WHEN DO EXTRA MAJORITY GATES HELP? POLYLOG(N) MAJORITY GATES ARE EQUIVALENT TO ONE

RICHARD BEIGEL

Abstract. Suppose that a Boolean function $f$ is computed by a constant depth circuit with $2^m$ AND-, OR-, and NOT-gates—and $m$ majority-gates. We prove that $f$ is computed by a constant depth circuit with $2^m\Omega(1)$ AND-, OR-, and NOT-gates—and a single majority-gate, which is at the root.

One consequence is that if a Boolean function $f$ is computed by an AC$^0$ circuit plus polylog majority-gates, then $f$ is computed by a probabilistic perceptron having polylog order. Another consequence is that if $f$ agrees with the parity function on three-fourths of all inputs, then $f$ cannot be computed by a constant depth circuit with $2^{n^{o(1)}}$ AND-, OR-, and NOT-gates, and $n^{o(1)}$ majority-gates.

Key words. Majority-gate; threshold-gate; symmetric gate; circuit; parity.

Subject classifications. 68Q15.

1. Introduction

One of the goals of complexity theory is to find ways to reduce the use of one resource. Typically this entails a modest increase in some other resources.

Recently, quasipolynomial size circuits have been the setting for unexpected resource tradeoffs (Allender 1989, Allender & Hertrampf 1994, Yao 1990, Beigel & Tarui 1994, Tarui 1993, and Beigel et al. 1991). In this paper we show how to reduce the number of majority-gates in many kinds of quasipolynomial size circuits from polylog to 1.
For example, consider constant depth quasipolynomial size circuits that consist of AND-, OR-, NOT-, and majority-gates. We show how to reduce the number of majority-gates from \( m \) to 1, while increasing the number of other gates by \( 2^m \cdot \text{polylog} n \) and increasing the depth by 2. As a corollary we convert such circuits to probabilistic perceptrons having small order. As another corollary, we unify the lower bounds of Aspnes et al. (1991), Yao (1985), Hastad (1989), and Siu et al. (1993) for computing parity with constant depth circuits, proving that if a constant depth circuit consisting of AND-, OR-, NOT-, and majority-gates computes parity correctly on three-fourths of all inputs, then the circuit must have exponential size or a polynomial number of majority-gates.

Our result is an exponential improvement on the bounds of Beigel et al. (1994). The proof uses the low-degree polynomials developed in that paper and an observation of Fortnow & Reingold (1991) in a nontrivial way.

We also show how to reduce the number of symmetric gates in many kinds of quasipolynomial size circuits from \( O(\log \log n) \) to 1. Consider constant depth quasipolynomial size circuits that consist of AND-, OR-, NOT-, and \( \text{MOD}_k \)-gates and symmetric gates. We show how to reduce the number of symmetric gates from \( m \) to 1, while increasing the number of other gates by \( 2^m \cdot \text{polylog} n \) and increasing the depth by 1. The proof uses base-\( B \) representation in the same way as Papadimitriou & Zachos (1983). As a corollary we extend results of Yao (1990) and Beigel & Tarui (1994) on \( \text{ACC} \), showing that any function computed by a constant depth, quasipolynomial size circuit consisting of AND-, OR-, NOT-, and \( \text{MOD}_k \)-gates and \( O(\log \log n) \) symmetric gates is in fact computed by a depth 2, quasipolynomial size circuit with a symmetric gate at the root and AND-gates having polylog fanin at the bottom level.

Our results are for circuits with a single Boolean output gate. Amir et al. (1990) obtained contrasting results for circuits with multiple output gates. They proved in a very general setting that \( m + 1 \) majority-gates permit such circuits to compute more functions than only \( m \) majority-gates; in fact, their result holds not only for threshold-gates but for any nontrivial gate.

2. Representing Boolean functions

Boolean values are often represented as elements of \( \{0, 1\} \), 0 denoting false, and 1 denoting true. They can also be represented as elements of \( \{-1, 1\} \), -1 denoting false and 1 denoting true. Real polynomials or rational functions