Tighter bounds on CNF encodings for cardinality constraints

Andrew Krapivin
Carnegie Mellon University, USA

Benjamin Przybocki
Carnegie Mellon University, USA

Bernardo Subercaseaux
Carnegie Mellon University, USA

Abstract

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. Previously, the best known encoding was Chen's product encoding, which uses $2n + 4\sqrt{n} + O(\sqrt[4]{n})$ clauses and was conjectured by Chen to be optimal. Our construction also yields a smaller monotone circuit for the threshold-2 function, improving on a 50-year-old construction of Adleman and resolving a long-standing open problem in circuit complexity. On the other hand, we prove that any CNF encoding of $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ requires at least $2n$ clauses for $n \geq 6$, which is the first nontrivial unconditional lower bound for this problem. Finally, we give a CNF encoding of $\mathsf{AtMostk}(x_1, \ldots, x_n)$ using $2n + o(n)$ clauses when $k = o(\log n / \log\log n)$, which improves upon an encoding using $7n - 3\lfloor \log n \rfloor - 6$ clauses due to Sinz.

2012 ACM Subject Classification Replace ccsdesc macro with valid one

Keywords and phrases SAT Encodings, Cardinality Constraints

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

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

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., [4, 26, 21, 10, 14, 3, 6, 19, 23, 13, 24]). Additionally, analogous problems have been studied largely independently in the context of circuit complexity (see, e.g., [7, 11, 27, 16, 25]).

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. This can of course be encoded using $\binom{n}{2}$ clauses:

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

For large n, this quadratic blowup in the number of clauses is undesirable. Fortunately, by introducing auxiliary variables, there are several encodings for $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ using only $O(n)$ clauses, such as the sequential counter encoding from [26]. A compact encoding is crucial when applying SAT solvers to problems with $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ constraints for large n [23]. Which encoding is the most performant in practice depends on a number of

© Andrew Krapivin, Benjamin Przybocki, and Bernardo Subercaseaux;
licensed under Creative Commons License CC-BY 4.0

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:13
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

factors, 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 *arc consistent* [15], meaning that the entailments $\mathsf{AtMostOne}(x_1, \ldots, x_n) \wedge x_i \models \overline{x_j}$ can be derived by unit propagation for all $i \neq j$.[1] Bernardo: here we should probably just say something more general like "propagation properties" and then discuss the levels for this, since there are a few related definitions Ben: I don't think we discuss any encodings that only satisfy the intermediate conditions (like consistent but not arc consistent). But, if you prefer to define something like this in a preliminaries section rather than here, feel free to make that change.

Prior to the present work, the smallest known encoding for $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ was the product encoding [10], which uses $2n+4\sqrt{n}+O(\sqrt[4]{n})$ clauses and $2\sqrt{n}+O(\sqrt[4]{n})$ auxiliary variables and is also arc consistent. In the same paper in which he proposed it, Chen conjectured that this encoding is optimal with respect to the number of clauses. Our first contribution is to refute this conjecture:

▶ **Theorem 1.** *There is an arc-consistent 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.*

Notice that our encoding improves the product encoding with respect to both the number of clauses and the number of auxiliary variables.

With regard to lower bounds, Kučera, Savický, and Vorel [19] proved that every arc-consistent 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 arc consistency. Our second contribution provides such a bound:

▶ **Theorem 2.** *Every encoding of the* $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ *constraint has at least* $2n$ *clauses for* $n \geq 6$.

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 asymptotic to $2n$.

While the product encoding was only proposed in the context of SAT in 2010, the same idea was independently discovered by Adleman in 1976 and first mentioned in print by Bloniarz [7] a few years later. Let the *threshold-2* function, denoted $T_2(x_1, \ldots, x_n)$, be 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. Despite being revisited and generalized several times [11, 27, 16, 25], this fundamental result in circuit complexity has not been improved in the 50 years since it was discovered. Our construction from Theorem 1 can naturally be adapted to circuits, yielding the first improvement to Adleman's result:

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

Sergeev [25] proved that every monotone boolean circuit for $T_2(x_1, \ldots, x_n)$ has at least $2n + \sqrt{(2n-4)/3} - 19/6$ gates, so Theorem 3 is almost optimal.

Say that a monotone boolean circuit is *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 $T_2(x_1, \ldots, x_n)$ has at least $2n + 2\sqrt{n+11} - 10$ gates. Thus,

[1] The same property is called *propagation completeness* in [19].

Adleman's construction is essentially optimal for single-level circuits, and a corollary of Theorem 3 is that the smallest monotone boolean circuits for $T_2(x_1, \ldots, x_n)$ are not single level. This answers a 47-year-old open question from Bloniarz [7, p. 158]. This should be contrasted with a result of Krichevskii [18] that single-level monotone boolean *formulas* are optimal for $T_2(x_1, \ldots, x_n)$. 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 [7, p. 158] and was further studied by Lenz and Wegener [20] and several other authors (see, e.g., [9, 22, 2]). The conjecture was finally disproved by Jukna [17] using a carefully constructed quadratic boolean function. It may therefore be surprising that the conjecture already fails for $T_2(x_1, \ldots, x_n)$, the simplest quadratic boolean function of all.

Finally, we turn to the constraint $\mathsf{AtMostk}(x_1, \ldots, x_n)$, which asserts that at most k of the boolean variables $x_1, \ldots, x_n$ are true. Sinz [26] 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 $\mathsf{AtMostk}(x_1, \ldots, x_n)$ as measured by the number of clauses. Our third contribution is a smaller encoding when k is small relative to n:

▶ **Theorem 4.** *There is an encoding of the $\mathsf{AtMostk}(x_1, \ldots, x_n)$ constraint using $2n + O(kn^{k/(k+1)})$ clauses and $O(kn^{k/(k+1)})$ auxiliary variables.*

In particular, for $k = o(\log n / \log \log n)$, our encoding uses $2n + o(n)$ clauses and $o(n)$ auxiliary variables. Unfortunately, neither our encoding nor Sinz's encoding is arc consistent; for this constraint, arc consistency means that the entailments $\mathsf{AtMostk}(x_1, \ldots, x_n) \wedge \bigwedge_{i \in S} x_i \models \overline{x_j}$ can be derived by unit propagation for all subsets $S \subseteq [n]$ of size k and $j \notin S$. The smallest known arc-consistent encoding when k is a small constant is the generalized product encoding [14], which uses $(k+1)n + O(k^2 n^{k/(k+1)})$ clauses and $O(kn^{k/(k+1)})$ auxiliary variables when $k = o(\log n / \log \log n)$. For large k, the smallest known arc-consistent encoding is based on the AKS sorting network [1] and a CNF encoding of sorting networks [12, 3], and it uses $O(n \log k)$ clauses and auxiliary variables.[2]

2 Preliminaries

define what an encoding is, and maybe arc consistency

3 A smaller encoding for AtMostOne

In this section, we prove Theorem 1. We start by presenting Chen's product encoding for $\mathsf{AtMostOne}$ 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 the encoding is by arranging the input variables into a grid as in Figure 1. The key insight underlying the encoding is that at most one input variable is

[2] In practice, the AKS sorting network is not used because the constant factors are too large. A practical alternative is Batcher's sorting network [5], although this raises the complexity to $O(n \log^2 k)$ clauses and auxiliary variables.

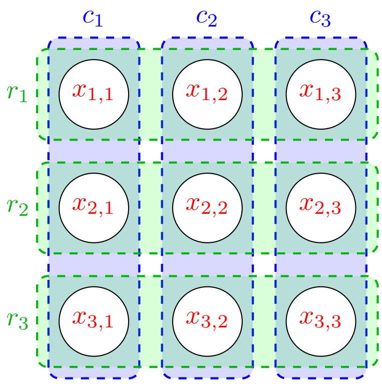


Figure 1 Grid interpretation

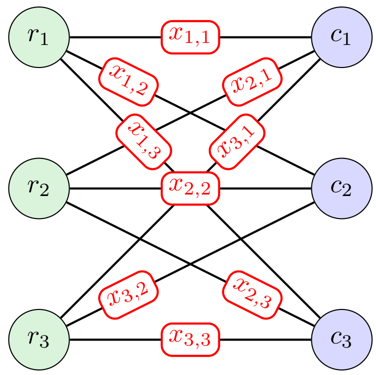


Figure 2 Bipartite interpretation

true if and only if (a) at most one row contains a true variable and (b) at most one column contains a true variable. We encode this as follows. For each row, we introduce an auxiliary variable r_i that is implied by each variable in that row, and we do something similar for the columns. Then, we recursively encode 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 a variable is assigned. Then, the product encoding is as follows:

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

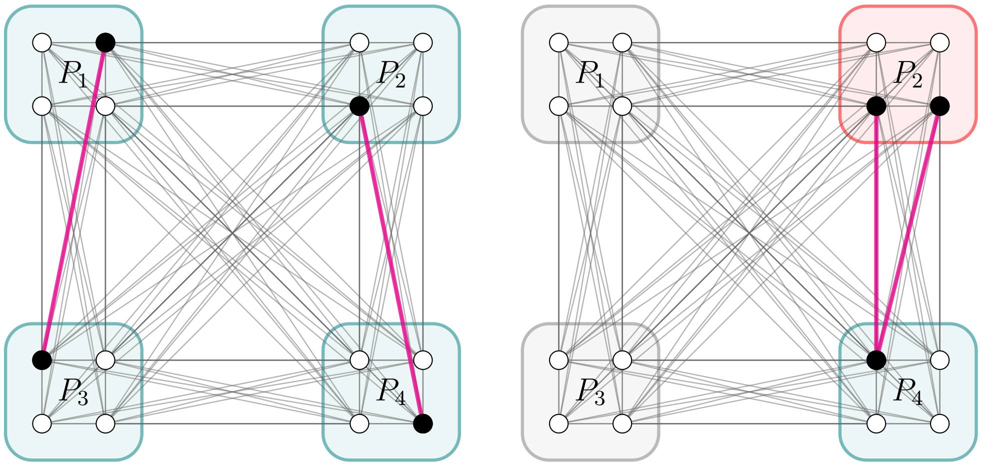
For the base case (say, when $n \leq 4$), we use the direct encoding: $\bigwedge_{1 \leq i < j \leq n} \overline{x_i} \vee \overline{x_j}$.

But there is also another interpretation of the product encoding, illustrated in Figure 2. 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 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 y_i for each vertex $i \in V(G)$, and we spend $2|E|$ clauses to make each edge imply its endpoints: $\overline{x_{i,j}} \vee y_i$ and $\overline{x_{i,j}} \vee y_j$. 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 edge is selected.

3.2 The multipartite encoding

We now describe an encoding for 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. We therefore call our



(a) The AtMostTwo parts constraint is violated. **(b)** The AtMostOne constraint within P_2 is violated.

Figure 3 Illustration of the multipartite encoding. Parts violating the AtMostOne constraint are shaded red. Parts for which z_k is true are shaded teal.

encoding the *multipartite encoding*. This turns out to be a good choice of G 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. Indeed, 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 3).

To construct this encoding, we first require an intermediate construction. Let $\mathsf{AtMostOne}_z(x_1, \dots, x_n)$ be the constraint asserting that at most one of the variables $x_1, \dots, x_n$ is true and that x_i implies z for each $i \in [n]$; that is:

$$\mathsf{AtMostOne}_z(x_1, \dots, x_n) \iff \left(\bigwedge_{1 \le i < j \le n} \overline{x_i} \vee \overline{x_j} \right) \wedge \left(\bigwedge_{i \in [n]} \overline{x_i} \vee z \right).$$

To say that an encoding of this constraint is arc consistent means that the entailments $\mathsf{AtMostOne}_z(x_1, \dots, x_n) \wedge x_i \models \overline{x_j}$ can be derived by unit propagation for all $i \neq j$, and the entailments $\mathsf{AtMostOne}_z(x_1, \dots, x_n) \wedge x_i \models z$ can be derived by unit propagation for all $i \in [n]$.

▶ **Lemma 5.** *There is an arc-consistent encoding of the* $\mathsf{AtMostOne}_z(x_1, \dots, x_n)$ *constraint using* $2n + O(\sqrt{n})$ *clauses and* $O(\sqrt{n})$ *auxiliary variables.*

Proof. 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 a variable is assigned. Then, we can use the following extension of the product encoding:

$$\begin{aligned} \mathsf{AtMostOne}_z(\{x_{i,j} \mid (i,j) \in E\}) := & \left(\bigwedge_{(i,j) \in E} (\overline{x_{i,j}} \vee r_i) \wedge (\overline{x_{i,j}} \vee c_j) \right) \\ & \wedge \mathsf{PE}(r_1, \dots, r_p) \wedge \mathsf{PE}(c_1, \dots, c_p) \wedge \bigwedge_{i \in [p]} (\overline{r_i} \vee z). \end{aligned}$$

It is not hard to see that the encoding uses $2n + O(\sqrt{n})$ clauses and $O(\sqrt{n})$ auxiliary variables, and the justification of the correctness and arc-consistency of this encoding is very similar to that of the product encoding. ◀

Now we can prove Theorem 1.

Proof of Theorem 1. Let G be the complete p-partite graph with q vertices within each part, where $p = \lceil \sqrt[6]{n} \rceil + 1$ and $q = \lceil \sqrt{2} \cdot \sqrt[3]{n} \rceil$, so $|E(G)| \geq n$. 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 a variable is assigned. Discard any vertices of G not incident to an edge from E. Introduce auxiliary variables y_i for each $i \in V(G)$ and z_k for each $k \in [p]$. Our encoding is as follows:

$$\mathsf{ME}(\{x_{\{i,j\}} \mid \{i,j\} \in E\}) := \left(\bigwedge_{\{i,j\} \in E} (\overline{x_{\{i,j\}}} \vee y_i) \wedge (\overline{x_{\{i,j\}}} \vee y_j) \right)$$

$$\wedge \left(\bigwedge_{k \in [p]} \mathsf{AtMostOne}_{z_k}(\{y_i \mid i \in P_k\}) \right) \wedge \mathsf{AtMostTwo}(z_1, \ldots, z_p),$$

where we use an arc-consistent encoding of $\mathsf{AtMostTwo}(z_1, \ldots, z_p)$ using $3p + O(p^{2/3})$ clauses and $O(p^{2/3})$ auxiliary variables.

First, we argue that the encoding is correct and arc consistent. Suppose that at most one input variable is true. If all input variables are false, then $\mathsf{ME}(\{x_{\{i,j\}} \mid \{i,j\} \in E\})$ is satisfiable by setting all of the y 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 y_i, y_j, z_k, and z_ℓ to true and all of the other y and z auxiliary variables to false.

It remains to show that $\mathsf{ME}(\{x_{\{i,j\}} \mid \{i,j\} \in E\}) \wedge x_{\{i,j\}} \models \overline{x_{\{i',j'\}}}$ for all $\{i', j'\} \in E \setminus \{\{i,j\}\}$. If $x_{\{i,j\}}$ is true, then y_i, y_j, z_k, and z_ℓ are derivable by unit propagation. Since our encodings of $\mathsf{AtMostOne}_{z_k}(\{y_i \mid i \in P_k\})$ and $\mathsf{AtMostOne}_{z_\ell}(\{y_i \mid i \in P_\ell\})$ are arc consistent by Lemma 5, we can derive $\overline{y_{i'}}$ by unit propagation for all $i' \in P_k \cup P_\ell \setminus \{y_i, y_j\}$. Thus, we can derive $\overline{x_{\{i',j'\}}}$ by unit propagation for all $\{i', j'\}$ between P_k and P_ℓ other than $\{i, j\}$. Since our encoding of $\mathsf{AtMostTwo}(z_1, \ldots, z_p)$ is arc consistent, $\overline{z_{k'}}$ is derivable by unit propagation for all $k' \notin \{k, \ell\}$, and therefore $\overline{y_{i'}}$ is derivable by unit propagation for all $i' \in P_{k'}$ for all $k' \notin \{k, \ell\}$. Thus, we can derive $\overline{x_{\{i',j'\}}}$ by unit propagation for all $\{i', j'\}$ not between P_k and P_ℓ. We conclude that the encoding is correct and arc consistent.

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

3.3 A smaller circuit for T_2

Now we describe how the same construction yields a circuit for $T_2(x_1, \ldots, x_n)$. Let S_2 be the boolean operator (T_1, T_2). We make use of two standard facts about circuits for threshold functions. First, as an analogue of Lemma 5, S_2 has a circuit of size $2n + O(\sqrt{n})$ [25]. Second, as an analogue of the generalized product encoding, T_3 has a circuit of size $3n + O(n^{2/3})$ [11, 27], where T_3 is the negation of $\mathsf{AtMostTwo}$.

Proof of Theorem 3. Let G be the complete p-partite graph with q vertices within each part, where $p = \lceil \sqrt[6]{n} \rceil + 1$ and $q = \lceil \sqrt{2} \cdot \sqrt[3]{n} \rceil$, so $|E(G)| \geq n$. 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_e is the variable assigned to the edge e. Let E be the set of edges of G to which a variable is assigned. Discard any vertices of G not incident to an edge from E.

For each $i \in V(G)$, let $y_i = \bigvee_{\substack{e \in E \\ i \in e}} x_e$. Then, let $(z_k, w_k) = S_2(\{y_i \mid i \in P_k\})$ for each $k \in [p]$. Then, our circuit for T_2 is as follows:

$$T_2(\{x_e \mid e \in E\}) := \bigvee_{k \in [p]} w_k \vee T_3(z_1, \ldots, z_p).$$

The justification for the correctness of the circuit is similar to the proof of Theorem 1. The number of gates required to compute all of the y_i variables is at most $2n - \sqrt{2n}$. The number of gates required to compute all of the (z_k, w_k) variables is at most $p \cdot (2q + O(\sqrt{q})) = 2\sqrt{2n} + O(\sqrt[3]{n})$. In total, the gate complexity of the circuit is $2n + \sqrt{2n} + O(\sqrt[3]{n})$, as desired. ◀

4 A lower bound for AtMostOne [rough notes]

We use some terminology and results from [19]. Let $\varphi(\vec{x}, \vec{y})$ be a 2-CNF encoding of $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ of minimum size. Suppose first that φ is in restricted regular form. Given a literal ℓ, let $L_{\varphi,\ell} = \{g \mid \overline{\ell} \vee g \in \varphi\}$. Let G be a graph with $V(G) = \bigcup_{i \in [n]} L_{\varphi,x_i}$ and $E(G) = \{L_{\varphi,x_i} \mid i \in [n]\}$. For every $\ell \in V(G)$, we have $|L_{\varphi,\ell}| \geq 2$.

▶ **Lemma 6.** *If G has a vertex of degree d, where $d \geq 6$, then φ has at least $2n + 2d$ clauses.*

Proof. Let $\ell \in V(G)$ have degree d, and let $e_1, \ldots, e_d$ be its incident edges. Without loss of generality, $e_i = L_{\varphi,x_i}$ for each $i \in [d]$. For each $i \in [d]$, let ℓ_i be such that $e_i = \{\ell, \ell_i\}$. For every distinct $i, j \in [d]$, we have $\varphi \wedge x_i \wedge x_j \models \bot$, so $\varphi \wedge \ell \wedge \ell_i \wedge \ell_j \models \bot$. Since φ is 2-CNF, there is some 2-element subset $S \subseteq \{\ell, \ell_i, \ell_j\}$ such that $\varphi \wedge S \models \bot$. But $\varphi \wedge \ell \wedge \ell_i \not\models \bot$ and $\varphi \wedge \ell \wedge \ell_j \not\models \bot$, so $\varphi \wedge \ell_i \wedge \ell_j \models \bot$. Let φ' be the subset of φ not including clauses containing input variables. Then, $\varphi' \wedge \ell_i \wedge \ell_j \models \bot$ for every $i, j \in [d]$. Furthermore, $\varphi' \wedge \ell_i \not\models \bot$ for every $i \in [d]$, since $\varphi \wedge x_i \not\models \bot$. Thus, φ' encodes the at most one constraint with respect to the variables $\{\ell_i \mid i \in [d]\}$, so $|\varphi'| \geq 2d$ and therefore $|\varphi| \geq 2n + 2d$. ◀

Let us say that φ is in *very restricted regular form* if it is in restricted regular form and, furthermore, (a) every vertex of G has degree at most $2\sqrt{n} - 1$, and (b) for every clause $\overline{\ell_1} \vee \overline{\ell_2} \in \varphi$ with $\ell_1, \ell_2 \in V(G)$, we have $\max(|L_{\varphi,\ell_1}|, |L_{\varphi,\ell_2}|) \geq 3$.

If φ does not satisfy condition (a), then φ has at least $2n + 4\sqrt{n} - 2$ clauses by the lemma, and we are done. Next, we say something about when φ does not satisfy condition (b).

▶ **Lemma 7.** *Suppose that φ is a 2-CNF encoding of $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ of the minimum size such that φ is in restricted regular form and every vertex of G has degree at most $2\sqrt{n} - 1$. Suppose further that there is a clause $\overline{\ell_1} \vee \overline{\ell_2} \in \varphi$ with $\ell_1, \ell_2 \in V(G)$ such that $\max(|L_{\varphi,\ell_1}|, |L_{\varphi,\ell_2}|) \leq 2$. Let d be the degree of ℓ_1 in G. Then, $|\varphi| \geq \mathcal{P}_2(n - d) + 2d + 4$.*

Proof. We define a new formula φ' from φ as follows:

$$\varphi' = \varphi \setminus (\{\overline{x_i} \vee \ell_1 \mid \overline{x_i} \vee \ell_1 \in \varphi\} \cup \{\}$$

◀

5 A smaller encoding for AtMostk

Our encoding for $\mathsf{AtMostk}$ is inspired by the generalized product encoding, first described in the context of SAT by Frisch and Giannaros [14]. For the sake of exposition, we begin by presenting the generalized product encoding. Then, we describe our modifications that allow for a more compact encoding.

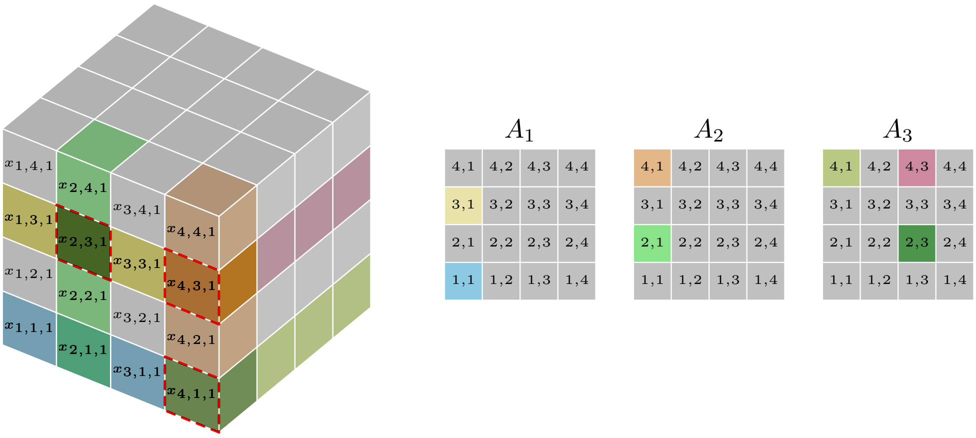


Figure 4 Illustration of the generalized product encoding for AtMostTwo. If $x_{2,3,1}$ is true, then the auxiliary variables $A_{1,(3,1)}$, $A_{2,(2,1)}$, $A_{3,(2,3)}$ are forced to be true. Similarly, if $x_{4,1,1}$ is true, then the auxiliary variables $A_{1,(1,1)}$, $A_{2,(4,1)}$, $A_{3,(4,1)}$ are forced to be true. Finally, variable $x_{4,3,1}$ forces $A_{1,(3,1)}, A_{2,(4,1)}, A_{3,(4,3)}$.

5.1 The generalized product encoding

Rather than imagining the input variables in a two-dimensional grid as in Figure 1, we instead imagine them in a $(k+1)$-dimensional grid (see Figure 4). The key insight of the product encoding was that if at least two input variables are selected (i.e., true), then either at least two rows are selected (i.e., contain a true input variable) or at least two columns are selected. This fact generalizes as follows. Given a tuple $\vec{i} \in [p]^k$ and $d \in [k]$, let $\vec{i}/d$ be $\vec{i}$ with its dth coordinate omitted.

▶ **Lemma 8.** *Given k distinct points $\vec{i_1}, \ldots, \vec{i_k} \in \mathbb{N}^k$, there is some $d \in [k]$ such that $\vec{i_1}/d, \ldots, \vec{i_k}/d$ are distinct.*[3]

Proof. Suppose for a contradiction that we have k distinct points $\vec{i_1}, \ldots, \vec{i_k} \in \mathbb{N}^k$ and yet for each $d \in [k]$, we have $\vec{i_m}/d = \vec{i_n}/d$ for some distinct $m, n \in [k]$. We now create a graph whose vertices are the points $\vec{i_1}, \ldots, \vec{i_k}$, and the edges are as follows. For each $d \in [k]$, choose distinct $m, n \in [k]$ such that $\vec{i_m}/d = \vec{i_n}/d$ and create an edge between $\vec{i_m}$ and $\vec{i_n}$. Note that this edge can be interpreted as saying that we can travel from $\vec{i_m}$ to $\vec{i_n}$ just by moving along one coordinate, which is determined by the edge.

Since our graph has k vertices and k edges, it contains a cycle. Geometrically, this means that we can travel some nonzero distance along each dimension and return where we started, which is absurd. ◀

We now describe how this fact can be leveraged to encode AtMostk$(x_1, \ldots, x_n)$. As the base case, if $n \leq (k+1)^k$, then use the sequential counter encoding from [26], which is arc consistent and uses $O(kn)$ clauses and auxiliary variables. Otherwise, rename the input variables $x_1, \ldots, x_n$ to be of the form $x_{\vec{i}}$ with $\vec{i} \in [p]^{k+1}$, where $p = \lceil n^{1/(k+1)} \rceil$. Let $I \subseteq [p]^{k+1}$ be the set of tuples to which a variable is assigned. For each $d \in [k+1]$ and $\vec{i} \in [p]^k$, introduce

[3] For points in $\{0,1\}^k$, this fact is known as Bondy's theorem [8].

an auxiliary variable $A_{d,\vec{i}}$. Then, the generalized product encoding is as follows:

$$\mathsf{PE}_k(\{x_{\vec{i}} \mid \vec{i} \in I\}) := \bigwedge_{d\in[k+1]} \left(\bigwedge_{\vec{i}\in I} \overline{x_{\vec{i}}} \vee A_{d,\vec{i}/d} \right) \wedge \mathsf{PE}_k(\{A_{d,\vec{i}} \mid \vec{i} \in [p]^k\}).$$

The correctness of the generalized product encoding follows straightforwardly from Lemma 8: If at least $k+1$ input variables are true, then there is some $d \in [k+1]$ such that at least $k+1$ of the variables $\{A_{d,\vec{i}} \mid \vec{i} \in [p]^k\}$ are forced to be true.

5.2 The disjunctive generalized product encoding

In the generalized product encoding, the number of clauses of the form $\overline{x_{\vec{i}}} \vee A_{d,\vec{i}/d}$ is $(k+1)n$. Since our goal is to construct an encoding with $\sim 2n$ clauses, we cannot afford these clauses. It turns out that by exploiting wider clauses, we can make a similar strategy work with $\sim 2n$ clauses. In constructing what we call the *disjunctive generalized product encoding*, we start with the following $2n$ clauses:

$$\bigwedge_{\vec{i}\in I} (\overline{x_{\vec{i}}} \vee A_{1,\vec{i}/1}) \wedge \bigwedge_{\vec{i}\in I} \left(\overline{x_{\vec{i}}} \vee \bigvee_{d\in[2,k+1]} A_{d,\vec{i}/d} \right).$$

Intuitively, every selected input variable is projected onto the first k-dimensional facet (represented using the $A_{1,\vec{i}/1}$ variables) and onto at least one of the remaining k-dimensional facets (represented using the $A_{d,\vec{i}/d}$ variables for $d \in [2, k+1]$). Making this strategy work requires imposing some constraints on which of the $A_{d,\vec{i}/d}$ variables can be true; these constraints are dependent on the values of the $A_{1,\vec{i}/1}$ variables.

Proof of Theorem 4. If $n \le (k+1)^k$, then use the parallel counter encoding from [26], which uses $O(n)$ clauses and auxiliary variables. Otherwise, rename the input variables $x_1, \ldots, x_n$ to be of the form $x_{\vec{i}}$ with $\vec{i} \in [p]^{k+1}$, where $p = \lceil n^{1/(k+1)} \rceil$. Let $I \subseteq [p]^{k+1}$ be the set of tuples to which a variable is assigned. For each $d \in [k+1]$ and $\vec{i} \in [p]^k$, introduce an auxiliary variable $A_{d,\vec{i}}$. For each $d \in [2, k+1]$, introduce an auxiliary variable w_d. Then, the disjunctive generalized product encoding is as follows:

$$\begin{aligned}
\mathsf{DPE}_k(\{x_{\vec{i}} \mid \vec{i} \in I\}) := {} & \bigwedge_{\vec{i}\in I} (\overline{x_{\vec{i}}} \vee A_{1,\vec{i}/1}) \wedge && (1)\\
& \bigwedge_{\vec{i}\in I} \left(\overline{x_{\vec{i}}} \vee \bigvee_{d\in[2,k+1]} A_{d,\vec{i}/d} \right) \wedge && (2)\\
& \bigwedge_{d\in[2,k+1]} \mathsf{AtMostk}(\{A_{d,\vec{i}} \mid \vec{i} \in [p]^k\}) \wedge && (3)\\
& \bigwedge_{d\in[2,k+1]} \bigwedge_{\vec{i}\in[p]^{k-1}} (\overline{w_d} \vee \mathsf{AtMostOne}(\{A_{1,\vec{i'}} \mid \vec{i'}/(d-1) = \vec{i}\})) \wedge && (4)\\
& \mathsf{AtMostOne}(\{w_d \mid d \in [2, k+1]\}) \wedge && (5)\\
& \bigwedge_{d\in[2,k+1]} \bigwedge_{\vec{i}\in[p]^k} (\overline{A_{d,\vec{i}}} \vee w_d). && (6)
\end{aligned}$$

For the AtMostk constraints within this encoding, we use the parallel counter encoding from [26] (rather than recursion); for the AtMostOne constraints, we use any encoding with $O(n)$ clauses and auxiliary variables.

First, we argue that the encoding is correct. Suppose that at most k input variables are true. Let $A_{1,\vec{i}}$ be true if and only if $x_{\vec{i}'}$ is true for some $\vec{i}'$ with $\vec{i}'/1 = \vec{i}$. Then clauses (1) are satisfied. Let I_1 be the set of $\vec{i} \in [p]^k$ such that $A_{1,\vec{i}}$ is true. Clearly, $|I_1| \le k$. By Lemma 8, there is some $d \in [2, k+1]$ such that the $\vec{i}/(d-1)$ are distinct for $\vec{i} \in I_1$. Choose such a d arbitrarily and let w_d be true and the remaining $w_{d'}$ be false. Then, let $A_{d,\vec{i}}$ be true if and only if $x_{\vec{i}'}$ is true for some $\vec{i}'$ with $\vec{i}'/d = \vec{i}$, and let $A_{d',\vec{i}}$ be false for all $d' \in [2, k+1] \setminus \{d\}$. Then, clauses (2), (3), (5), and (6) are satisfied. Also, clauses (4) are satisfied by our choice of d. Thus, the formula is satisfiable.

Conversely, suppose that at least $k+1$ base variables are true. By clauses (1), we must have $A_{1,\vec{i}/1}$ true for each $\vec{i} \in I$ such that $x_{\vec{i}}$ is true. By clauses (2) and (6), we must have w_d true for some $d \in [2, k+1]$, so let d be such that w_d is true. By clauses (5), $w_{d'}$ is false for all $d' \in [2, k+1] \setminus \{d\}$. By clauses (2) and (6), we must have $A_{d,\vec{i}/d}$ true for each $\vec{i} \in I$ such that $x_{\vec{i}}$ is true. Then, by clauses (4), for each $\vec{i} \in [p]^{k-1}$, we have at most one $A_{1,\vec{i}'}$ true such that $\vec{i}'/(d-1) = \vec{i}$. Hence, for each $\vec{i} \in [p]^k$, we have at most one $x_{\vec{i}'}$ true such that $\vec{i}'/d = \vec{i}$. Thus, there are at least $k+1$ variables among $\{A_{d,\vec{i}} \mid \vec{i} \in [p]^k\}$ true, contradicting clauses (3). We conclude that the encoding is correct.

Next, we count the number of clauses and auxiliary variables. The number of clauses in (1) and (2) is n each; the number of clauses in (3), (4), and (6) is $O(kn^{k/(k+1)})$; the number of clauses in (5) is $O(k)$. In total, the number of clauses is $2n + O(kn^{k/(k+1)})$, as desired. The number of auxiliary variables of the form $A_{d,\vec{i}}$ is $(k+1) \cdot p^k = O(kn^{k/(k+1)})$; the number of auxiliary variables of the form w_d is k; the number of auxiliary variables used by the $\mathsf{AtMostk}$ and $\mathsf{AtMostOne}$ constraints in (3) and (4) is $O(kn^{k/(k+1)})$; the number of auxiliary variables used by the $\mathsf{AtMostOne}$ constraint in (5) is $O(k)$. In total, the number of auxiliary variables is $O(kn^{k/(k+1)})$, as desired. ◀

6 A smaller encoding for AtLeastk [WIP]

▶ **Theorem 9.** *There is a unit refutation complete encoding of the* $\mathsf{AtLeastTwo}(x_1, \ldots, x_n)$ *constraint using* $2 \cdot \lceil \log n \rceil$ *clauses and* $\lceil \log n \rceil - 2$ *auxiliary variables.*

Proof. Rename the variables $x_1, \ldots, x_n$ to be of the form x_w with $w \in \{0,1\}^{\lceil \log n \rceil}$. For each $i \in [\lceil \log n \rceil]$ and $b \in \{0,1\}$, let $W_{i,b}$ be the set of elements of $\{0,1\}^{\lceil \log n \rceil}$ whose ith bit is b. Introduce auxiliary variables $y_1, \ldots, y_{\lceil \log n \rceil}$. Our encoding is as follows:

$$\mathsf{PHF}_2(\{x_w \mid w \in \{0,1\}^{\lceil \log n \rceil}\}) := \left(\bigvee_{y \in [\lceil \log n \rceil]} y_i \right) \wedge \left(\bigwedge_{\substack{i \in [\lceil \log n \rceil] \\ b \in \{0,1\}}} \left(\overline{y_i} \vee \bigvee_{w \in W_{i,b}} x_w \right) \right).$$

This encoding has $2 \cdot \lceil \log n \rceil + 1$ clauses and $\lceil \log n \rceil$ auxiliary variables, but if we resolve away any two of the auxiliary variables, we obtain an equivalent encoding with $2 \cdot \lceil \log n \rceil$ clauses and $\lceil \log n \rceil - 2$ auxiliary variables.

◀

7 Open problems

Is there an arc-consistent encoding of $\mathsf{AtMostk}(x_1, \ldots, x_n)$ with $O(n)$ clauses?

8 Conclusion

We solved a fundamental problem in the theory of CNF encodings by showing that the minimum number of clauses in an encoding of $\mathsf{AtMostk}(x_1, \dots, x_n)$ is asymptotic to $2n$ for each fixed k. We also tightened the upper bound on the minimum number clauses in an encoding of AtMostOne, refuting a conjecture of Chen [10] and resolving a long-standing open problem in circuit complexity.

Our constructions introduce several new techniques for constructing CNF encodings, which manifest as unique properties that, to the best of our knowledge, are not present in any previously known encodings. The multipartite encoding for AtMostOne is notable for using clauses of width 3, despite the fact that AtMostOne is a 2-CNF function. Kučera, Savický, and Vorel [19] asked whether the smallest arc-consistent 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 AtMostOne, the simplest antitone 2-CNF function.[4] The fact that our encoding is not 2-CNF is related to the fact that the corresponding circuit is not single level.

Wide clauses play an even more central role in in the disjunctive generalized product encoding for AtMostk. At a high level, the algorithm underlying this encoding can be summarized as follows. First, we project the selected input variables onto the first k-dimensional facet. Based on the content of this projection (i.e., the values of the $A_{1,\vec{i}/1}$ variables), we determine a $d \in [2, k+1]$ and project the selected input variables onto the dth k-dimensional facet. These two projections give us enough information to determine if at most k input variables are selected. One barrier when constructing compact encodings based on an algorithm like this one is that every branch of the algorithm's execution must be part of the encoding, which can make the encoding larger than one would expect based on the runtime of the algorithm. The disjunctive generalized product encoding shows that it is sometimes possible to avoid this overhead by using wide clauses to handle multiple branches at once. In this encoding, we do this using clauses of the form $\overline{x_{\vec{i}}} \vee \bigvee_{d \in [2,k+1]} A_{d,\vec{i}/d}$, which obviates the need to include clauses $\overline{x_{\vec{i}}} \vee A_{d,\vec{i}/d}$ for every $d \in [2, k+1]$ as in the generalized product encoding.

Another interesting aspect of the disjunctive generalized product encoding is that, unlike the multipartite encoding, it does not correspond to a monotone circuit of comparable size. Indeed, Sergeev [25] has proved that every monotone boolean circuit for $T_3(x_1, \dots, x_n)$ has at least $3n - 5$ gates for $n \geq 4$. This culprit here is the use of the wide clauses $\overline{x_{\vec{i}}} \vee \bigvee_{d \in [2,k+1]} A_{d,\vec{i}/d}$, which cannot be nicely translated into a circuit. Although previous researchers have noted the close connection between CNF encodings and circuit complexity (see, e.g., [19, 25, 13]), Theorem 4 together with Sergeev's lower bound shows a substantial sense in which these models of computation differ.

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" [13], we expect the further development of this theory to be of great practical significance.

[4] Unfortunately, this does not yet answer Kučera, Savický, and Vorel's question, since we have not ruled out that there is an even better encoding for AtMostOne that is 2-CNF, although we conjecture that this is not the case.

References

1. M. Ajtai, J. Komlós, and E. Szemerédi. Sorting in $c \log n$ parallel steps. *Combinatorica*, 3(1):1–19, 1983. doi:10.1007/BF02579338.
2. 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.
3. Roberto Asín, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell. Cardinality networks: a theoretical and empirical study. *Constraints*, 16(2):195–221, 2011. URL: https://doi.org/10.1007/s10601-010-9105-0, doi:10.1007/S10601-010-9105-0.
4. Olivier Bailleux and Yacine Boufkhad. Efficient CNF encoding of boolean cardinality constraints. In Francesca Rossi, editor, *Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003, Proceedings*, volume 2833 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2003. doi:10.1007/978-3-540-45193-8_8.
5. Kenneth E. Batcher. Sorting networks and their applications. In *American Federation of Information Processing Societies: AFIPS Conference Proceedings: 1968 Spring Joint Computer Conference, Atlantic City, NJ, USA, 30 April - 2 May 1968*, volume 32 of *AFIPS Conference Proceedings*, pages 307–314. Thomson Book Company, Washington D.C., 1968. doi:10.1145/1468075.1468121.
6. 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 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings*, volume 7317 of *Lecture Notes in Computer Science*, pages 397–409. Springer, 2012. doi:10.1007/978-3-642-31612-8_30.
7. 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.
8. J. A. Bondy. Induced subsets. *J. Combinatorial Theory Ser. B*, 12:201–202, 1972. doi:10.1016/0095-8956(72)90025-1.
9. Siegfried Bublitz. Decomposition of graphs and monotone formula size of homogeneous functions. *Acta Inform.*, 23(6):689–696, 1986. doi:10.1007/BF00264314.
10. 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://api.semanticscholar.org/CorpusID:15322159.
11. Paul E. Dunne. *Techniques for the analysis of monotone Boolean networks.* PhD thesis, University of Warwick, 1984.
12. Niklas Eén and Niklas Sörensson. Translating pseudo-boolean constraints into SAT. *J. Satisf. Boolean Model. Comput.*, 2(1-4):1–26, 2006. URL: https://doi.org/10.3233/sat190014, doi:10.3233/SAT190014.
13. Gregory Emdin, Alexander S. Kulikov, Ivan Mihajlin, and Nikita Slezkin. CNF encodings of symmetric functions. *Theory Comput. Syst.*, 68(5):1291–1311, 2024. URL: https://doi.org/10.1007/s00224-024-10168-w, doi:10.1007/S00224-024-10168-W.
14. 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.
15. 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.
16. M. I. Grinchuk. On the monotone complexity of threshold functions. *Metody Diskret. Anal.*, (52):41–48, 120, 1992.
17. Stasys Jukna. Disproving the single level conjecture. *SIAM J. Comput.*, 36(1):83–98, 2006. doi:10.1137/S0097539705447001.
18. R. E. Krichevskii. Complexity of contact circuits realizing a function of logical algebra. *Sov. Phys. Dokl.*, 8:770–772, 1964.

19 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. URL: `https://doi.org/10.1016/j.tcs.2018.09.003`, `doi:10.1016/J.TCS.2018.09.003`.

20 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`.

21 João Marques-Silva and Inês Lynce. Towards robust CNF encodings of cardinality constraints. In Christian Bessiere, editor, *Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings*, volume 4741 of *Lecture Notes in Computer Science*, pages 483–497. Springer, 2007. `doi:10.1007/978-3-540-74970-7\_35`.

22 R. Mirwald and C.-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`.

23 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`.

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

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

26 Carsten Sinz. Towards an optimal CNF encoding of boolean cardinality constraints. In Peter van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, volume 3709 of *Lecture Notes in Computer Science*, pages 827–831. Springer, 2005. `doi:10.1007/11564751\_73`.

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