Chen's Optimality Conjecture is False: A Smaller Encoding for the AtMostOne Constraint

Benjamin Przybocki
Carnegie Mellon University
benjamin.przybocki@gmail.com

Bernardo Subercaseaux
Carnegie Mellon University
bersub@cmu.edu

August 11, 2025

Abstract

We refute a conjecture of Chen [Che10] about the optimal size of CNF encodings for the AtMostOne$(x_1, \ldots, x_n)$ constraint, which is arguably the most well-studied building block of SAT encodings. Chen's *"product encoding"* uses only $2n+(4+o(1))\sqrt{n}$ clauses and $(2+o(1))\sqrt{n}$ auxiliary variables, almost matching the lower bounds proven by Kučera, Savický, and Vorel [KSV19]. In this note we present an encoding using only $2n+(2\sqrt{2}+o(1))\sqrt{n}$ clauses, and only $(\sqrt{2}+o(1))\sqrt{n}$ auxiliary variables, thus beating the product encoding on both counts.

1. Introduction

Chen's *"Product Encoding"* [Che10] has been the most succinct for the AtMostOne$(x_1, \ldots, x_n)$ constraint for 15 years: it uses $2n + 4\sqrt{n} + o(\sqrt{n})$ clauses and only $2\sqrt{n} + o(\sqrt{n})$ auxiliary variables. The same construction, in its complementary form AtLeastTwo$(x_1, \ldots, x_n)$, can be traced back to the 1970s circuit complexity literature [Blo79].

To the best of our knowledge, the product encoding is the only SAT encoding in the literature with $2n + o(n)$ clauses (see e.g., [NNKB21]). Furthermore, in his 2010 paper, Chen conjectured his encoding to be optimal in terms of the number of clauses, and despite a plethora of papers regarding encodings for the AtMostOne constraint, Chen's conjecture has survived the test of time. It is known that the product encoding is very close to being optimal: Kučera, Savický, and Vorel have proved that encodings with 'good propagation properties' cannot beat $2n + \sqrt{n}$ clauses, and even a stronger lower bound of $2n + 2\sqrt{n}$ clauses for 2-CNF encodings [KSV19].[1] The main result of this note is a refutation of Chen's conjecture.

Theorem 1. *The AtMostOne$(x_1, \ldots, x_n)$ constraint can be encoded using $2n + 2\sqrt{2n} + o(\sqrt{n})$ clauses and $\sqrt{2n} + o(\sqrt{n})$ auxiliary variables, while maintaining propagation completeness.*

Thus, not only does our encoding use fewer clauses than the product encoding, it also uses asymptotically fewer auxiliary variables.

Our main ingredient is a novel conceptual perspective on AtMostOne encodings, which we call the *"unique-edge framework"* (Section 3). The unique-edge framework vastly generalizes the product encoding, assigning to each undirected graph $G = (V, E)$ a CNF formula φ_G that encodes

[1] We include as well in Appendix B (Corollary 9), a new lower bound of $2n$ clauses that holds for all encodings regardless of their propagation properties.

$\mathsf{AtMostOne}(x_1, \ldots, x_{|E|})$ by using roughly $|V|$ auxiliary variables corresponding to the vertices of G. In a nutshell, we use conditions on the vertices of G to enforce that at most one edge of G is selected. In this framework, the product encoding corresponds to taking $G = K_{\sqrt{n},\sqrt{n}}$, a complete bipartite graph. We prove Theorem 1, in Section 4, by considering instead a complete multipartite graph with roughly $\sqrt[4]{n}$ parts of $\sqrt[4]{n}$ vertices each.

In order to keep the presentation of our main result brief, we defer the introduction of standard notation of SAT encodings to Appendix A.

2. The Product Encoding (Grid Version)

Chen's product encoding for $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ is recursive, and usually presented in terms of a grid, as we show next. The base case of the recursion is when $n \leqslant 4$ (or some other constant), where the pairwise encoding is used. For $n > 4$, place the n variables $x_1, \ldots, x_n$ into a $p \times q$ grid with $p \cdot q \geqslant n$, as illustrated in Figure 1. We can assume without affecting the asymptotic analysis that $p = q = \lceil \sqrt{n} \rceil$. For ease of notation, rename the variables so that $x_{i,j}$ is the variable placed on the intersection of row i and column j. We denote by E the set of indices (i, j) such that variable $x_{i,j}$ exists.

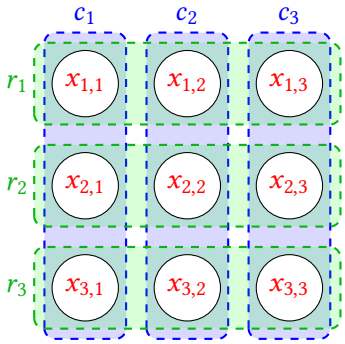


Figure 1: Grid interpretation.

Then, create auxiliary variables $r_1, \ldots, r_p$ and $c_1, \ldots, c_p$, where intuitively r_i will represent that some variable in row i is true, and c_j that some variable in column j is true. We enforce this by adding, for each variable $x_{i,j}$, clauses $(\overline{x_{i,j}} \vee r_i)$ and $(\overline{x_{i,j}} \vee c_j)$, incurring in a total of $2n$ clauses. Finally, we add constraints $\mathsf{AtMostOne}(r_1, \ldots, r_p)$ and $\mathsf{AtMostOne}(c_1, \ldots, c_q)$, encoded recursively with this same encoding. The resulting formula is of the form:

$$\mathsf{PE}(\{x_{i,j} \mid (i,j) \in E\}) \equiv \left(\bigwedge_{(i,j)\in E} (\overline{x_{i,j}} \vee r_i) \wedge (\overline{x_{i,j}} \vee c_j) \right) \wedge \mathsf{PE}(r_1, \ldots, r_p) \wedge \mathsf{PE}(c_1, \ldots, c_q). \quad (1)$$

Correctness can be argued by contradiction: if we suppose a satisfying assignment with variables x_{i_1,j_1} and x_{i_2,j_2} assigned to $\top$ (true), then every variable in $S := \{r_{i_1}, r_{i_2}, c_{j_1}, c_{j_2}\}$ must be assigned to $\top$ as well, and since $(i_1, j_1) \neq (i_2, j_2)$, we have $|S| \geqslant 3$, from where the pigeonhole principle implies there must be either 2 row variables or 2 column variables assigned to $\top$, contradicting the encoding.

Let $C_{\mathsf{PE}}(n)$ be the number of clauses used by the product encoding for $\mathsf{AtMostOne}(x_1, \ldots, x_n)$, and $A_{\mathsf{PE}}(n)$ the number of auxiliary variables used. Then, from Equation (1), we have:

$$C_{\mathsf{PE}}(n) = \begin{cases} \binom{n}{2} & \text{if } n \leqslant 4 \\ 2n + 2C_{\mathsf{PE}}\left(\lceil \sqrt{n} \rceil\right) & \text{otherwise} \end{cases} \quad ; \quad A_{\mathsf{PE}}(n) = \begin{cases} 0 & \text{if } n \leqslant 4 \\ 2\lceil \sqrt{n} \rceil + 2A_{\mathsf{PE}}\left(\lceil \sqrt{n} \rceil\right) & \text{otherwise.} \end{cases}$$

A simple inductive argument, which we detail in Appendix C for completeness, yields the following.

Proposition 2 ([Che10]). *$C_{PE}(n) = 2n + 4\sqrt{n} + o(\sqrt{n})$ and $A_{PE}(n) = 2\sqrt{n} + o(\sqrt{n})$.*

3. The Unique-Edge Framework: a Graph Perspective on AtMostOne encodings

Let $G = (V, E)$ be an undirected graph, with $n = |V|$ and $m = |E|$. For each edge $\{u, v\} \in E$, create a variable $e_{\{u,v\}}$, and for each vertex $v \in V$, create a variable x_v. Let $\boldsymbol{e}$ denote the set of edge variables $e_{\{u,v\}}$ and $\boldsymbol{x}$ the set of vertex-variables x_v. We may have other additional auxiliary variables, all of which form a set $\boldsymbol{r}$.

Now, suppose we have a CNF formula $F(\boldsymbol{e}, \boldsymbol{x}, \boldsymbol{r})$ such that for each assignment τ of the $\boldsymbol{e}$ variables, the remaining formula $F|_\tau$ is satisfiable if and only if τ assigns at most one edge variable to $\top$. Then, F is an encoding of AtMostOne($\boldsymbol{e}$). So far, we have done nothing other than to identify the base variables of AtMostOne with edges of a graph, and some of the auxiliary variables with the vertices of the graph — however, this graph reformulation turns out to provide a framework towards better encodings. In this framework, it is natural to start encodings by spending $2|E|$ clauses in making each edge variable imply its endpoints:

$$(\overline{e_{\{u,v\}}} \vee x_u) \quad \text{and} \quad (\overline{e_{\{u,v\}}} \vee x_v).$$

After this, the key idea will be to use the vertex variables to succinctly encode that at most one edge variable is selected. The concrete way of doing this will depend on the graph G chosen. Let us see a couple of examples.

3.1. The Biclique Reformulation of the Product Encoding

Chen's product encoding can be seen in this framework as follows. Let (X, Y, E) be a complete bipartite graph, with $|X| = |Y| = n$ and $|E| = n^2$. Then, for each edge $\{u, v\}$, we add the clauses $(\overline{e_{\{u,v\}}} \vee x_u)$ and $(\overline{e_{\{u,v\}}} \vee x_v)$, stating that each edge implies its endpoints. Now, to say that at most one edge variable will be assigned to $\top$, it suffices to say that at most one vertex from X will be assigned to $\top$ (i.e., AtMostOne($\{x_v \mid v \in X\}$) and at most one vertex from Y will be assigned to $\top$ (i.e., AtMostOne($\{x_v \mid v \in Y\}$)). Indeed, assigning 2 edges variables e_1, e_2 to $\top$ would imply at least 3 vertices assigned to $\top$ (3 in case e_1 and e_2 share an endpoint, and 4 otherwise), and by the pigeonhole principle there would be either 2 in X or 2 in Y.

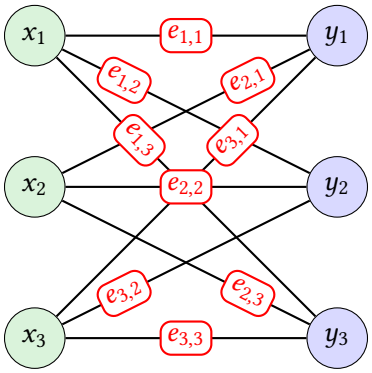


Figure 2: Bipartite interpretation.

Once again, we encode AtMostOne($\{x_v \mid v \in X\}$) and AtMostOne($\{x_v \mid v \in Y\}$) recursively with this same procedure. If we denote by $g(|E|)$ the number of clauses to encode that at most one edge of E is selected with this encoding, we have $g(n^2) = 2n^2 + 2g(n)$, which, after doing the change of variable $N := n^2$, yields $g(N) = 2N + 2g(\sqrt{N})$, essentially the same recursive equation describing Chen's encoding.

3.2. The Clique Encoding

While the previous subsection only showed a change in perspective, here we will show a simple example of how taking a different graph G can lead to interesting encodings. For example, instead of letting G be a complete bipartite graph, we can choose a complete graph. Let $G = K_p$, where

$p = \lceil \sqrt{2n} \rceil + 1$, and thus $|E(G)| = \binom{p}{2} = \frac{p(p-1)}{2} \geqslant \frac{(\sqrt{2n}+1)\sqrt{2n}}{2} = \frac{2n+\sqrt{2n}}{2} \geqslant n$.

Then, we assign the variables $x_1, \ldots, x_n$ to unique edges of K_p, 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. Create auxiliary variables y_i for each $i \in [p]$. Our encoding includes the clauses $(\overline{x_{\{i,j\}}} \vee y_i)$ and $(\overline{x_{\{i,j\}}} \vee y_j)$ for each $\{i, j\} \in E$. We have selected at most one edge if and only if we have selected at most two vertices, so we add the constraint $\mathsf{AtMostTwo}(y_1, \ldots, y_p)$. Thus, our encoding is as follows:

$$\mathsf{CE}(\{x_{\{i,j\}} \mid \{i, j\} \in E\}) \equiv \left(\bigwedge_{\{i,j\} \in E} (\overline{x_{\{i,j\}}} \vee y_i) \wedge (\overline{x_{\{i,j\}}} \vee y_j) \right) \wedge \mathsf{AtMostTwo}(y_1, \ldots, y_p).$$

Let $C_{\mathsf{CE}}(n)$ be the number of clauses used by the clique encoding for $\mathsf{AtMostOne}(x_1, \ldots, x_n)$, and $A_{\mathsf{CE}}(n)$ the number of auxiliary variables used. Given that $\mathsf{AtMostTwo}(y_1, \ldots, y_p)$ can be encoded with $3p + o(p)$ clauses and $o(p)$ auxiliary variables [FG10], we have

$$\begin{aligned} C_{\mathsf{CE}}(n) &= 2n + 3p + o(p) = 2n + 3\sqrt{2n} + o(\sqrt{n}) \\ A_{\mathsf{CE}}(n) &= p + o(p) = \sqrt{2n} + o(\sqrt{n}). \end{aligned}$$

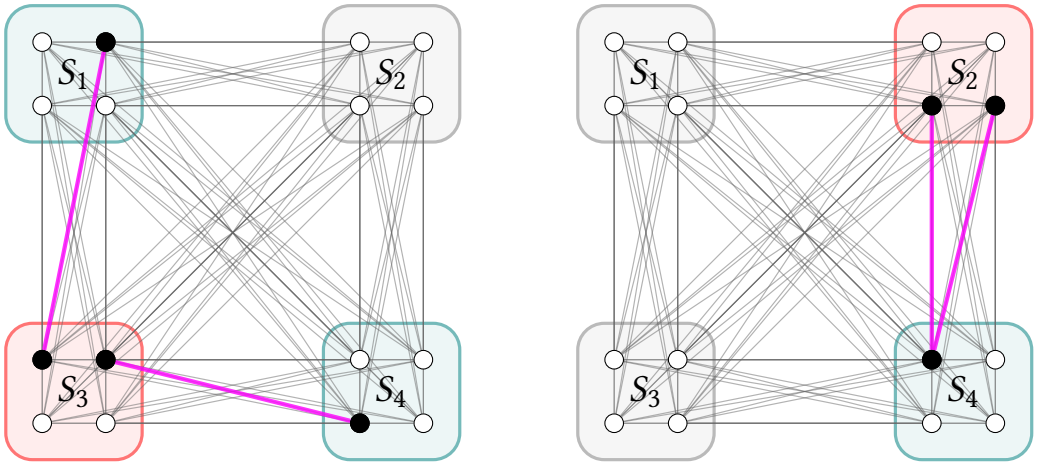
Since $3\sqrt{2} > 4$, the clique encoding has slightly more clauses than the product encoding. But, since $\sqrt{2} < 2$, the clique encoding has asymptotically fewer auxiliary variables than the product encoding.

4. The Improved Encoding

We now describe an encoding for $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ with $2n + 2\sqrt{2n} + o(\sqrt{n})$ clauses and $\sqrt{2n} + o(\sqrt{n})$ auxiliary variables, thus proving Theorem 1. Since complete bipartite graphs and complete graphs both turned out to be useful choices of graphs within our framework, it is natural to try a complete multipartite graph. We therefore call this encoding the *multipartite encoding*. Our construction and its analysis are not sensitive to the exact parameters of G, but for concreteness let $G = K_{p \times p}$ (i.e., the complete p-partite graph with p vertices within each part), where $p = \lceil \sqrt[4]{2n} \rceil + 1$, and thus $p^2 \cdot \binom{p}{2} \geq n$.

Then, we assign the variables $x_1, \ldots, x_n$ to unique edges of $K_{p \times p}$, 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. Create auxiliary variables y_i for each $i \in [p^2]$. As before, we include the clauses $(\overline{x_{\{i,j\}}} \vee y_i)$ and $(\overline{x_{\{i,j\}}} \vee y_j)$ for each $\{i, j\} \in E$. We have selected at most one edge if and only if (a) we have selected at most one vertex from each part and (b) we have selected vertices from at most two parts. We encode this as follows. Let $S_1, \ldots, S_p$ be the parts of G. For each $i \in [p]$, we will add to our encoding the formula $\mathsf{AtMostOne}_{z_i}(\{y_j \mid j \in S_i\})$, which asserts that at most one of $\{y_j \mid j \in S_i\}$ is true and creates an auxiliary variable z_i that is implied by each of the variables in $\{y_j \mid j \in S_i\}$. Then, we add the constraint $\mathsf{AtMostTwo}(\{z_i \mid i \in [p]\})$. Thus, our encoding is as follows:

$$\begin{aligned} \mathsf{ME}(\{x_{\{i,j\}} \mid \{i, j\} \in E\}) \equiv &\left(\bigwedge_{\{i,j\} \in E} (\overline{x_{\{i,j\}}} \vee y_i) \wedge (\overline{x_{\{i,j\}}} \vee y_j) \right) \\ &\wedge \left(\bigwedge_{i \in [p]} \mathsf{AtMostOne}_{z_i}(\{y_j \mid j \in S_i\}) \right) \wedge \mathsf{AtMostTwo}(\{z_i \mid i \in [p]\}). \end{aligned}$$



(a) Both the AtMostOne constraint within S_3 and the AtMostTwo parts constraint are violated.

(b) Only the AtMostOne constraint within S_2 is violated.

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

We claim that $\mathsf{AtMostOne}_z(x_1, \ldots, x_n)$ can be encoded with $2n + o(n)$ clauses and $o(n)$ auxiliary variables. Indeed, we can use the following extension of the product encoding:

$$\mathsf{AtMostOne}_z(\{x_{i,j} \mid (i,j) \in E\}) \equiv \left(\bigwedge_{(i,j)\in E} (\overline{x_{i,j}} \vee r_i) \wedge (\overline{x_{i,j}} \vee c_j) \right) \wedge \mathsf{PE}(r_1, \ldots, r_p) \wedge \mathsf{PE}(c_1, \ldots, c_p) \wedge \bigwedge_{i \in p} (\overline{r_i} \vee z).$$

Let $C_{\mathsf{ME}}(n)$ be the number of clauses used by the multipartite encoding for $\mathsf{AtMostOne}(x_1, \ldots, x_n)$, and $A_{\mathsf{ME}}(n)$ the number of auxiliary variables used. We have

$$C_{\mathsf{ME}}(n) = 2n + p \cdot (2p + o(p)) + (3p + o(p)) = 2n + 2p^2 + o(p^2) = 2n + 2\sqrt{2n} + o(\sqrt{n})$$
$$A_{\mathsf{ME}}(n) = p^2 + p \cdot o(p) + o(p) = p^2 + o(p^2) = \sqrt{2n} + o(\sqrt{n}).$$

5. Concluding Remarks

We have presented a fruitful framework for encodings of the AtMostOne constraint, which has led us to improve on a construction dating back from the 1970s and conjectured optimal over 15 years ago.

The most pressing question at this point is whether the multipartite encoding we have presented is optimal. We are not sure, but we believe that within our framework $2n + (2\sqrt{2} + o(1))\sqrt{n}$ might be optimal. Proving this is a lower bound within our framework is part of our most immediate future work.

An interesting property of the multipartite encoding is that, as opposed to most AtMostOne encodings, it is not 2-CNF: the AtMostTwo constraint uses clauses of width 3 for the base case of its recursion. As observed by Kučera et al. [KSV19], the class of 2-CNF formula is closed under resolution, and thus if a boolean function $f(x_1, \ldots, x_n)$ cannot be written in 2-CNF without auxiliary variables,

there is no way to encode it as a 2-CNF with auxiliary variables, as otherwise resolving away the auxiliary variable would yield a 2-CNF for f in terms of its base variables, a contradiction. But it is easy to see that the AtMostTwo$(x_1, \ldots, x_n)$ boolean function cannot be written as a 2-CNF in terms of its base variables. This implies that our multipartite encoding is fundamentally not encodable as 2-CNF. This comes tantalizingly close to answering a question of Kučera, Savický and Vorel [KSV19, Question 7.1]:

"Assume, $f(\boldsymbol{x})$ is a boolean function expressible by a monotone or antimonotone 2-CNF formula. Is there a propagation complete encoding of the function f of minimum size, which is, moreover, a 2-CNF formula?"

If it turns out that no 2-CNF encoding can match our multipartite encoding in terms of the number of clauses, that would mean a negative answer to their question. Therefore, the multipartite encoding is a natural candidate for further studying this problem.

We have several plans to generalize and extend this line of research, for example, studying whether $(K+1)$-uniform hypergraphs can lead to better encodings for AtMostK constraints. We have omitted here, for the sake of conciseness, a variety of observations and related results that we plan to include in an upcoming conference version of this work.

References

[BBH+09] A. Biere, A. Biere, M. Heule, H. van Maaren, and T. Walsh. *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. IOS Press, NLD, 2009.

[Blo79] 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.

[Che10] 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.

[FG10] 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.

[KSV19] Petr Kučera, Petr Savický, and Vojtěch Vorel. A lower bound on CNF encodings of the at-most-one constraint. *Theoretical Computer Science*, 762:51–73, 2019.

[NNKB21] Van-Hau Nguyen, Van-Quyet Nguyen, Kyungbaek Kim, and Pedro Barahona. Empirical study on sat-encodings of the at-most-one constraint. In *The 9th International Conference on Smart Media and Applications*, SMA 2020, page 470–475, New York, NY, USA, 2021. Association for Computing Machinery.

A. Notation

For a general overview of satisfiability and encodings, we refer the reader to *"the handbook"* [BBH+09]. We use $\top$ and $\bot$ to denote *true* and *false*, respectively. The negation of a variable x is denoted $\overline{x}$, and a *literal* is either a variable or the negation of a variable. Literals x and $\overline{x}$ are said to be complementary,

for any variable x. A *clause* is a set of non-complementary literals, and a *formula* is a set of clauses (we will thus identify $\wedge$ with $\cup$ when operating over clauses, and $\ell_1 \vee \ell_2$ with $\{\ell_1\} \cup \{\ell_2\}$ when operating over literals). The size of a formula is simply its number of clauses. We denote the set of variables appearing in a formula F as $\mathsf{Var}(F)$. Given a set $\mathcal{V}$ of variables, a $\mathcal{V}$-assignment is a function $\tau : \mathcal{V} \to \{\bot, \top\}$. For a variable $x \in \mathcal{V}$, we say $\tau \models x$ if $\tau(x) = \top$, and similarly, $\tau \models \overline{x}$ if $\tau(x) = \bot$. For a clause C, we say $\tau \models C$ if $\bigvee \ell \in C\tau \models C$, and a for a formula F, $\tau \models F$ if $\bigwedge_{C \in F} \tau \models C$. When writing this, we assume implicitly that $\mathsf{Var}(F) \subseteq \mathcal{V}$. For an assignment $\tau : \mathcal{V} \to \{\bot, \top\}$ and a formula F, now with $\mathcal{V} \subseteq \mathsf{Var}(F)$, we denote by $F|_\tau$ the formula obtained by eliminating from F each clause satisfied by τ, and then from each remaining clause eliminating every literal ℓ such that $\tau \models \overline{\ell}$. Note that $\mathsf{Var}(F|_\tau) = \mathsf{Var}(F) \setminus \mathcal{V}$. We will write $\mathsf{SAT}(F)$ to say that $\tau \models F$ for some assignment τ, and $\mathsf{UNSAT}(F)$ to mean that no such assignment exists.

Definition 3 (Encoding). Let $f : \{\bot, \top\}^n \to \{\bot, \top\}$ be a boolean function. Let $\mathcal{X} := \{x_1, \ldots, x_n\}$ be a set of "base" variables, and $\mathcal{Y} := \{y_1, \ldots, y_s\}$ a set of "auxiliary" variables. Then, we say a formula F over $\mathcal{X} \cup \mathcal{Y}$ is an *encoding* for f if every $\mathcal{X}$-assignment τ holds that

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

B. Unconditional Lower Bounds

Let $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ be the boolean function that returns $\top$ if and only if at most one of its inputs x_i is $\top$, and similarly, $\mathsf{ExactlyOne}(x_1, \ldots, x_n)$ is the boolean function that returns $\top$ when exactly one of its inputs x_i is $\top$. Let us denote by $a(n)$ the minimum number of clauses in any encoding of $\mathsf{AtMostOne}(x_1, \ldots, x_n)$, and $e(n)$ the minimum number of clauses in any encoding of $\mathsf{ExactlyOne}(x_1, \ldots, x_n)$. Before we can prove lower bounds on $a(n)$ and $e(n)$, we will need an auxiliary lemma about resolution. For any formula F, containing a literal ℓ, we define $F(\ell) \subseteq F$ as the subset of clauses containing ℓ. Then, if F does not contain tautological clauses (i.e., clauses with complimentary literals), and x is any variable of F, we can partition F into three sets: $F(x), F(\overline{x})$, and $F' := F \setminus F(x) \setminus F(\overline{x})$. The *"resolution rule"* is then:

Lemma 4 (Folklore). *Let F be a formula and $x \in Var(F)$ be one of its variables. Then, the formula*

$$F_x := F' \cup \{(C \setminus \{x\}) \cup (C' \setminus \{\overline{x}\}) \mid C \in F(x), C' \in F(\overline{x})\}$$

has $Var(F_x) = Var(F) \setminus \{x\}$, and for every $Var(F_x)$-assignment τ, we have that

$$\tau \models F_x \iff SAT(F|_\tau).$$

Moreover,

$$|F_x| = |F| - |F(x)| - |F(\overline{x})| + |F(x)| \cdot |F(\overline{x})|. \tag{2}$$

Now, let us say a formula F is an *optimal encoding* for a boolean function f if F is an encoding of f with the minimum number of clauses among all formulas encoding f. Then, the simple lemma we need is as follows. (Note: This is almost the same as Lemma 3.6 in the "A lower bound on CNF encodings of the at-most-one constraint" paper)

Lemma 5. *Let $f(x_1, \ldots, x_n)$ be any boolean function, and F be a formula over the variables $\{x_1, \ldots, x_n, y_1, \ldots, y_s\}$ that optimally encodes f. Then, every auxiliary variable y_i appearing in F must appear in at least 2 different clauses as the literal y_i and in at least 2 different clauses as the literal $\overline{y_i}$.*

Proof. First, note that F cannot contain any tautological clauses, as otherwise removing them would contradict the optimality of F as encoding for f. Suppose an auxiliary variable y_i that appears in the formula does so only once as a literal $\ell \in \{y_i, \overline{y_i}\}$, and appears in $t \geqslant 0$ different clauses as $\overline{\ell}$. Then, we can use the resolution rule to obtain the formula F_ℓ of size

$$|F_\ell| = |F| - |F(\ell)| - |F(\overline{\ell})| + |F(\ell)| \cdot |F(\overline{\ell})| = |F| - 1 - t + 1 \cdot t = |F| - 1.$$

If we prove that F_ℓ encodes f, we will reach our desired contradiction, since F was assumed to be an optimal encoding for f and $|F_\ell| < |F|$. Indeed, let τ be a $\{x_1, \ldots, x_n\}$-assignment. Then, whether $\mathrm{SAT}(F_\ell|_\tau)$ or not is equivalent to the existence of a $\{x_1, \ldots, x_n, y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_s\}$-assignment σ that extends τ and such that $\sigma \models F_\ell$. But by Lemma 4, $\sigma \models F_\ell \iff \mathrm{SAT}(F|_\sigma)$, so we conclude

$$\begin{aligned}
\mathrm{SAT}(F_\ell|_\tau) &\iff \exists \{x_1, \ldots, x_n, y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_s\}\text{-assignment } \sigma \text{ extending } \tau \text{ s.t. } \mathrm{SAT}(F|_\sigma) \\
&\iff \mathrm{SAT}(F|_\tau) \\
&\iff f(\tau(x_1), \ldots, \tau(x_n)) = \top,
\end{aligned}$$

where the last bi-implication follows from the assumption that F encodes f. □

Now, the key insight to prove a lower bound on $a(n)$ is that its helpful to have a lower bound on $e(n)$ first. Thus, we start by the following theorem.

Theorem 6. *For every $n \geqslant 1$, we have $e(n) \geqslant \min(3n - 6, 2n)$.*

Proof. The proof is by induction on n, with the base cases $n \leqslant 2$ holding trivially since then $3n - 6 \leqslant 0$ for $n \leqslant 2$. For $3 \leqslant n \leqslant 5$ it can be checked that $e(n) \geqslant 3n - 6$ by computational case analysis.

Now, assume the statement is true for $n \geqslant 6$, and let us prove that it is true for $n + 1$. Let F be any formula such that F is an encoding of $\mathsf{ExactlyOne}(x_1, \ldots, x_{n+1})$, and $|F| = e(n + 1)$. We now split the proof into two cases:

Case I. There is some variable x_i such that the literal $\overline{x_i}$ appears in 2 or more different clauses of F.

Case II. No such variable exists.

Let us start by addressing **Case I**, with x_i being the variable predicated in the case. Consider the $\{x_i\}$-assignment τ such that $\tau(x_i) = \bot$. The formula $F' := F|_\tau$ corresponds therefore to removing from F all clauses containing $\overline{x_i}$, and removing x_i from all remaining clauses containing it. Since by assumption $\overline{x_i}$ appears in 2 or more different clauses of F, we have $|F'| \leqslant |F| - 2$. We now need the following observation.

Remark 7. The formula F' is an encoding of $\mathsf{ExactlyOne}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n+1})$.

Proof of Remark 7. Note first that any $\{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n+1}\}$-assignment σ, can be extended into a unique $\{x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_{n+1}\}$-assignment σ' with $\sigma'(x_i) = \bot$. Then, it holds that

$$F'|_\sigma = F|_{\sigma'}. \tag{3}$$

We can now easily finish the proof of this remark. Indeed, if some $\{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n+1}\}$-assignment σ satisfies exactly one of its variables, then so does σ', and thus by Definition 3 we have that $F|_{\sigma'}$ is satisfiable, which implies that $F'|_\sigma$ is satisfiable by Equation (3). Conversely, if some $\{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n+1}\}$-assignment σ does not satisfy exactly one of its variables, then it satisfies either 0 or more than 1. In either case, σ' also does not satisfy exactly one of its variables, and thus by Definition 3 we have that $F|_{\sigma'}$ is unsatisfiable, which by Equation (3) implies that $F'|_\sigma$ is unsatisfiable.

(End of proof of Remark 7) ■

Using the inductive hypothesis on top of Remark 7 we have that $|F'| \geqslant 2n$, and as we had $|F'| \leqslant |F| - 2$, we conclude that

$$|F| \geqslant |F'| + 2 \geqslant \min(3n - 6, 2n) + 2 = 2n + 2 = 2(n + 1),$$

since for $n \geqslant 6$ we have $3n - 6 \geqslant 2n$. This finishes the analysis of **Case I**, and it now remains for us to address **Case II**.

By virtue of being on **Case II**, the literal $\overline{x_{n+1}}$ appears either 0 or 1 times in F. It is easy to see that it cannot appear 0 times, as otherwise F could not distinguish between an $\{x_1, \ldots, x_{n+1}\}$-assignment γ that sets only $\gamma(x_1) = \top$ and an $\{x_1, \ldots, x_{n+1}\}$-assignment γ' that sets only $\gamma'(x_{n+1}) = \gamma'(x_1) = \top$. Therefore, $\overline{x_{n+1}}$ appears exactly once, in a clause $C := (\overline{x_{n+1}} \vee C')$. We can readily observe that C' cannot contain other x_i variables: if it contained a literal x_i, then by taking the assignment γ such that $\gamma(x_i) = \gamma(x_{n+1}) = \top$, and $\gamma(x_j) = \bot$ for $j \notin \{i, n+1\}$, we would have that $F|_\gamma$ is satisfiable, which is a contradiction. If C' contained a literal $\overline{x_i}$, then the assignment γ that sets $\gamma(x_{n+1}) = \gamma(x_k) = \top$ for some $k \notin \{i, n+1\}$ would make $F|_\gamma$ satisfiable, which is also a contradiction. Thus, we infer that $C := (\overline{x_{n+1}} \vee \ell_1 \vee \ldots \vee \ell_k)$ with each ℓ_i being an auxiliary variable y_j or its negation. Then, let $\tau^\star$ be the $\{x_1, \ldots, x_{n+1}\}$-assignment defined by setting $\tau^\star(x_{n+1}) = \top$, and $\tau^\star(x_i) = \bot$ for $i \neq n+1$. We thus have that $F|_{\tau^\star}$ is satisfiable, and we let $\sigma^\star$ be a satisfying assignment. Then, since $\sigma^\star \models C$, but $\sigma^\star \not\models \overline{x_{n+1}}$, we have that $\sigma^\star \models \ell_i$ for some $1 \leqslant i \leqslant k$. We take an arbitrary such ℓ_i, and define the formula $F^\dagger$ to be exactly like F but with the clause C replaced by $(\overline{x_{n+1}} \vee \ell_i)$. We now claim the following:

Remark 8. The formula $F^\dagger$ is an encoding of $\mathsf{ExactlyOne}(x_1, \ldots, x_n, \ell_i)$.

Proof of Remark 8. Let τ be any $\{x_1, \ldots, x_n, \ell_i\}$-assignment. Assume for the forward direction that τ assigns exactly one of its variables to $\top$. If said variable is some x_j, then we have $\tau(\ell_i) = \bot$. In such a case, let τ' be a $\{x_1, \ldots, x_{n+1}\}$-assignment defined by

$$\tau'(x_k) = \begin{cases} \tau(x_k) & \text{if } k \neq n+1 \\ \bot & \text{if } k = n+1. \end{cases}$$

As τ' assigns exactly one of the $\{x_1, \ldots, x_{n+1}\}$ variables, we have by Definition 3 that $F|_{\tau'}$ is satisfiable, and thus there is an assignment σ that extends τ' such that $\sigma \models F$. We also have $\sigma \models F^\dagger$, since the only clause in which F and $F^\dagger$ differ is C, which is satified by σ since $\sigma(x_{n+1}) = \bot$. We now prove by contradiction that $\sigma(\ell_i)$ must be $\bot$, since otherwise the $\{x_1, \ldots, x_{n+1}, y_1, \ldots, y_s\}$-assignment σ' defined by

$$\sigma'(\ell) = \begin{cases} \sigma(\ell) & \text{if } \ell \neq x_{n+1} \\ \top & \text{if } \ell = x_{n+1} \end{cases}$$

would also hold $\sigma' \models F$, as the only difference between σ' and σ is that $\sigma'(x_{n+1}) = \top$, which could only fail to satisfy the clause C as that is the only clause in which $\overline{x_{n+1}}$ appears. But since we assumed

expecting a contradiction that $\sigma(\ell_i) = \top$, we have that $\sigma' \models C$. Thus, we have proven that $\sigma(\ell_i) = \bot$, and therefore σ is an assignment extending τ and such that $\sigma \models F^\dagger$, which concludes this case.

The case in which the variable assigned to $\top$ by τ is ℓ_i follows more easily: We consider the assignment $\sigma^\star$ defined right before this remark, and note that $\sigma^\star \models F^\dagger$ as well as that $\sigma^\star$ extends τ, which concludes this case.

We now have to prove the backward direction, in which we assume that τ does not assign exactly one of its variables to $\top$. If τ assigns 2 different x-variables to $\top$, then trivially $F|_\tau$ is unsatisfiable, and in consequence so is $F^\dagger|_\tau$. If τ assigns one x-variable to $\top$ and also ℓ_i to $\top$, then again if F was satisfied by some assignment σ extending τ, defining σ' by

$$\sigma'(\ell) = \begin{cases} \sigma(\ell) & \text{if } \ell \neq x_{n+1} \\ \top & \text{if } \ell = x_{n+1} \end{cases}$$

would yield a satisfying assignment for F that assigns 2 different x-variables to $\top$, a contradiction. Thus, $F|_\tau$ is unsatisfiable and in consequence so is $F^\dagger|_\tau$. The only case remaining is to show that when τ assigns zero of its variables to $\top$, we have that $F^\dagger|_\tau$ is unsatisfiable. Indeed, for $F^\dagger|_\tau$ to be satisfiable, its satisfying assignment σ would have to assign $\sigma(x_{n+1}) = \top$, as otherwise we would have that σ is a satisfying assignment of $F^\dagger$ that extends a $\{x_1, \ldots, x_{n+1}\}$-assignment with zero variables set to $\top$, which would imply a satisfying assignment of F extending $\{x_1, \ldots, x_{n+1}\}$-assignment with zero variables set to $\top$, a contradiction. But if $\sigma(x_{n+1}) = \top$ and since $\sigma \models F^\dagger$, we have $\sigma \models (\overline{x_{n+1}} \vee \ell_i)$, and thus $\sigma(\ell_i) = \top$, which contradicts the assumption that τ assigned zero of its variables to $\top$. This concludes the backward direction and thus the proof of this remark. *(End of proof of Remark 8)* ■

With Remark 8 we have the necessary ingredients to finish **Case II**: $F^\dagger$ is an encoding of

$$\mathsf{ExactlyOne}(x_1, \ldots, x_n, \ell_i),$$

and $|F^\dagger| = |F| = e(n+1)$. But x_{n+1} is an auxiliary variable for $F^\dagger$, and by using Lemma 5, since $F^\dagger$ is an optimal encoding, we have that variable x_{n+1} must appear on at least two different clauses as the literal $\overline{x_{n+1}}$, which contradicts the assumption of **Case II**. Therefore, only **Case I** can arise, which completes our proof. □

Corollary 9. *For every $n \geqslant 1$, we have $a(n) \geq \min(3n-5, 2n-1)$.*

Proof. It suffices to note that $\mathsf{ExactlyOne}(x_1, \ldots, x_n)$ can be written as $\mathsf{AtMostOne}(x_1, \ldots, x_n) \wedge \left(\bigvee_{i=1}^n x_i\right)$, and thus an encoding for $\mathsf{AtMostOne}(x_1, \ldots, x_n)$ with fewer than $\min(3n-5, 2n-1)$ would yield an encoding of $\mathsf{ExactlyOne}(x_1, \ldots, x_n)$ with fewer than $\min(3n-6, 2n)$ clauses, contradicting Theorem 6. □

C. Deferred Calculations

While Chen's analysis of his encoding is essentially correct, it is rather informal [Che10]. For the sake of completeness, let us restate and prove the corresponding theorem.

Proposition 10 ([Che10]). *$C_{PE}(n) = 2n + 4\sqrt{n} + o(\sqrt{n})$ and $A_{PE}(n) = 2\sqrt{n} + o(\sqrt{n})$.*

Proof. Let us first prove a worse bound by a simple induction: $(\dagger) : C_{\mathsf{PE}}(n) \leqslant 4n$ for every $n \geqslant 1$. Indeed, recall that

$$C_{\mathsf{PE}}(n) = \begin{cases} \binom{n}{2} & \text{if } n \leqslant 4 \\ 2n + 2C_{\mathsf{PE}}\left(\lceil \sqrt{n} \rceil\right) & \text{otherwise} \end{cases}$$

So for $n \leqslant 4$, $C_{\mathsf{PE}}(n) = \binom{n}{2} \leqslant 4n$, and similarly, it can be checked computationally that $C_{\mathsf{PE}}(n) \leqslant 4n$ for $5 \leqslant n \leqslant 64$. We thus assume $n \geqslant 64$, from where

$$C_{\mathsf{PE}}(n) = 2n + 2C_{\mathsf{PE}}\left(\lceil \sqrt{n} \rceil\right) \overset{\text{I.H.}}{\leqslant} 2n + 8\lceil \sqrt{n} \rceil \leqslant 2n + 8(\sqrt{n} + 1) \leqslant 2n + \sqrt{n}(\sqrt{n} + 1) = 3n + \sqrt{n} \leqslant 4n.$$

Having proven $(\dagger)$, we have for any $n \geqslant 25$ that

$$\begin{aligned} C_{\mathsf{PE}}(n) &= 2n + 2C_{\mathsf{PE}}\left(\lceil \sqrt{n} \rceil\right) \\ &= 2n + 2\left(2\lceil \sqrt{n} \rceil + 2C_{\mathsf{PE}}\left(\left\lceil \sqrt{\lceil \sqrt{n} \rceil} \right\rceil\right)\right) && \text{(Since } n \geqslant 25, \lceil \sqrt{n} \rceil \geqslant 5\text{)} \\ &= 2n + 4\lceil \sqrt{n} \rceil + 4C_{\mathsf{PE}}\left(\left\lceil \sqrt{\lceil \sqrt{n} \rceil} \right\rceil\right) \\ &\leqslant 2n + 4\sqrt{n} + 4 + 16\left\lceil \sqrt{\lceil \sqrt{n} \rceil} \right\rceil && \text{(By } (\dagger)\text{)} \\ &= 2n + 4\sqrt{n} + o(\sqrt{n}). \end{aligned}$$

The case of $A_{\mathsf{PE}}(n)$ can be proved with the same methodology.

□