New Bounds on the Multiplicative Complexity of Boolean Functions

Meltem Sönmez Turan
National Institute of Standards and Technology

Abstract

Multiplicative Complexity (MC) is defined as the minimum number of AND gates required to implement a function with a circuit over the basis \{AND, XOR, NOT\}. This complexity measure is relevant for many advanced cryptographic protocols such as fully homomorphic encryption, multi-party computation, and zero-knowledge proofs, where processing AND gates is more expensive than processing XOR gates. Although there is no known asymptotically efficient technique to compute the MC of a random Boolean function, bounds on the MC of Boolean functions are successfully used to show existence of Boolean functions with a particular MC. In 2000, Boyar et al. [1] showed that, for all \(n \geq 0\), at most \(2^{k^2 + 2k + 2kn + n + 1}\) \(n\)-variable Boolean functions can be computed with \(k\) AND gates. This bound is used to prove the existence of a 8-variable Boolean functions with MC greater than 7. In this paper, we improve the Boyar et al. bound.

1 Introduction

Multiplicative Complexity (MC) is defined as the minimum number of AND gates required to implement a function with a circuit over the basis \{AND, XOR, NOT\}. This complexity measure is relevant for many advanced cryptographic protocols (e.g., [2]), fully homomorphic encryption (e.g., [3]), and zero-knowledge proofs (e.g., [4]), where processing nonlinear gates such as AND, NAND, is more expensive than processing linear gates such as XOR. These protocols benefit from new symmetric-key primitives that can be implemented with small number of AND gates (e.g., Rasta [5], LowMC [6]).

There is no known asymptotically efficient technique to compute the MC of a random Boolean function. Boyar et al. [4] showed that the MC of an \(n\)-variable random Boolean function is at least \(2^{n/2} - O(n)\) with high probability. For arbitrary \(n\), it is known that under standard cryptographic assumptions, it is not possible to compute the MC in polynomial time in the length of the truth table [7]. The degree bound states that the MC of a Boolean function having degree \(d\) is at least \(d - 1\) [8]. This bound is commonly used to prove that a given Boolean function implementation is optimal.

For up to 6 variables, the MC of each Boolean function has been established in [9, 10]. There are also known bounds for special classes of Boolean functions. The MC of affine Boolean functions is zero. Mirwald and Schnorr [11] showed that the MC of a quadratic function \(f\) is \(k\), iff \(f\) is affine equivalent to the canonical form \(\bigoplus_{i=1}^{k} x_{2i-1} x_{2i}\). This implies the MC of quadratic functions is at most \(\lceil \frac{n}{2} \rceil\). Turan and Peralta [12] improved the bounds on MC of cubic Boolean functions. Brandão et al. [13] studied the MC of symmetric Boolean functions and constructed circuits for all such functions with up to 25 variables. In 2017, Find et al. [14] characterized the Boolean functions with MC 2 by using the fact
that MC is invariant with respect to affine transformations. In 2020, Çalık et al. extended the result to Boolean functions with MC up to 4 [15]. In 2022, Häner and Soeken [16] showed the MC of interval checking.

A particular value of interest is the number of \( n \)-variable Boolean functions with MC \( k \), denoted \( \lambda(n, k) \). In [1], it is shown that \( \lambda(n, k) \leq 2^{k^2 + 2k + 2kn + n + 1} \). Using this bound, it is easy to see that 7 AND gates are not enough to compute all 8-variable Boolean functions, i.e., there exists 8-variable Boolean functions with MC at least 8. In 2002, Fischer and Peralta [17] showed that \( \lambda(n, 1) \) is equal to \( 2^{2^n} \). In 2017, Find et al. [14] showed that \( \lambda(n, 2) = 2^{n}(2^n - 1)(2^n - 2)(2^n - 4) \left( \frac{2}{21} + \frac{2^n - 8}{12} + \frac{2^n - 8}{360} \right) \).

Çalık and Turan [15] studied the Boolean functions with MC 3 and 4, and provided a closed formula for \( \lambda(n, 3) \) and \( \lambda(n, 4) \), by summing the sizes of all the affine equivalence classes with MC 3 (total of 24 classes) and 4 (total of 1277 classes).

For large values of \( n \) and \( k \), the bound \( \lambda(n, k) \leq 2^{k^2 + 2k + 2kn + n + 1} \) is essentially tight, but it is unclear to what extent this is true for small constant values of \( k \). In this paper, we improve the Boyar et al. bound and provide new bounds on the maximum multiplicative complexity for \( n \)-variable Boolean functions.

## 2 Preliminaries

### 2.1 Boolean functions

Let \( \mathbb{F}_2 \) be the finite field with two elements. An \( n \)-variable Boolean function \( f \) is a mapping from \( \mathbb{F}_2^n \) to \( \mathbb{F}_2 \). Let \( B_n \) be the set of \( n \)-variable Boolean functions and \( B_n^c \) be the set of \( n \)-variable cubic Boolean functions.

The algebraic normal form (ANF) of \( f \) is the multivariate polynomial 
\[
    f(x_1, \ldots, x_n) = \sum_{u \in \mathbb{F}_2^n} a_u x^u,
\]

where \( a_u \in \mathbb{F}_2 \) and \( x^u = x_1^{u_1} x_2^{u_2} \cdots x_n^{u_n} \) is a monomial containing the variables \( x_i \) where \( u_i = 1 \). The degree of the monomial \( x^u \) is the number of variables appearing in \( x^u \). The degree of a Boolean function, denoted \( \deg(f) \), is the highest degree among the monomials appearing in its ANF.

Two functions \( f, g \in B_n \) are affine equivalent if \( f \) can be written as
\[
    f(x) = g(Ax + a) + b^T x + c,
\]

where \( A \) is a non-singular \( n \times n \) matrix over \( \mathbb{F}_2 \), \( a, b \) are column vectors in \( \mathbb{F}_2^n \), and \( c \in \mathbb{F}_2 \). We use \( [f] \) to denote the affine equivalence class of the function \( f \). Degree and MC are invariant under affine transformations.

### 2.2 Boolean Circuits

A Boolean circuit \( C \) with \( n \) inputs and \( m \) outputs is a directed acyclic graph, where the inputs and the gates are the nodes, and the edges correspond to the Boolean-valued wires. The fanin and fanout of a node is the number of wires going in and out of the node, respectively. The nodes with fanin zero are called the input nodes and are labeled with an input variable from \( \{x_1, \ldots, x_n\} \). The circuits considered in this study only contain gates from the complete basis \{AND, XOR, NOT\} and have exactly one node with fanout zero (i.e., \( m = 1 \)), which is called the output node. For our purposes, we assume AND gates have fan-in two, but XOR gates have arbitrary fan-in > 0.
3 Number of Boolean functions with MC \( k \)

3.1 Previous results

Boyar et al. [1] showed that \( \lambda(n, k) \leq 2^{k^2 + 2k + 2kn + n + 1} \), for all \( n \geq 0 \). To compute this bound, the authors considered an abstraction of Boolean circuits having binary AND gates and XOR gates with unbounded inputs. Each AND gate is assumed to input a subset of input variables, outputs of AND gates (that are topologically located before the gate) and the constant function \( 1 \), i.e., for the \( i \)th AND gate \( a_i \) the (right and the left) input is subset of \( \{ x_1, x_n, a_1, a_2, \ldots, a_{i-1}, 1 \} \). Hence, for \( a_i \), there are \( 2^{n}+1+\binom{i}{2} \) possible choices for its left and right inputs. The bound increases by \( 2(n+k) + 3 \) for each addition of the new AND gate to the circuit.

The exact values of \( \lambda(n, k) \) are known for the following values of \( n \) and \( k \):

- \( \lambda(n, 1) = 2^{\binom{n}{3}} \)
- \( \lambda(n, 2) = 2^n(2^n-1)(2^n-2)(2^n-4) \left( \frac{2^n}{12} + \frac{2^n-8}{360} \right) \).
- \( \lambda(n, 3) = \sum_{d=4}^{6} \left( 2^{n-d} \prod_{i=0}^{d-1} \frac{2^n-2^i}{2^n-2} \beta(d, 3) \right) \) where
  \[ \beta(4, 3) = 32768, \]
  \[ \beta(5, 3) = 775728128, \]
  \[ \beta(6, 3) = 183894007808. \]
- \( \lambda(n, 4) = \sum_{d=5}^{8} \left( 2^{n-d} \prod_{i=0}^{d-1} \frac{2^n-2^i}{2^n-2} \beta(d, 4) \right) \) where
  \[ \beta(5, 4) = 3515396096, \]
  \[ \beta(6, 4) = 7944313921970176, \]
  \[ \beta(7, 4) = 8217135092528316, \]
  \[ \beta(8, 4) = 5502415308673798144. \]

Table 1 shows the gap between the Boyar et al. bound and the exact number of functions with MC up to 4.

<table>
<thead>
<tr>
<th>MC</th>
<th>Bound</th>
<th>n = 6</th>
<th>n = 7</th>
<th>n = 8</th>
<th>n = 9</th>
<th>n = 10</th>
<th>n = 11</th>
<th>n = 12</th>
<th>n = 13</th>
<th>n = 14</th>
<th>n = 15</th>
<th>n = 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Exact</td>
<td>16.34</td>
<td>19.38</td>
<td>22.38</td>
<td>25.40</td>
<td>28.41</td>
<td>31.41</td>
<td>34.41</td>
<td>37.41</td>
<td>40.41</td>
<td>43.41</td>
<td>46.41</td>
</tr>
<tr>
<td>1</td>
<td>Bound</td>
<td>22</td>
<td>25</td>
<td>28</td>
<td>31</td>
<td>34</td>
<td>37</td>
<td>40</td>
<td>43</td>
<td>46</td>
<td>49</td>
<td>52</td>
</tr>
<tr>
<td>2</td>
<td>Exact</td>
<td>26.13</td>
<td>31.30</td>
<td>36.38</td>
<td>41.42</td>
<td>46.44</td>
<td>51.45</td>
<td>56.45</td>
<td>61.45</td>
<td>66.46</td>
<td>71.46</td>
<td>76.46</td>
</tr>
<tr>
<td>2</td>
<td>Bound</td>
<td>39</td>
<td>44</td>
<td>49</td>
<td>54</td>
<td>59</td>
<td>64</td>
<td>69</td>
<td>74</td>
<td>79</td>
<td>84</td>
<td>89</td>
</tr>
<tr>
<td>3</td>
<td>Exact</td>
<td>38.03</td>
<td>45.64</td>
<td>52.92</td>
<td>60.05</td>
<td>67.12</td>
<td>74.15</td>
<td>81.17</td>
<td>88.18</td>
<td>95.18</td>
<td>102.18</td>
<td>109.18</td>
</tr>
<tr>
<td>3</td>
<td>Bound</td>
<td>58</td>
<td>65</td>
<td>72</td>
<td>79</td>
<td>86</td>
<td>93</td>
<td>100</td>
<td>107</td>
<td>114</td>
<td>121</td>
<td>128</td>
</tr>
<tr>
<td>4</td>
<td>Exact</td>
<td>52.81</td>
<td>63.15</td>
<td>71.94</td>
<td>80.29</td>
<td>88.46</td>
<td>96.56</td>
<td>104.63</td>
<td>112.70</td>
<td>120.82</td>
<td>129.02</td>
<td>137.35</td>
</tr>
<tr>
<td>4</td>
<td>Bound</td>
<td>79</td>
<td>88</td>
<td>97</td>
<td>106</td>
<td>115</td>
<td>124</td>
<td>133</td>
<td>142</td>
<td>151</td>
<td>160</td>
<td>169</td>
</tr>
</tbody>
</table>

Table 1: Number of Boolean functions with MC 1, 2, 3, and 4 compared to the Boyar et al. bound [1] on a log scale with base 2

3.2 Improving the Boyar et al. bound

Next, we present two observations on Boolean circuits to improve the bound.
1. **Elimination of equivalent inputs** Let \( f_1 \) and \( f_2 \) be \( n \)-bit Boolean functions representing the left and right inputs to an AND gate, respectively, and let \( f_3 \) be the Boolean function XORed to the output to the AND gate. The output of the AND gate is \( f_1 \cdot f_2 + f_3 \). It is easy to see that the following inputs also produce the same output as \( f_1, f_2, \) and \( f_3 \).

\[
\begin{align*}
(f_1, f_2, f_3) & \rightarrow f_1 \cdot f_2 + f_3 \\
(f_2, f_1, f_3) & \rightarrow f_1 \cdot f_2 + f_3 \\
(f_1 + f_2, f_2, f_3 + f_2) & \rightarrow f_1 + f_2 + f_2 + f_3 = f_1 + f_2 + f_3 \\
(f_2, f_1 + f_2, f_3 + f_2) & \rightarrow f_2 + f_1 + f_2 + f_3 = f_1 + f_2 + f_3 \\
(f_1, f_2 + f_1, f_3 + f_1) & \rightarrow f_2 + f_1 + f_1 + f_3 + f_1 = f_1 + f_2 + f_3 \\
(f_2 + f_1, f_1, f_3 + f_1) & \rightarrow f_2 + f_1 + f_1 + f_3 + f_1 = f_1 + f_2 + f_3
\end{align*}
\]

In the Boyar et al. bound, each of these cases are counted separately.

2. **Elimination of the constant 1 function.** Boolean functions can be partitioned into those \( f \) for which \( f(0) = 0 \) and those \( f \) for which \( f(0) = 1 \). One set can be mapped bijectively into the other by the transformation \( g(x) = f(x) + 1 \). A function \( f(x) \) for which \( f(0) = 0 \) can be computed by a circuit which is both optimal with respect to multiplicative complexity and has no negations. Thus, considering circuits that do not have the constant 1 as input would produce the same set of Boolean functions. Boyar et al. bound computes the two inputs \( (f_1, f_2, f_3) \), and \( (f_1, f_2 + 1, f_3 + f_1) \) separately, although they both results in the same output.

\[
\begin{align*}
(f_1, f_2, f_3) & \rightarrow f_1 \cdot f_2 + f_3 \\
(f_1 + 1, f_2, f_3 + f_2) & \rightarrow f_1 \cdot f_2 + f_2 + f_3 = f_1 + f_2 + f_3 \\
(f_1, f_2 + 1, f_3 + f_1) & \rightarrow f_1 \cdot f_2 + f_1 + f_3 + f_1 = f_1 + f_2 + f_3 \\
(f_1 + 1, f_2 + 1, f_3 + f_1 + f_2) & \rightarrow f_2 \cdot f_1 + f_1 + f_2 + f_3 + f_1 + f_2 = f_1 + f_2 + f_3
\end{align*}
\]

Using the observations given above, the bound on the number of \( n \)-variable Boolean functions that can be generated using \( k \) AND gates, over the basis \( \{\text{AND, XOR, NOT}\} \), can be improved, by a factor of 24 for each AND gate. In other words, the function \( f_1 + f_2 + f_3 \) can be generated using 24 different choices of inputs for each AND gate, and instead of trying all 24 inputs (each combination of cases from (1) and (2)), it is possible to try only one of the inputs.

**Theorem 3.1** The number of \( n \)-variable Boolean functions that can be generated with \( k \)-AND gates is at most

\[
\lambda(n,k) \leq 2^{n+k+1} \prod_{i=1}^{k} \frac{1}{24} (2^{n+i+1})^2,
\]

\[
\leq 2^{k^2+2nk+n-k+1}3^{-k}.
\]

4. **Discussion**

In this note, we improved the Boyar et al. [1] the bound on the number of \( n \)-variable Boolean functions that can be generated using \( k \) AND gates by a factor of \( 2^{3k^2}3^k \) (by a
factor of 24 for each AND gate), which can be used to provide bounds on the maximum MC across all $n$-variable Boolean functions.

Acknowledgments

The author thanks L. Brandão, and R. Peralta for helpful discussions and suggestions. The author also thanks the anonymous reviewers for their useful comments.

References


