}$$

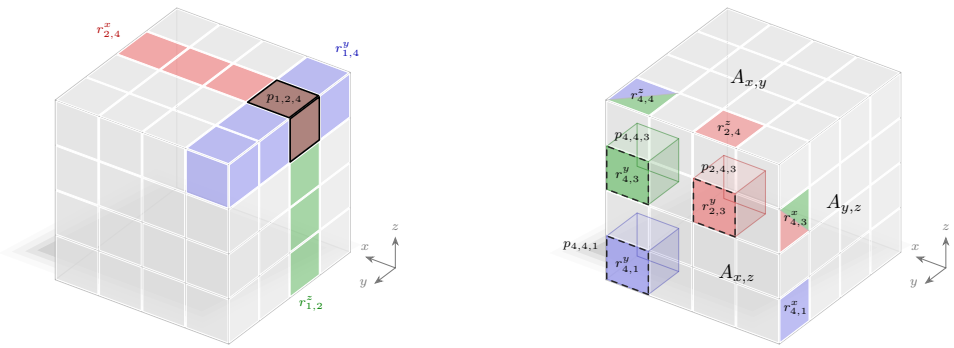
and we complete our encoding by adding the following clauses:

$$(d\to\overline{r_1})\wedge(d\to\overline{r_2})\wedge(d\to\overline{r_3})\wedge(\overline{d}\to\overline{c_1})\wedge(\overline{d}\to\overline{c_2})\wedge(\overline{d}\to\overline{c_3}). \tag{4}$$

Intuitively, if d is true, then it "turns off" all the r_i variables, and thus (i) their AtMostOne constraint in (3) is trivially satisfied, and (ii) the disjunctive implications in constraint (2) reduce to $x_{i,j}\to c_j$, and thus (3) enforces an at-most-one-column constraint. Naturally, for the other direction, $\overline{d}$ "turns off" the c_j variables through (4), and thus (2) together with (3) end up enforcing an at-most-one-row constraint. This disjunctive-switching encoding uses $9+6+6=21$ clauses, saving 3 from the naïve one. For an $n\times n$ grid, the naïve encoding would have $2n^2+O(n)$ clauses, whereas the disjunctive-switching one would only have $n^2+O(n)$. Next, we will see that the same principle used in this encoding can serve as the basis of a compact encoding for AtMost_k.

4.2 The generalized product encoding

Frisch and Giannaros proposed a nice generalization of Chen's product encoding [17]; rather than imagining the input variables in a two-dimensional grid as in Figure 1a, we instead imagine them in a $(k+1)$-dimensional grid, and use the higher-dimensional analogues of rows and columns to recursively encode AtMost_k. Let us detail the particular case of $k=2$.



(a) Rods implied by $p_{1,2,4}$: $r^x_{2,4}$, $r^y_{1,4}$, and $r^z_{1,2}$.

(b) Three input variables and rods visible from different faces.

Figure 4 Example of the generalized product encoding for $k=2$.

For $k = 2$, the input variables $p_{x,y,z}$ take place in a three-dimensional grid (for n variables, it will be a $\lceil \sqrt[3]{n} \rceil \times \lceil \sqrt[3]{n} \rceil \times \lceil \sqrt[3]{n} \rceil$ grid), and each variable implies the three "rods" it belongs to, as e.g., $(p_{1,2,4} \to r^x_{2,4}) \wedge (p_{1,2,4} \to r^y_{1,4}) \wedge (p_{1,2,4} \to r^z_{1,2})$, depicted in Figure 4a.

Then, we let $A_{y,z} = \left\{ r^x_{y,z} \mid 1 \leq y, z \leq \lceil \sqrt[3]{n} \rceil \right\}$ be the rods visible from the yz face of the grid, and similarly let $A_{x,z}$ and $A_{x,y}$ be the rods visible from the xz and xy faces. The encoding is then completed by recursively enforcing that from every face at most two rods are visible: $\mathsf{AtMost}_2(A_{y,z}) \wedge \mathsf{AtMost}_2(A_{x,z}) \wedge \mathsf{AtMost}_2(A_{x,y})$.

Clearly, if at most two input variables are true, then the encoding is satisfiable. Less obviously, if at least three input variables are true, then there will be some face yz, xz, or xy from which at least three implied rods are visible, making the encoding unsatisfiable. This is directly analogous to the key insight justifying the product encoding, namely that if at least two input variables are true, then there is some *side* (i.e., x-axis or y-axis) from which at least two implied rows/columns are visible.

Note that there are $3n$ *projection* clauses, each stating that an input variable $p_{x,y,z}$ implies a rod variable. Moreover, there are $O(n^{2/3})$ rods $A_{y,z}$, $A_{x,z}$, and $A_{x,y}$, from where the recursive AtMost_2 constraints use only $O(n^{2/3})$ clauses. Hence, the encoding is of size $3n + O(n^{2/3})$.

4.3 The disjunctive generalized product encoding

The bottleneck of the generalized product encoding is the projection clauses, which consist of $3n$ clauses for $k = 2$. Using disjunctive switching, we can reduce this burden to only $2n$ clauses, yielding a $2n + o(n)$ encoding for fixed k. Here, we describe the encoding for $k = 2$, leaving a full description to the arXiv version [26].

Recall that the correctness of the previous encoding relied on the following fact: if at least three input variables are true, then there is some face yz, xz, or xy from which at least three implied rods are visible. Call such a face a *witnessing* face. If we knew a priori which face is witnessing, then we could readily apply disjunctive switching to encode the projections using only n clauses. Our main insight for this encoding is that, while we do not know which face is witnessing a priori, we can determine it after projecting onto the first face yz. For example, in Figure 4b, the visible rods from the yz-face are $r^x_{4,1}$ and $r^x_{4,3}$, and since these rods share a y-coordinate, projecting onto the xy-face is a bad idea: the $r^x_{4,1}$ and $r^x_{4,3}$ variables do not carry information of the x-coordinates of the input variables implying them, so if those x-coordinates turned out to be equal (as is the case for $p_{4,4,1}$ and $p_{4,4,3}$ in Figure 4b), their xy-projections would coincide ($r^z_{4,4}$ in Figure 4b). Therefore, we conclude that xz must be the witnessing face if there is one.

We are now ready for a formal description of the encoding. First, we include projection clauses, direct ones for the yz-face and "disjunctivized" ones for the other two:

$$\bigwedge_{x,y,z \in [\lceil \sqrt[3]{n} \rceil]} \left(\overline{p_{x,y,z}} \vee r^x_{y,z}\right) \wedge \bigwedge_{x,y,z \in [\lceil \sqrt[3]{n} \rceil]} \left(\overline{p_{x,y,z}} \vee r^y_{x,z} \vee r^z_{x,y}\right). \tag{5}$$

Then, we impose the following constraints on the xz- and xy-faces: $\mathsf{AtMost}_2(A_{x,z}) \wedge \mathsf{AtMost}_2(A_{x,y})$; it will turn out to be superfluous to impose $\mathsf{AtMost}_2(A_{y,z})$.

Next, we introduce an auxiliary variable w, whose intended semantics is that xz is a witnessing face (if there is one). By the discussion above, if there are two visible rods r^x_{y,z_1} and r^x_{y,z_2}, then w must be true. Thus, we include the following constraint:

$$\bigwedge_{y \in [\lceil \sqrt[3]{n} \rceil]} \left(w \vee \mathsf{AtMostOne}\left(\left\{ r^x_{y,z} \mid z \in \left[\lceil \sqrt[3]{n} \rceil\right] \right\}\right)\right),$$

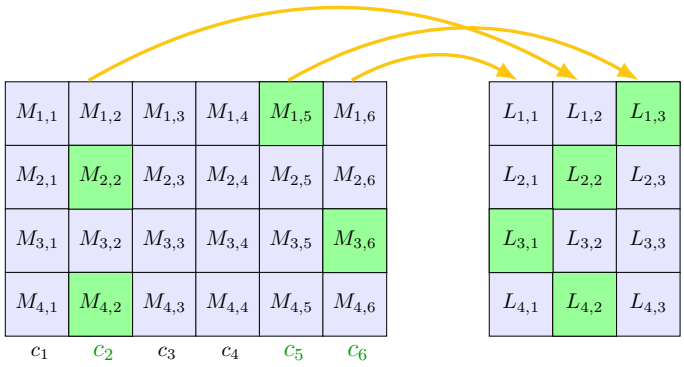


Figure 5 Illustration of the grid compression framework.

and analogously for the symmetric case,

$$\bigwedge_{z\in[\lceil\sqrt[3]{n}\rceil]} \left(\overline{w} \vee \mathsf{AtMostOne}\left(\{r^x_{y,z} \mid y \in [\lceil\sqrt[3]{n}\rceil]\}\right)\right).$$

Finally, based on w, we constrain the projection variables for the xz- and xy-faces (cf. (4)):

$$\bigwedge_{x,y\in[\lceil\sqrt[3]{n}\rceil]} \left(\overline{r^z_{x,y}} \vee \overline{w}\right) \wedge \bigwedge_{x,z\in[\lceil\sqrt[3]{n}\rceil]} \left(\overline{r^y_{x,z}} \vee w\right).$$

This completes the description of the disjunctive generalized product encoding for $k = 2$. There are $2n$ projection clauses in (5), and all other constraints use $O(n^{2/3})$ clauses. Thus, the total encoding size is $2n + O(n^{2/3})$.

5 Grid Compression Encodings for AtMost_k

The disjunctive generalized product encoding for AtMost_k has $2n + o(n)$ clauses for $k = o(\log n/\log\log n)$. In this section, we introduce a technique called *grid compression*, which allows us to give an encoding for AtMost_k with $2n + o(n)$ clauses for $k = o(\sqrt[3]{n})$. We start by describing an encoding with $4n + o(n)$ clauses for any $k = o(n)$. Then, we show how a "disjunctivized" version of this yields a $2n + o(n)$ encoding for $k = o(\sqrt[3]{n})$.

5.1 General framework

We arrange the input variables $x_1, \ldots, x_n$ in a $\lceil n/m\rceil \times m$ grid M, where m is a parameter satisfying $k = o(m)$ and $m = o(n)$ that will be specified later for each encoding. If at most k input variables are set to true, then at most k columns of the grid contain input variables set to true (call these columns *occupied*). Grid compression works by *compressing* M into a $\lceil n/m\rceil \times \ell$ grid L – where ℓ is a parameter satisfying $k = o(\ell)$ and $\ell = o(m)$ – which will contain a copy of each occupied column in M, although potentially in a different order (see Figure 5). To that end, we will have auxiliary variables $L_{i,j}$ for $i \in [\lceil n/m\rceil]$ and $j \in [\ell]$.

To implement grid compression, we will employ intermediate constructions that assign the indices of occupied columns in M to indices of columns in L. Then, if column index j of M is mapped to column index p of L, we will have another set of constraints that ensure that $M_{:,j} = L_{:,p}$. We then enforce

$$\mathsf{AtMost}_k(\{L_{i,j} \mid (i,j) \in [\lceil n/m\rceil] \times [\ell]\}) \tag{6}$$

using $O(\lceil n/m\rceil \times \ell) = o(n)$ clauses through, e.g., the parallel counter encoding [36].

For each column $j \in [m]$, we will have an auxiliary variable c_j representing whether that column of M is occupied. Correspondingly, we will add clauses of the form

$$\bigwedge_{(i,j)\in[\lceil n/m\rceil]\times[m]} (\overline{M_{i,j}} \vee c_j). \tag{7}$$

While both encodings we present next include constraints (6) and (7), the specific implementation of the copying from M to L depends on which variant of the encoding we employ. The commonality is that, for each $j \in [m]$, we associate some *small* set $H_j \subset [\ell]$, which will represent the set of columns in L that *may* store a copy of the jth column of M. At a high level, our encodings need to use small sets H_j to save clauses, and yet these small sets need to be chosen carefully so that it is still possible to choose a different target column from each H_j such that column j is occupied; this is a traditional problem in hashing, for which our two encodings use different solutions.

5.2 Grid compression encoding

We set $m = \Theta(\sqrt[3]{kn^2})$ and $\ell = \Theta(\sqrt[3]{k^2 n})$ for suitable constants. We will have $|H_j| = 3$ for every $j \in [m]$, and the construction of these sets will be specified later. We introduce auxiliary variables $\mathsf{copy}_{j,p}$, representing that $M_{:,j}$ is to be copied to $L_{:,p}$, whose semantics are enforced by the following clauses:

$$\bigwedge_{(i,j)\in[\lceil n/m\rceil]\times[m]} \bigwedge_{p\in H_j} \left(\overline{M_{i,j}} \vee \overline{\mathsf{copy}_{j,p}} \vee L_{i,p}\right). \tag{8}$$

We next enforce that each occupied column j of M is copied to some column p of L with $p \in H_j$:

$$\bigwedge_{j\in[m]} \left(\overline{c_j} \vee \bigvee_{p\in H_j} \mathsf{copy}_{j,p}\right). \tag{9}$$

Finally, we enforce that no two columns of M can be copied to the same column of L:

$$\bigwedge_{p\in[\ell]} \mathsf{AtMostOne}(\{\mathsf{copy}_{j,p} \mid j \in [m] \text{ such that } p \in H_j\}). \tag{10}$$

To make the encoding correct, we must choose our collection of sets $\mathcal{H} := \{H_j \mid j \in [m]\}$ in such a way that every subset $\mathcal{F} \subset \mathcal{H}$ of size at most k has a *transversal*, i.e., it is possible for each $F \in \mathcal{F}$ to pick a distinct point $p_F \in F$. We show that such a set $\mathcal{H}$ exists by a probabilistic argument based on Hall's marriage theorem.

▶ **Lemma 12.** *We can choose $m = \Theta(\sqrt[3]{kn^2})$ and $\ell = \Theta(\sqrt[3]{k^2 n})$ such that there exists a set $\mathcal{H}$ such that (a) $|\mathcal{H}| = m$, (b) $H_i \in \binom{[\ell]}{3}$ for each $H_i \in \mathcal{H}$, and (c) any subset $\mathcal{F} \subset \mathcal{H}$ of size at most k admits a transversal.*

This completes the description of the grid compression encoding; both correctness and a detailed count of the clauses are proved in the extended version [26], but in summary, constraint (7) uses n clauses, constraint (8) uses $3n$ clauses, and the remaining constraints all use $O(\sqrt[3]{kn^2})$ clauses. This establishes Theorem 3.

5.3 Disjunctive grid compression

Now we set $m = \Theta\left(\sqrt{nk \log_k n}\right)$ and $\ell = \Theta\left(k^2 \log_k^2 n\right)$ for suitable choices of constants, and we will ensure that the H_i are distinct and each of size $\Theta(k \log_k n)$ for a suitable choice of constant. In this encoding, the disjunctive switching manifests as the following set of clauses enforcing that each true variable in $M_{:,j}$ is copied to $L_{:,p}$ for *some* column $p \in H_j$:

$$\bigwedge_{(i,j)\in[\lceil n/m\rceil]\times[m]} \left(\overline{M_{i,j}} \vee \bigvee_{p\in H_j} L_{i,p}\right). \tag{11}$$

This constraint, however, could allow true variables in different columns of M to be copied to the same variable in L, and thus we need some mechanism to prevent this. To that end, we introduce auxiliary variables ov_p for each $p \in [\ell]$, which intuitively represent whether column p of L is *overloaded*, meaning that there are multiple occupied columns of M that could be copied to p. The semantics for ov_p are given by the following constraints:

$$\bigwedge_{p\in[\ell]} \mathsf{ov}_p \vee \mathsf{AtMostOne}(\{c_j \mid j \in [m] \text{ such that } p \in H_j\}). \tag{12}$$

Finally, to make the disjunctive switching work, we enforce that if we copy into some $L_{i,p}$, then column p must not be overloaded:

$$\bigwedge_{i\in[\lceil n/m\rceil]} \bigwedge_{p\in[\ell]} \left(\overline{L_{i,p}} \vee \overline{\mathsf{ov}_p}\right). \tag{13}$$

These are all of the constraints for the disjunctive grid compression encoding, and it remains to specify $\mathcal{H} := \{H_j \mid j \in [m]\}$. For the encoding to be correct, we require the following: if we pick any k sets $H_{j_1}, \ldots, H_{j_k} \in \mathcal{H}$, then $H_{j_1} \not\subseteq H_{j_2} \cup \cdots \cup H_{j_k}$. This so-called $(k-1)$-*cover-free property* has been studied by numerous authors [16, 21, 25]. We prove the existence of a $(k-1)$-cover-free family $\mathcal{H}$ that satisfies our requirements using an elegant algebraic construction from [16]:

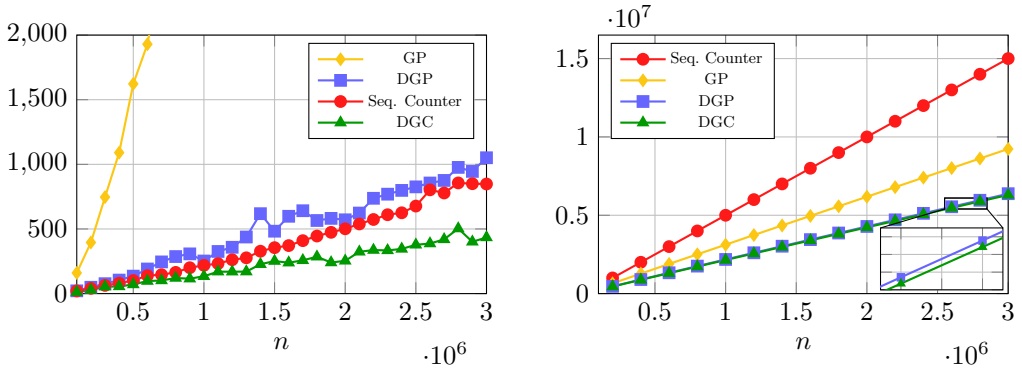
▶ **Lemma 13.** *There exists $q = \Theta(k \log_k n)$ such that there exists a $(k-1)$-cover-free set $\mathcal{H}$ satisfying $H_i \in \binom{[\ell]}{q}$ for each $H_i \in \mathcal{H}$ and $|\mathcal{H}| \geq m = \Theta\left(\sqrt{nk \log_k n}\right)$.*

With $\mathcal{H}$ chosen according to the lemma, the encoding is complete. In contrast to the probabilistic proof of Lemma 12, the construction in Lemma 13 is deterministic and efficient.

6 Concluding Remarks and Empirical Evaluation

We solved a fundamental problem in the theory of CNF encodings by showing that the minimum number of clauses in an encoding of $\mathsf{AtMost}_k(x_1, \ldots, x_n)$ is $2n + \widetilde{\Theta}(\sqrt{n})$ for each fixed k. We also tightened the upper bound on the minimum number of clauses in an encoding of $\mathsf{AtMostOne}$, refuting a conjecture of Chen [11] and resolving a long-standing open problem in circuit complexity [8, p. 158]. En route to these results, we introduced (a) a graph-theoretic framework for constructing and analyzing $\mathsf{AtMostOne}$ encodings and (b) disjunctive switching, a general technique for compactly encoding switch statements into CNF, and (c) grid compression, a framework for constructing CNF encodings of boolean functions supported on inputs of small Hamming weight.

Our constructions are structurally very different from other encodings in the literature; for instance, the multipartite encoding for $\mathsf{AtMostOne}$ is notable for using clauses of width 3, despite the fact that $\mathsf{AtMostOne}$ is a 2-CNF function. Kučera, Savický, and Vorel [29] asked



(a) Runtime (ms). **(b)** Number of clauses.

Figure 6 Comparison of encodings for AtMost_2 (UNSAT). We abbreviate the generalized product encoding as GP and its disjunctive variant as DGP.

whether the smallest propagation-complete encoding of an antitone 2-CNF function is always 2-CNF. While an affirmative answer has some *prima facie* plausibility, our construction concretely demonstrates how wide clauses can be useful even for $\mathsf{AtMostOne}$, the simplest antitone 2-CNF function. In fact, we conjecture that Chen's product encoding is essentially optimal for 2-CNF encodings, in the sense that every 2-CNF encoding for $\mathsf{AtMostOne}$ has at least $2n + 4\sqrt{n} - o(\sqrt{n})$ clauses.

Our encodings from Theorems 4 and 11 are also unique for employing disjunctive switching, which has not been used in any previous CNF encodings to the best of our knowledge. Notably, they allow us to surpass the $3n$ lower bound for monotone circuits [35]. Thus, although previous researchers have noted the close connection between CNF encodings for AtMost_k and circuits for threshold functions (see, e.g., [15, 29, 35]), our results demonstrate the importance of viewing CNF encodings as constituting an important model of computation in their own right. We expect disjunctive switching, and the innovative use of wide clauses more generally, to be important for harnessing the full power of CNF encodings in other contexts as well.

The above considerations demonstrate how rich the theory of CNF encodings is, even for very simple boolean functions. Given the importance of CNF encodings to SAT solving, and the fact that "not much is known about CNF encodings from a theoretical point of view" [15], we expect the further development of this theory to be of great practical significance.

Empirical results. Even though propagation completeness is often described as an essential requirement for practical encodings of cardinality constraints [24, p. 95], exploratory experiments with our encodings show they can be competitive in practice – especially the disjunctive grid compression (DGC) encoding. Figure 6 depicts results on a family of instances in which one would expect propagation completeness to play a role; 3 random disjoint subsets of 10 variables (out of n) are chosen, and we add clauses enforcing that at least one variable in each subset is true. Then, a global AtMost_2 constraint makes the instance unsatisfiable. We compared against all cardinality encodings present in PySAT [22], of which the *sequential counter* [36] performed best. The number of clauses does *not* reliably predict which encoding performed best, and yet this proxy led us to develop encodings that perform well in practice. The full version of this paper includes a more detailed experimental evaluation [26, Appendix G], but in a nutshell, we believe our results challenge the folklore claim of propagation completeness being absolutely necessary.

References

1 Scott Aaronson. $P \stackrel{?}{=} NP$. In John Forbes Nash, Jr. and Michael Th. Rassias, editors, *Open Problems in Mathematics*, pages 1–122. Springer International Publishing, 2016. doi:10.1007/978-3-319-32162-2_1.

2 Miklós Ajtai, János Komlós, and Endre Szemerédi. Sorting in $c \log n$ parallel steps. *Combinatorica*, 3(1):1–19, 1983. doi:10.1007/BF02579338.

3 Kazuyuki Amano and Akira Maruoka. The monotone circuit complexity of quadratic boolean functions. *Algorithmica*, 46(1):3–14, 2006. doi:10.1007/s00453-006-0073-0.

4 Carlos Ansótegui and Felip Manyà. Mapping problems with finite-domain variables to problems with boolean variables. In Holger H. Hoos and David G. Mitchell, editors, *Theory and Applications of Satisfiability Testing*, pages 1–15. Springer Berlin Heidelberg, 2005.

5 Roberto Asín, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell. Cardinality networks: a theoretical and empirical study. *Constraints*, 16(2):195–221, 2011. doi:10.1007/S10601-010-9105-0.

6 Olivier Bailleux and Yacine Boufkhad. Efficient CNF encoding of boolean cardinality constraints. In Francesca Rossi, editor, *Principles and Practice of Constraint Programming - CP*, volume 2833 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2003. doi:10.1007/978-3-540-45193-8_8.

7 Yael Ben-Haim, Alexander Ivrii, Oded Margalit, and Arie Matsliah. Perfect hashing and CNF encodings of cardinality constraints. In Alessandro Cimatti and Roberto Sebastiani, editors, *Theory and Applications of Satisfiability Testing - SAT*, volume 7317 of *Lecture Notes in Computer Science*, pages 397–409. Springer, 2012. doi:10.1007/978-3-642-31612-8_30.

8 Peter Anthony Bloniarz. *The complexity of monotone boolean functions and an algorithm for finding shortest paths on a graph.* PhD thesis, Massachusetts Institute of Technology, 1979.

9 Lucas Bordeaux and Joao Marques-Silva. Knowledge Compilation with Empowerment. In Mária Bieliková, Gerhard Friedrich, Georg Gottlob, Stefan Katzenbeisser, and György Turán, editors, *SOFSEM 2012: Theory and Practice of Computer Science*, pages 612–624. Springer, 2012. doi:10.1007/978-3-642-27660-6_50.

10 Siegfried Bublitz. Decomposition of graphs and monotone formula size of homogeneous functions. *Acta Inform.*, 23(6):689–696, 1986. doi:10.1007/BF00264314.

11 Jingchao Chen. A new SAT encoding of the at-most-one constraint. In *Proc. of the Tenth Int. Workshop of Constraint Modelling and Reformulation*, 2010. URL: https://pierre-flener.github.io/research/ModRef10/papers/Jingchao%20Chen.%20A%20New%20SAT%20Encoding%20of%20the%20At-Most-One%20Constraint%20-%20ModRef%202010.pdf.

12 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms.* MIT Press, Cambridge, MA, third edition, 2009.

13 Paul E. Dunne. *Techniques for the analysis of monotone Boolean networks.* PhD thesis, University of Warwick, 1984.

14 Niklas Eén and Niklas Sörensson. Translating pseudo-boolean constraints into SAT. *J. Satisf. Boolean Model. Comput.*, 2(1-4):1–26, 2006. doi:10.3233/SAT190014.

15 Gregory Emdin, Alexander S. Kulikov, Ivan Mihajlin, and Nikita Slezkin. CNF encodings of symmetric functions. *Theory Comput. Syst.*, 68(5):1291–1311, 2024. doi:10.1007/S00224-024-10168-W.

16 Paul Erdős, Peter Frankl, and Zoltán Füredi. Families of finite sets in which no set is covered by the union of r others. *Israel J. Math.*, 51(1-2):79–89, 1985. doi:10.1007/BF02772959.

17 Alan M. Frisch and Paul A. Giannaros. SAT encodings of the at-most-k constraint: Some old, some new, some fast, some slow. In *Proc. of the Tenth Int. Workshop of Constraint Modelling and Reformulation*, 2010.

18 Ian P. Gent. Arc consistency in SAT. In Frank van Harmelen, editor, *Proceedings of the 15th European Conference on Artificial Intelligence, ECAI'2002, Lyon, France, July 2002*, pages 121–125. IOS Press, 2002.

19 Ian P. Gent and Peter Nightingale. A new encoding of AllDifferent into SAT. In *International Workshop on Modelling and Reformulating Constraint Satisfaction*, volume 3, pages 95–110, 2004.

20 Mikhail I. Grinchuk. On the monotone complexity of threshold functions. *Metody Diskret. Anal.*, 52:41–48, 120, 1992.

21 Thaís Bardini Idalino and Lucia Moura. *A Survey of Cover-Free Families: Constructions, Applications, and Generalizations*, pages 195–239. Springer Nature Switzerland, 2023. doi:10.1007/978-3-031-48679-1_11.

22 Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. PySAT: A Python toolkit for prototyping with SAT oracles. In *SAT*, pages 428–437, 2018. doi:10.1007/978-3-319-94144-8_26.

23 Stasys Jukna. Disproving the single level conjecture. *SIAM J. Comput.*, 36(1):83–98, 2006. doi:10.1137/S0097539705447001.

24 Michał Karpiński. *CNF Encodings of Cardinality Constraints Based on Comparator Networks*. PhD thesis, University of Wrocław, 2019. arXiv:1911.00586.

25 William Kautz and Richard Singleton. Nonrandom binary superimposed codes. *IEEE Transactions on Information Theory*, 10(4):363–377, 1964. doi:10.1109/TIT.1964.1053689.

26 Andrew Krapivin, Benjamin Przybocki, and Bernardo Subercaseaux. Near-optimal encodings of cardinality constraints, 2026. doi:10.48550/arXiv.2603.28954.

27 Rafail E. Krichevskii. Complexity of contact circuits realizing a function of logical algebra. *Sov. Phys. Dokl.*, 8:770–772, 1964.

28 Petr Kučera and Petr Savický. Bounds on the size of PC and URC formulas. *Journal of Artificial Intelligence Research*, 69:1395–1420, 2020. URL: http://dx.doi.org/10.1613/jair.1.12006, doi:10.1613/jair.1.12006.

29 Petr Kučera, Petr Savický, and Vojtěch Vorel. A lower bound on CNF encodings of the at-most-one constraint. *Theor. Comput. Sci.*, 762:51–73, 2019. doi:10.1016/J.TCS.2018.09.003.

30 Katja Lenz and Ingo Wegener. The conjunctive complexity of quadratic boolean functions. *Theoret. Comput. Sci.*, 81(2):257–268, 1991. doi:10.1016/0304-3975(91)90194-7.

31 João Marques-Silva and Inês Lynce. Towards robust CNF encodings of cardinality constraints. In Christian Bessiere, editor, *Principles and Practice of Constraint Programming - CP*, volume 4741 of *Lecture Notes in Computer Science*, pages 483–497. Springer, 2007. doi:10.1007/978-3-540-74970-7_35.

32 Roland Mirwald and Claus P. Schnorr. The multiplicative complexity of quadratic boolean forms. *Theoret. Comput. Sci.*, 102(2):307–328, 1992. doi:10.1016/0304-3975(92)90235-8.

33 Van-Hau Nguyen, Van-Quyet Nguyen, Kyungbaek Kim, and Pedro Barahona. Empirical study on SAT-encodings of the at-most-one constraint. In *SMA 2020: The 9th International Conference on Smart Media and Applications, Jeju, Republic of Korea, September 17 - 19, 2020*, pages 470–475. ACM, 2020. doi:10.1145/3426020.3426170.

34 Joseph E. Reeves. *Cardinality Constraints in Boolean Satisfiability Solving*. PhD thesis, Carnegie Mellon University, 2025.

35 Igor S. Sergeev. On the complexity of monotone circuits for threshold symmetric boolean functions. *Diskret. Mat.*, 32(1):81–109, 2020. doi:10.4213/dm1547.

36 Carsten Sinz. Towards an optimal CNF encoding of boolean cardinality constraints. In Peter van Beek, editor, *Principles and Practice of Constraint Programming - CP*, volume 3709 of *Lecture Notes in Computer Science*, pages 827–831. Springer, 2005. doi:10.1007/11564751_73.

37 Joost P. Warners. A linear-time transformation of linear inequalities into conjunctive normal form. *Information Processing Letters*, 68(2):63–69, October 1998. doi:10.1016/s0020-0190(98)00144-6.

38 Ingo Wegener. *The complexity of Boolean functions*. John Wiley & Sons, Inc., 1987.