On the Minimum Number of Negations Leading to Super-polynomial Savings

Stasys Jukna *†‡§

Abstract

We show that an explicit sequence of monotone functions $f_n : \{0,1\}^n \to \{0,1\}^m$ ($m \leq n$) can be computed by Boolean circuits with polynomial (in n) number of And, Or and Not gates, but every such circuit must use at least $\log n - O(\log\log n)$ Not gates. This is almost optimal because results of Markov (1957) and Fisher (1974) imply that, with only small increase of the total number of gates, *any* circuit in n variables can be simulated by a circuit with at most $\lceil \log(n+1) \rceil$ Not gates.

1 Introduction

We consider Boolean circuits with And, Or and Not gates. Such a circuit is just a sequence $g_1, \ldots, g_t$ of Boolean functions where each g_i is either one of the input variables $x_1, \ldots, x_n$ or is obtained from the previous functions by applying an Or, And or Not operation (such applications are called *gates*). The length t of this sequence is the *size* of the circuit. A multi-output function $f : \{0,1\}^n \to \{0,1\}^m$ can be looked at as a sequence $f = (f_1, \ldots, f_m)$ of m Boolean functions, where for $\vec{a} \in \{0,1\}^n$, $f_i(\vec{a})$ is the i-th bit of $f(\vec{a})$. A circuit computes f if it contains all $f_1, \ldots, f_m$. The minimum number of Not gates in a circuit computing f us the *inversion complexity* of f.

More than forty years ago, Markov [7, 8] has found the following surprisingly tight combinatorial characterization of the inversion complexity. Given a chain in the binary n-cube (that is, a sequence $\vec{a}_1 < \vec{a}_2 < \ldots < \vec{a}_t$ of vectors in $\{0,1\}^n$), the decrease of $f = (f_1, \ldots, f_m)$ on this chain is the number of indices i such that $f(\vec{a}_i) \not\leq f(\vec{a}_{i+1})$ (that is, $f_j(\vec{a}_i) > f_j(\vec{a}_{i+1})$ for at least one $1 \leq j \leq m$). Taking the

*Universität Frankfurt, Institut für Informatik, D-60054 Frankfurt, Germany
†Institute of Mathematics and Informatics, LT-2600 Vilnius, Lithuania
‡Email: jukna@thi.informatik.uni-frankfurt.de
§Research supported in part by a DFG grant SCHN 503/2-1.

maximum over all chains we obtain the *decrease* $d(f)$ of f. Note that $0 \leq d(f) \leq n$ for every function f in n variables.

Theorem 1.1 (Markov [7, 8]). *For every function f, the minimum number of Not gates contained in a circuit computing f is precisely*

$$M(f) = \lceil \log(d(f) + 1) \rceil.$$

This, in particular, implies that *any* function in n variables can be computed by a circuit with at most $\lceil \log(n+1) \rceil$ negations, and for some functions, like the *inverter* $I_n(x) = (\neg x_1, \ldots, \neg x_n)$, so many negations are also necessary. Fischer [5] has shown that $\lceil \log(n+1) \rceil$ negations are also enough to compute $I_n(x)$ by a circuit of size $O(n^2 \log^2 n)$. His construction was subsequently improved in [14, 3]: the circuit constructed in [14] has size $O(n \log n)$ and depth $O(\log n)$. Thus, restricting the number of negations in a circuit to $\lceil \log(n+1) \rceil$ entails only a small blowup in circuit size: using de Morgan's lows we can push all negations to the inputs (this can increase the size by at most twice) and use the circuit for I_n to compute these inputs from $x_1, \ldots, x_n$.

Thus, if a function f_n has n variables and is *feasible*, i.e. can be computed by a circuit of polynomial in n size, then it remains feasible even if we allow at most $\lceil \log(n+1) \rceil$ Not gates. For the inverter $I_n(x)$ this is optimal: this function is feasible, and (according to Markov's theorem) it cannot be computed using fewer than $\lceil \log(n+1) \rceil$ Not gates. The reason here is that this function has maximal decrease, implying that $M(I_n) = \lceil \log(n+1) \rceil$. But what about functions whose decrease is much smaller than n? In particular, what about functions with $M(f_n) = 0$, i.e. about *monotone* functions? This motivates the following definition.

Definition 1.2. *For a feasible function f_n, let $\Delta(f_n)$ be the minimum number such that f_n can be computed by a circuit of polynomial size using $M(f_n)+\Delta(f_n)$ negations.*

Few words about the notation. As it is customary in circuit complexity theory, by a function in n variables we actually mean an infinite *sequence* $(f_n : n = 1, 2, \ldots)$ of functions with growing number n of variables. Hence, $\Delta(f_n) > r(n)$ means that for any constant $c > 0$ there exist infinitely many n such that every circuit computing f_n with $M(f_n) + r(n)$ negations has size larger than n^c.

We know that $\Delta(f_n) \leq \lceil \log(n+1) \rceil - M(f_n)$. But what about *lower* bounds on $\Delta(f_n)$? Berkowitz [4] and Valiant [16] have shown that for so-called *slice* functions (these are monotone Boolean functions which are non-trivial only on one slice of the n-cube) negations are powerless, i.e., *cannot* lead to a super-polynomial savings. Hence, there is a large class of feasible monotone functions f_n for which $\Delta(f_n) = 0$, and the first natural question was whether using Not gates can lead to super-polynomial savings in size *at all*?

This question was considered by several authors under additional restrictions on the topology and/or on the use of Not gates. In particular, Okolnishnikova [9], and Ajtai and Gurevich [1] have shown that there exists monotone functions that can be computed with polynomial size, *constant* depth circuits, but cannot be computed with *monotone*, polynomial size, constant depth circuits. Moreover, it was shown by Santha and Wilson [13] that in this class we may need much more than $\lceil \log(n+1) \rceil$ negations: *any* depth-d circuit computing the inverter $I_n(x)$ must use at least $\Omega(n/\log^{d+3} n)$ negations. (Note that this result does not contradict the Markov–Fischer upper bound: their simulation requires logarithmic depth.) Another line of research was to restrict the *use* of Not gates. Raz and Wigderson [10] have proved that, if we require that all the negations are placed on the input variables, then there is an explicit monotone function (corresponding to the connectivity problem for graphs) that can be computed with polynomial size, depth $O(\log^2 n)$ circuits, but can not be computed with polynomial size, depth $k \log n$ circuits using only $o(n/2^k)$ negated variables.

The case of general circuits (no restrictions on the topology or on the use of Not gates) was resolved by Razborov [11]: $\Delta(f) > 0$ for a very natural monotone Boolean function detecting a presence of a perfect matching in bipartite graphs. Another feasible function (defined using Lovász-capacity of graphs), for which presence of Not gates yields even exponential savings, was later given by Tardos [15]. After this progress it still remained unclear how to construct feasible functions f_n for which, say, $\Delta(f_n) > 1$.

In this note we show that, if we allow f_n be a multi-output function, then the above mentioned results [11, 15] *can* be used to move the threshold $\Delta(f_n)$ of the first super-polynomial decrease in size quite near to the Markov–Fischer upper bound $\lceil \log(n+1) \rceil - M(f_n)$.

2 The result

As mentioned above, $\Delta(f_n) \leq \lceil \log(n+1) \rceil - M(f_n)$ for *every* feasible function f_n in n variables. On the other hand, we have the following:

Theorem 2.1. *There is an explicit sequence $f_n : \{0,1\}^n \to \{0,1\}^m$ of feasible monotone functions with $m \leq n$ such that*

$$\Delta(f_n) \geq \log n - O(\log \log n).$$

The proof of this theorem is based on the following lemma allowing us to eliminate some Not gates without increasing the size of the circuit. Let $h = h(X)$ be a Boolean function in m variables $X = \{x_1, \ldots, x_m\}$. Let $k = 2^r$ and $n = km$. A function

$f : \{0,1\}^n \to \{0,1\}^k$ is an *r-fold extension* of h if it computes k copies of h on disjoint copies $X_1, \ldots, X_k$ of X. That is, given an input $(\vec{a}_1, \ldots, \vec{a}_k)$ with $\vec{a}_i \in \{0,1\}^m$, the function outputs the sequence $(h(\vec{a}_1), \ldots, h(\vec{a}_k))$. Important here is that the i-th output bit $h(\vec{a}_i)$ is independent of inputs $\vec{a}_j$ for $j \neq i$. Note also that the extensions of monotone functions are monotone. A *minterm* of a monotone Boolean function is a minimal set of variables which, if assigned the value 1, forces the function to take the value 1 regardless of the values assigned to the remaining variables.

Lemma 2.2. *Let h be a Boolean function, and r be a nonnegative integer. If the r-fold extension f of h can be computed by a circuit using at most s negations, then h can be computed by a circuit of the same size using at most $s - r$ negations.*

Proof. Given a circuit G which computes f and has r Not gates, we eliminate these gates one-by-one. To do this, consider the *first* Not gate in G, and let g be the monotone Boolean function computed at the input to this gate (i.e., immediately before this gate). Let Y_0 and Y_1 be the union of the first 2^{r-1} and, respectively, the last 2^{r-1} blocks of variables $X_1, \ldots, X_{2^r}$. Let $f_0(Y_0)$ and $f_1(Y_1)$ be the corresponding $(r-1)$-fold extensions of h.

Since g is monotone, all its minterms are positive (no negated literals). Hence, we have only two possibilities: either some minterm of g lies entirely in Y_1, or not. In the first case we assign constant 1 to all the variables in Y_1, whereas in the second case we assign constant 0 to all the variables in Y_0. As the function g is monotone, in both cases it turns to a constant function (constant 1 in the first case, and constant 0 in the second), and the subsequent Not gate can be eliminated.

Since the sets of variables Y_0 and Y_1 are disjoint, assigning constant ϵ to all the variables in Y_ϵ does not affect the function $f_{1-\epsilon}$. Hence, we obtain a circuit which computes an $(r-1)$-fold extension of h, and has one Not gate fewer. Repeating this argument r times we will obtain a circuit of the same (or smaller) size which computes h and has only $s - r$ Not gates. □ □

Proof of Theorem 2.1. Take the explicit monotone Boolean function T_m in m variables considered in [15]. As shown in [15], this function is feasible (can be computed by a circuit of size $m^{O(1)}$) but every monotone circuit computing it requires size at least $2^{cm^{1/8}}$ for some constant $c > 0$. Set $r = r(m) := \lfloor m^{1/9} - \log m \rfloor$ and consider the r-fold extension f_n of T_m. The function f_n is a monotone function in $n = 2^r m$ variables. By the choice of r and n, we have that $2^r = \Theta\left(n/(\log n)^9\right)$. Moreover, the function f_n is feasible since it can be computed by a circuit of size $2^r \cdot m^{O(1)} \leq n^{O(1)}$.

However, in order to do this we must use at least $\Delta(f_n) \geq r$ negations. Indeed, given a circuit G with at most s Not gates computing f_n, we can apply Lemma 2.2 and obtain a circuit G' of the same size which has at most $s - r$ negations and computes T_m. If $s \leq r$, then the circuit G' has no negations at all, implying that the circuit G',

and hence, the circuit G must have size at least 2^N where $N = cm^{1/8} = c(n/2^r)^{1/8} = \Omega\left((\log n)^{9/8}\right)$. Thus, $\Delta(f_n) > r \geq \log n - O(\log\log n)$. □

3 Concluding remarks

1. One may ask what happens if besides And, Or and Not gates we will allow, say, any monotone *real-valued* functions as gates—does then the use of negations can still lead to a drastic decrease of size? If yes, how many negations do we actually need to achieve such a decrease? The question makes sense because, using a counting argument, Rosenbloom [12] has shown that there exist monotone Boolean functions that can be computed by monotone real circuits (no Not gates) of linear size $O(n)$, but require Boolean circuits with And, Or and Not gates of size $2^{\Omega(n)}$. On the other hand, it was shown in [6] that Tardos' function T_m requires even monotone real-valued circuits of size $2^{\Omega(m^{1/8})}$. Since also in real-valued circuits the input to a Not gate must be a Boolean function, Lemma 2.2 holds also for such circuits. This means that the function f_n considered in Theorem 2.1 captures the role of negations in a quite strong sense: this function is feasible but cannot be computed by a circuit of polynomial size with fewer than $\log n - O(\log\log n)$ Not gates, even if we allow any monotone real-valued functions with two variables as gates.

2. In the proof of Theorem 2.1 we have essentially used the fact that the function f_n is a *multi-output* function. To prove a similar lower bound for a feasible Boolean (i.e. *single-output*) function remains an interesting open problem. A step in this direction was made by Amano and Maruoka [2] who modified Razborov's argument to show that the Clique function cannot be computed by circuits of polynomial size using only $c\log\log n$ negations. However, the measure $\Delta(f)$ makes sense only if the function f itself is *feasible*, and the Clique function is (most likely) not such.

3. So far, the only known way to reduce the number of Not gates is to compute the inverter $I_n(x_1,\ldots,x_n) = (\neg x_1,\ldots,\neg x_n)$ using only $\lceil\log(n+1)\rceil$ such gates ([5, 14, 3]). However, by Markov's theorem, *every* circuit computing I_n must contain $\lceil\log(n+1)\rceil$ negations. Hence, this type of simulation will *always* leave us with a circuit containing $\lceil\log(n+1)\rceil$ negations, even when the function f is such that the necessary number $M(f)$ of negations is much smaller. Is it possible to do more subtle simulation without computing I_n?

References

[1] M. Ajtai and Y. Gurevich (1987): Monotone versus positive, *J. of the ACM* **34**, 1004–1015.

[2] K. Amano and A. Maruoka (1988): A superpolynomial lower bound for a circuit computing the clique function with at most $(1/6)\log\log n$ negation gates, in: *Springer Lect. Notes in Comput. Sci.*, vol. 1450, 399–408.

[3] R. Beals, T. Nishino, and K. Tanaka (1998): On the complexity of negation-limited Boolean networks, *SIAM J. Comput.* **27**:5, 1334–1347.

[4] S.J. Berkowitz (1982): On some relationships between monotone and non-monotone circuit complexity. Technical Report, University of Toronto.

[5] M.J. Fischer (1974): The complexity of negation-limited networks–a brief survey, in: *Springer Lect. Notes in Comput. Sci.*, vol. 33, 71–82.

[6] S. Jukna (1999): Combinatorics of monotone computations, *Combinatorica* **19**:1, 65–85.

[7] A.A. Markov (1957): On the inversion complexity of systems of functions, *Doklady Academii Nauk SSSR*, **116**, 917–919 (in Russian). English translation in: *J. of ACM*, **5**:4 (1958), 331–334.

[8] A.A. Markov (1963): On the inversion complexity of systems of Boolean functions, *Doklady Academii Nauk SSSR*, **150**:3, 477–479 (in Russian). English translation in: *Soviet Math. Doklady* **4** (1963), 694–696.

[9] E.A. Okolnishnikova (1982): On the influence of negation on the complexity of realization of monotone Boolean functions by formulas of bounded depth, in: *Metody Diskretnogo Analiza* **38**, 74–80 (in Russian)

[10] R. Raz and A. Wigderson (1989): Probabilistic communication complexity of Boolean relations, in: *Proc. of 30th Ann. IEEE Symp. on Foundations of Comput. Sci.*, 562–567.

[11] A.A. Razborov (1985): A lower bound on the monotone network complexity of the logical permanent, *Matematicheskie Zametki*, **37**:6, 887–990 (in Russian). English translation in: *Math. Notes Acad. of Sci. USSR* **37**:6 (1985), 485–493.

[12] A. Rosenbloom (1997): Monotone real circuits are more powerful than monotone Boolean circuits, *Information Processing Letters* **61**, 161–164.

[13] M. Santha and Ch. Wilson (1993): Limiting negations in constant depth circuits, *SIAM J. Comput.* **22**:2, 294–302.

[14] K. Tanaka and T. Nishino (1994): On the complexity of negation-limited Boolean networks, in: *Proc. 26th Annual ACM Symposium on the Theory of Computing* (ACM, New-York), 38–47.

[15] É. Tardos (1987): The gap between monotone and non-monotone circuit complexity is exponential, *Combinatorica*, **7**:4, 141–142.

[16] L.G. Valiant (1986): Negation is powerless for Boolean slice functions, *SIAM J. Comput.* **15**, 531–535.