Valiant's Polynomial-Size Monotone Formula for Majority

Oded Goldreich

July 11, 2011

Summary: This text provides an exposition of Valiant's proof of the existence of polynomial-size monotone formula for Majority. The exposition follows the main principles of Valiant's proof, but deviates from it in some details.

While it is easy to construct quasi-polynomial-size monotone formulae for majority (by relying on divide-and-conquer approaches)[1], it is less obvious how to construct polynomial-size formulae (let alone monotone ones; cf. [4] and the references there-in).

Notation. Suppose, for simplicity that n is odd, and consider the majority function $\texttt{MAJ} : \{0,1\}^n \to \{0,1\}$ defined as $\texttt{MAJ}(x) = 1$ if $\texttt{wt}(x) > n/2$ and $\texttt{MAJ}(x) = 0$ otherwise, where $\texttt{wt}(x) = \{i \in [n] : x_i = 1\}$ denotes the Hamming weight of $x = x_1 \cdots x_n$.

Theorem 1 *There exist polynomial-size monotone formulae for computing majority.*

The existence of polynomial-size (monotone) formulae is known to be equivalent to the existence of logarithmic-depth (monotone) circuits of bounded fan-in.[2] Anyhow, we shall prove the existence of logarithmic-depth monotone formulae (of bounded fan-in) for majority. Actually, two radically different proofs are known: The first proof uses a rather complicated construction of sorting networks of logarithmic depth [1, 2].[3] The second proof, presented below, uses the probabilistic method.

Proof: We prove the existence of logarithmic-depth monotone formulae (of bounded fan-in) for majority. The proof proceeds in two main steps. The first step consists of reducing the worst-case problem (i.e., of computing $\texttt{MAJ}$ on all inputs) to an average-case problem, denoted Π, where the point of the reduction is that it seems easier to cope with random inputs (than with all possible inputs). Specifically, we shall use a (simple) randomized reduction of the computation of $\texttt{MAJ}(x)$ to the computation of $\Pi(R(x))$, where $R(x)$ denotes the output of the reduction on input x. The key observation is that if the error probability is sufficiently low (i.e., lower than $2^{-|x|}$), then this randomized reduction yields a non-uniform reduction that is correct on all inputs. (Hence the existence of such a non-uniform reduction is proved by using the probabilistic method.) Next (i.e., in the second step), we show that a very simple (monotone) formula suffices for solving Π on random instances. Finally, composing the (monotone) reduction with the latter formula, we obtain the desired (monotone) formula.

[1]One way is using the recursion $\texttt{TH}_t(x'x'') = \vee_{i=0}^{t}(\texttt{TH}_i(x') \wedge \texttt{TH}_{t-i}(x''))$, where $\texttt{TH}_t(z) = 1$ iff $\texttt{wt}(z) \geq t$. Using $\texttt{MAJ}(x) = \texttt{TH}_{|x|/2}(x)$, this yields a size recursion of the form $S(n) = O(n) \cdot S(n/2)$, which solves to $S(n) = O(n)^{\log_2 n}$.

[2]One direction is almost trivial, for the other direction see [3].

[3]Sorting networks may be viewed as Boolean circuits with bit-comparison gates (a.k.a comperators), where each comperator is a (2-bit) sorting device. Observe that a comparator can be implemented by a small monotone circuit (i.e., $\texttt{comp}(x, y) = (\min(x, y), \max(x, y)) = ((x \wedge y), (x \vee y))$), and that the middle bit of the sorted sequence equals the majority value.

We start with the randomized reduction. Specifically, given an n-bit long input $x = x_1 \cdots x_n$, we consider a sequence of independent identically distributed 0-1 random variables $R(x) = (y_1, ..., y_m)$ such that for every $j \in [m]$ an index $i \in [n]$ is selected uniformly and y_j is set to x_i. Thus, $\Pr[y_i = 1] = \texttt{wt}(x)/n$. Now, let $F : \{0,1\}^m \to \{0,1\}$ be an arbitrary function. The key observation is captured by the following fact.

Fact 1.1 *If, for every $x \in \{0,1\}^n$, it holds that $\Pr[F(R(x)) = \texttt{MAJ}(x)] > 1 - 2^{-n}$, then there exists a choice of coin tosses ω for the random process R such that for every $x \in \{0,1\}^n$ it holds that $F(R_\omega(x)) = \texttt{MAJ}(x)$, where R_ω denotes the deterministic function obtained by fixing the coins of R to ω. Furthermore, for every fixed ω, the function R_ω just projects its input bits to fixed locations in its output sequence.*

(Here the probabilistic method is used to infer the existence of ω such that $F \circ R_\omega = \texttt{MAJ}$, based on $\Pr_\omega[(\forall x \in \{0,1\}^n)\ F(R_\omega(x)) = \texttt{MAJ}(x)] > 0$.) Note that, by the furthermore-clause, $F \circ R_\omega$ preserves the complexity and monotonicity of F.

Regarding the feasibility of the hypothesis of Fact 1.1 (i.e., $\Pr[F(R(x)) = \texttt{MAJ}(x)] > 1 - 2^{-n}$ for every x), consider the case that $m = \Theta(n^3)$ and $F = \texttt{MAJ}$. (This is merely a sanity check; we cannot afford using this F, because this would reduce the problem to itself on longer input length.) In this case, for every x it holds that

$$\Pr[\texttt{MAJ}(R(x)) = \texttt{MAJ}(x)] > \Pr\left[\left|\frac{\texttt{wt}(R(x))}{m} - \frac{\texttt{wt}(x)}{n}\right| < \frac{1}{2n}\right], \qquad (1)$$

which indeed is at least $1 - 2^{-n}$ (by Chernoff bound).

We now turn to the second step, which consists of presenting a monotone formula F of $O(\log n)$ depth. Generalizing the foregoing hypothesis, we wish F to satisfy the following condition: *If $Y_1, ..., Y_m$ are independent identically distributed 0-1 random variables such that for some b it holds that* $\Pr[Y_1 = b] \geq 0.5 + 1/2n$, *then* $\Pr[F(Y_1, ..., Y_m) = b] > 1 - 2^{-n}$. (For simplicity, we assume that m is a power of three.)

The construction uses a full ternary tree of depth $\ell = \log_3 m$, where internal vertices compute the majority of their three children. Specifically, let $\texttt{MAJ}_3$ denote the three-variable majority function. Then, $F_1(z_1, z_2, z_3) = \texttt{MAJ}_3(z_1, z_2, z_3)$ and for every $i \geq 1$

$$F_{i+1}(z_1, ..., z_{3^{i+1}}) = \texttt{MAJ}_3(F_i(z_1, ..., z_{3^i}), F_i(z_{3^i+1}, ..., z_{3^i+3^i}), F_i(z_{2\cdot 3^i+1}, ..., z_{3^{i+1}})). \qquad (2)$$

Finally, we let $F(z_1, ..., z_m) = F_\ell(z_1, ..., z_m)$.

The intuition is that each level in F amplifies the bias of the corresponding random variables (i.e., functions of $Y_1, ..., Y_m$) towards the majority value. This amplification is due to the amplification property of $\texttt{MAJ}_3$, which is stated next.

Fact 1.2 *Let Z_1, Z_2, Z_3 be three independent identically distributed 0-1 random variables, and let $p \stackrel{\text{def}}{=} \Pr[Z_1 = 1]$. Then:*

1. $p' \stackrel{\text{def}}{=} \Pr[\texttt{MAJ}_3(Z_1, Z_2, Z_3) = 1] = 3(1-p)p^2 + p^3$.

2. *Letting $\delta \stackrel{\text{def}}{=} p - 0.5$, it holds that $p' = 0.5 + (1.5 - 2\delta^2) \cdot \delta$.*

3. $p' < 3p^2$.

The three parts of the foregoing fact follow by straightforward calculations.[4]

The second part of Fact 1.2 asserts that if $p = 0.5 + \delta > 0.5$ and $\delta \leq \delta_0 < 0.5$, then $p' \geq 0.5 + (1.5 - 2\delta_0^2) \cdot \delta$, which means that the bias (i.e., $p - 0.5$) increases by a multiplicative factor in each iteration (until it exceeds δ_0). (Note that we assumed $p \geq 0.5 + 1/2n$, but similar considerations hold for $p \leq 0.5 - 1/2n$.)[5] This means that we can increase the bias from its initial level of at least $1/2n$ to any constant level of $\delta_0 < 1/2$, by using $\ell_1 = c_1 \cdot \log_2(2\delta_0 n)$ iterations of $\mathtt{MAJ}_3$, where $c_1 = 1/\log_2(1.5 - 2\delta_0^2)$.

The best result is obtained by using an arbitrary small $\delta_0 > 0$. In this case, we may use $c_1 \approx 1/\log_2(1.5) \approx 1.70951129$. Using $\ell_2 = O(1)$ additional iterations, we may increase the bias from δ_0 to, say, 0.4.

At this point, we use the third part of Fact 1.2, while considering the probability for a wrong majority value. In each such iteration, this probability is reduced from a current value of $1 - p$ to less than $3(1-p)^2$. Thus, using $\ell_3 = \log_2 n$ additional iterations, the probability of a wrong value reduces from $1 - (0.5 + 0.4) < 1/6$ to $3^{2^{\ell_3}-1} \cdot (1/6)^{2^{\ell_3}} < 2^{-2^{\ell_3}} = 2^{-n}$.

Letting $\ell = \ell_1 + \ell_2 + \ell_3 < 2.71 \log_2 n$ and $m = 3^\ell$, we obtain a formula $F = F_\ell$ on m variables, but this formula uses the non-standard $\mathtt{MAJ}_3$-gates. Yet, a $\mathtt{MAJ}_3$-gate can be implemented by a depth-three monotone formula (e.g., $\mathtt{MAJ}_3(z_1, z_2, z_3)$ equals $(z_1 \wedge z_2) \vee (z_2 \wedge z_3) \vee (z_3 \wedge z_1)$), and hence F is a monotone formula of depth $3\ell < 8.13 \log_2 n$. Note that if $Y_1, ..., Y_m$ are independent identically distributed 0-1 random variables such that for some b it holds that $\Pr[Y_1 = b] \geq 0.5 + 1/2n$, then $\Pr[F(Y_1, ..., Y_m) = 1 - b] < 2^{-n}$. Thus, for every $x \in \{0,1\}^n$ it holds that $\Pr_\omega[F(R_\omega(x)) = 1 - \mathtt{MAJ}(x)] < 2^{-n}$ and $\Pr_\omega[(\forall x \in \{0,1\}^n)\ F(R_\omega(x)) = \mathtt{MAJ}(x)] > 0$ follows. Hence, there exists a choice of ω such that $F \circ R_\omega$ computes the majority of n-bit inputs. ■

Comment. Interestingly, Valiant [4] obtains a better result by using an iterated construction that uses the function $V(z_1, z_2, z_3, z_4) = (z_1 \vee z_2) \wedge (z_3 \vee z_4)$ as the basic building block (rather than $\mathtt{MAJ}_3$). Since V is not a balanced predicate (i.e., $\Pr[V(U_4) = 1] = 9/16$), the random process used in [4] maps the string $x \in \{0,1\}^n$ to a sequence of independent identically distributed 0-1 random variables, $(y_1, ..., y_m)$, such that for every $j \in [m]$ the bit y_j is set to zero with some constant probability β (and is set to x_i otherwise, where $i \in [n]$ is uniformly distributed). The value of β is chosen such that if Z_1, Z_2, Z_3, Z_4 are independent identically distributed 0-1 random variables satisfying $\Pr[Z_1 = 1] = p \stackrel{\text{def}}{=} (1 - \beta)/2$, then $\Pr[V(Z_1, Z_2, Z_3, Z_4) = 1] = p$. It turns out that V amplifies deviation from p slightly better than $\mathtt{MAJ}_3$ does (w.r.t 1/2).[6] More importantly, V can be implemented by a monotone formula of depth two, whereas $\mathtt{MAJ}_3$ requires depth three. Thus, Valiant [4] performs $2.65 \log_2 n$ iterations (rather than $2.71 \log_2 n$ itertations), and obtains a formula of depth $5.3 \log_2 n$.

[4]For the second part, use $p' = (3 - 2p)p^2 = (3 - 1 - 2\delta) \cdot (0.25 + \delta + \delta^2)$, which implies $p' = 0.5 + 1.5\delta - 2\delta^3$.

[5]One way to see this is to define $p = \Pr[Z_1 = 0]$.

[6]This is surprising only if we forget that V takes four inputs rather than three.

Acknowledgments. Thanks to Alina Arbitman for her comments and suggestions regarding this write-up.

References

[1] M. Ajtai, J. Komlos, E. Szemerédi. An $O(n \log n)$ Sorting Network. In *15th ACM Symposium on the Theory of Computing*, pages 1–9, 1983.

[2] M.S. Paterson. Improved Sorting Networks with $O(\log N)$ Depth. *Algorithmica*, Vol. 5 (1), pages 75–92, 1990.

[3] P.M. Spira. On time hardware complexity trade-offs for Boolean functions. In the *4th Hawaii International Symposium on System Sciences*, pages 525–527, 1971.

[4] L.G. Valiant. Short Monotone Formulae for the Majority Function. *Journal of Algorithms*, Vol. 5 (3), pages 363–366, 1984.