On (Valiant's) Polynomial-Size Monotone Formula for Majority

Oded Goldreich

August 18, 2019

Abstract

This exposition provides a proof of the existence of polynomial-size monotone formula for Majority. The exposition follows the main principles of Valiant's proof (*J. Algorithms*, 1984), but deviates from it in some details. Specifically, we show that, with high probability,i a full ternary tree of depth $2.71 \log_2 n$ computes the majority of n values when assigning each leaf of the tree is assigned at random one of the n values.

This is a drastic revision of a text that was posted on the author's web-site in May 2011.[1] The original text was somewhat laconic, and the current revision was aimed to be more reader friendly.

1 The statement

It is easy to construct quasi-polynomial-size monotone formulae for majority by relying on divide-and-conquer approaches: For example, consider the recursion $\texttt{TH}_t(x'x'') = \vee_{i=0}^{t}(\texttt{TH}_i(x') \wedge \texttt{TH}_{t-i}(x''))$, where $\texttt{TH}_t(z) = 1$ if and only if $\texttt{wt}(z) \geq t$ (see notion below). 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 $S(n) = O(n)^{\log_2 n}$.

It is less obvious how to construct polynomial-size formulae (let alone monotone ones; cf. [5] and the references there-in). This exposition presents a variant of Valiant's classic proof of the existence of such formulae [5].

Theorem 1 (the classic theorem): *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] Hence, we shall focus on proving the existence of logarithmic-depth monotone formulae (of bounded fan-in) for majority.

We note that two radically different proofs are known for Theorem 1: The first proof uses the rather complicated construction of sorting networks of logarithmic depth [1, 3].[3] The second proof, presented below, uses the probabilistic method. Specifically, it combines a random projection of the n input bits to $m = \text{poly}(n)$ locations, and applying a simple formula to the resulting m-bit long string. Valiant's original proof uses a full binary tree with alternating AND and OR gates, whereas we shall use a full ternary tree with 3-WAY MAJORITY gates.

[1] See `http://www.wisdom.weizmann.ac.il/~oded/PDF/mono-maj.pdf`

[2] One direction is almost trivial, for the other direction see [4].

[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 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 of the original sequence.

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 $x = x_1 \cdots x_n$.

2 The proof

We prove the existence of logarithmic-depth monotone formulae (of bounded fan-in) for majority in two steps.

1. The first step consists of reducing the worst-case problem (i.e., of computing MAJ on all inputs) to the average-case problem of computing MAJ (on longer inputs), 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 $\texttt{MAJ}(R(x))$, where $R(x)$ denotes the output of the reduction on input x. The key observation is that if the error probability (of the average case solver on $R(x)$) 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.)

2. In the second step, we show that a very simple (monotone) formula suffices for solving MAJ on the average (w.r.t the distribution $R(x)$). Specifically, we shall use formulae obtained by iterating the three-way majority function; that is, the resulting formula isessentially a ternary tree of logarithmic depth with (3-way) majority gates in internal nodes and distinct variables in the leaves. A Boolean formula is obtained by a straightforward implementation of the three-way majority gates by depth-three formulae (of fan-in two)

Composing the (monotone) reduction with the latter formulae, we obtain the desired (monotone) formulae. Since the randomized reduction is merely a randomized projection (i.e., each output bit is assigned at random one of the input bits), the complexity of the final formulae equals the complexity of the formulae constructed in the second step.

The formula used in the second step is more intuitive than the one used by Valiant, which in turn also requires a slightly less natural randomized reduction; for details, see Section 3. Furthermore, our construction was instrumental for the subsequent work of Cohen *et. al.* [2].

2.1 The randomized reduction

Given an n-bit long input $x = x_1 \cdots x_n$, we consider a sequence of $m = \text{poly}(n)$ independent identically distributed 0-1 random variables $R(x) = (y_1, ..., y_m)$ such that $\mathsf{Pr}[y_j = 1] = \texttt{wt}(x)/n$ for each $j \in [m]$. In other words, for each $j \in [m]$, an index $i_j \in [n]$ is selected uniformly at random (independently of all other choices) and y_j is set to x_{i_j}.

Note that, for $m = \Omega(n^3)$, the following holds for every $x \in \{0,1\}^n$:

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

which is greater than $1 - 2^{-n}$ (by Chernoff bound). By itself, Eq. (1) is useless, since we have reduced the computation of an n-way majority to the computation of an $\Omega(n^3)$-way majority. But the point is that this is a worst-case to average-case reduction, and the average-case problem seems easier to solve; that is, we need to compute majority of typical sequences and need not worry about pathological ones. Note, however, that here average-case means being correct with probability greater than $1 - 2^{-n}$; we cannot afford straightforward error-reduction since it involves taking majority...

Before looking for a formula that computes majority in the foregoing average-case sense, let us see that having such a formula suffices. Indeed, let $F : \{0,1\}^m \rightarrow \{0,1\}$ be an arbitrary function that computes majority in the foregoing strong average-case sense. (Assuming that $0 < \mathsf{Pr}[F(R(x)) \neq \mathtt{MAJ}(x)] < 2^{-n}$, it must be that $m > n$).

Fact 2 (trivial derandomization): *Let $R : \{0,1\}^n \rightarrow \{0,1\}^m$ be a randomized process and $F : \{0,1\}^m \rightarrow \{0,1\}$ be a function. Suppose that, for every $x \in \{0,1\}^n$, it holds that $\mathsf{Pr}[F(R(x)) = \mathtt{MAJ}(x)] > 1 - 2^{-n}$. 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 $F(R_\omega(x)) = \mathtt{MAJ}(x)$, where R_ω denotes the residual function obtained by fixing the coins of R to ω.*

Turning back to the specific process R defined before, note that, for every fixed ω, the residual function R_ω just projects its input bits to fixed locations in its output sequence (i.e., letting $\omega = (i_1, ..., i_m) \in [n]^m$, it holds that $R_\omega(x_1x_2 \cdots x_n) = x_{i_1}x_{i_2} \cdots x_{i_m}$). Hence, $F \circ R_\omega$ preserves the depth (resp., size) complexity and monotonicity of F.

Proof: Using $\mathsf{Pr}[F(R(x)) \neq \mathtt{MAJ}(x)] < 2^{-n}$ (for every x), and applying a union bound, we get

$$\mathsf{Pr}_\omega[\exists x \in \{0,1\}^n \;\; F(R_\omega(x)) \neq \mathtt{MAJ}(x)] < 1,$$

and it follows that there exists ω such that $F(R_\omega(x)) = \mathtt{MAJ}(x)$ holds for every $x \in \{0,1\}^n$. ■

Digest: The probabilistic method is used to infer the existence of ω such that $F \circ R_\omega = \mathtt{MAJ}$, based on $\mathsf{Pr}_\omega[(\forall x \in \{0,1\}^n) \;\; F(R_\omega(x)) = \mathtt{MAJ}(x)] > 0$.

2.2 Solving the average-case problem

We now turn to the second step, which consists of presenting a monotone formula F of logarithmic depth that satisfies the hypothesis of Fact 2 (w.r.t the simple process R defined above). 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 $b \in \{0,1\}$ it holds that* $\mathsf{Pr}[Y_1 = b] \geq 0.5 + 1/2n$, *then* $\mathsf{Pr}[F(Y_1, ..., Y_m) = b] > 1 - 2^{-n}$.

The construction uses a full ternary tree of depth $\ell = \log_3 m$, where internal vertices compute the majority of their three children. (For simplicity, we assume that m is a power of three.) Specifically, let $\mathtt{MAJ}_3$ denote the three-variable majority function, and define $F_1(z_1, z_2, z_3) = \mathtt{MAJ}_3(z_1, z_2, z_3)$ and

$$F_{i+1}(z_1, ..., z_{3^{i+1}}) \stackrel{\text{def}}{=} \mathtt{MAJ}_3(F_i(z_1, ..., z_{3^i}), F_i(z_{3^i+1}, ..., z_{3^i+3^i}), F_i(z_{2\cdot 3^i+1}, ..., z_{3^{i+1}})). \tag{2}$$

for every $i \geq 1$. Finally, we let $F(z_1, ..., z_m) = F_\ell(z_1, ..., z_m)$.

The intuition is that each level in F increases the bias of the corresponding random variables (which are functions of $Y_1, ..., Y_m$) towards the majority value; that is, the probability that an internal vertex in the corresponding ternary tree evaluates to to b (where $\mathsf{Pr}[Y_1 = b] > 0.5$ for every $j \in [m]$), increases when going up the tree (i.e., away from the leaves). This effect is due to the bias-increasing property of $\mathtt{MAJ}_3$, which is stated next.

Fact 3 (three-way majority amplifies bias): *Let Z_1, Z_2, Z_3 be three independent identically distributed 0-1 random variables, and let $p \stackrel{\rm def}{=} \mathsf{Pr}[Z_1\!=\!1]$. Then:*

1. $p' \stackrel{\rm def}{=} \mathsf{Pr}[\mathtt{MAJ}_3(Z_1, Z_2, Z_3)\!=\!1] = 3 \cdot (1-p) \cdot p^2 + p^3$.
2. *Letting* $\delta \stackrel{\rm def}{=} p - 0.5$, *it holds that* $p' = 0.5 + (1.5 - 2\delta^2) \cdot \delta$.
3. $p' < 3p^2$.

We stress that the three parts hold for every $p \in [0,1]$, but we shall use Part 2 with $p > 0.5$ and use Part 3 with $p \ll 0.5$. Note that Part 2 implies that if $p \in (0.5,1)$, then $p' > 0.5 + \delta = p$.

Proof: The three parts follow by straightforward calculations. Specifically, Part 1 merely gives the expression for $\mathsf{Pr}[Z_1 + Z_2 + Z_3 \in \{2,3\}]$, and the other parts merely manipulate this expression (e.g., for Part 2 we use $p' = (3-2p)\cdot p^2 = (3-1-2\delta)\cdot(0.25+\delta+\delta^2)$, which implies $p' = 0.5 + 1.5\delta - 2\delta^3$, and for Part 3 we use $p' < 3\cdot(1-p)\cdot p^2 + 3\cdot p^3 = 3p^2$). ■

Analyzing $F_\ell(Y_1, ..., Y_m)$ using Fact 3. Fact 3 asserts that $\mathtt{MAJ}_3$ increases the bias of (independent and identically distributed) 0-1 random variables towards the majority value. The question is how fast does the bias (i.e., $p - 0.5$) tend to the extreme (i.e., 0.5) when the foregoing process is iterated. Applying Part 2 to the majority value, we see that as long as p is bounded away from 1 the value $p - 0.5$ increases by a constant factor. This will be useful towards increasing p from a value slightly above 0.5 (i.e., $p = 0.5 + 1/2n$) to a constant value in $(0.5, 1)$. At this point Part 3 becomes handy, provided that we apply it to the minority value. Doing so we drastically reduce the probability of the minority value (essentially squaring it). Details follow.

Part 2 of Fact 3 implies that if $p = 0.5 + \delta > 0.5$ and $\delta \le \delta_0 < 0.5$, then $p' \ge 0.5 + (1.5 - 2\delta_0^2)\cdot\delta$, 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 $p \ge 0.5 + 1/2n$, but similar considerations hold for $p \le 0.5 - 1/2n$.)[4] This means that we can increase the bias (i.e., $p - 0.5$) from its initial level of at least $1/2n$ to any constant level of $\delta_0 < 1/2$, by using $\ell_1 = \lceil c_1 \cdot \log_2(2\delta_0 n)\rceil$ iterations of $\mathtt{MAJ}_3$, where $c_1 = 1/\log_2(1.5 - 2\delta_0^2)$.[5]

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 (and Part 2), we may increase the bias from δ_0 to any larger constant that is smaller than 0.5. Specifically, we shall increase the bias to 0.4 (using $\ell_2 = \lceil \log_{1.18}(0.4/\delta_0)\rceil$).

At this point, we use Part 3 of Fact 3, 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 $3\cdot(1-p)^2$. Thus, using $\ell_3 = \lceil \log_2 n\rceil$ additional iterations, the probability of a wrong value reduces from $1 - (0.5 + 0.4) < 1/6$ to $3^{2^{\ell_3}-1}\cdot(1/6)^{2^{\ell_3}} < 2^{-2^{\ell_3}} \le 2^{-n}$.

Conclusion. 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 that, given $R(x)$, computes $\mathtt{MAJ}(x)$ with overwhelmingly high probability. That is:

[4] One way to see this is to define $p = \mathsf{Pr}[Z_1 = 0]$.

[5] Suppose that i iterations are necessary and sufficient for reducing the bias from $1/2n$ to δ_0. Then, $(1.5 - 2\delta_0^2)^i \cdot (1/2n) \ge \delta_0$ holds, which solves to $i \ge \log_{1.5-2\delta_0^2}(2\delta_0 n)$. Hence, we may use the minimal such $i \in \mathbb{N}$.

Theorem 4 (majority formulae via three-way majority): *For $x \in \{0,1\}^n$, let $\ell = 2.71 \log_2 n$ and $m = 3^\ell$, and consider the random process $R : \{0,1\}^n \to \{0,1\}^m$ such that each bit in $R(x)$ equals 1 with probability $\mathtt{wt}(x)/n$, independently of all other bits. Then, $\Pr[MAJ(x) = F_\ell(R(x))] > 1 - 2^{-n}$, where F_ℓ is as defined in Eq.* (2).[6]

Using Fact 2, Theorem 4 yields a formula (i.e., $F_\ell \circ R_\omega$) that computes $\mathtt{MAJ}(x)$ correctly on all inputs x, 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 we obtain a standard monotone monotone formula F' of depth $3\ell < 8.13 \log_2 n$. Recall 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 $\Pr[F'(Y_1, ..., Y_m) \neq b] < 2^{-n}$. Thus, for every $x \in \{0,1\}^n$ it holds that $\Pr_\omega[F'(R_\omega(x)) \neq \mathtt{MAJ}(x)] < 2^{-n}$ 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.

3 Comparison to Valiant's proof

Interestingly, Valiant [5] obtains a somewhat smaller formula 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., $\Pr_{z \in \{0,1\}^4}[V(z) = 1] = 9/16$), the random process used in [5] 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 $\Pr[Z_1 = 1] = p \overset{\text{def}}{=} (1 - \beta)/2$, then $\Pr[V(Z_1, Z_2, Z_3, Z_4) = 1] = p$.

It turns out that V amplifies the deviation from p slightly better than $\mathtt{MAJ}_3$ does (w.r.t 1/2).[7] More importantly, V can be implemented by a monotone formula of depth two (and fan-in two), whereas $\mathtt{MAJ}_3$ requires depth three. Thus, Valiant [5] performs $2.65 \log_2 n$ iterations (rather than $2.71 \log_2 n$ iterations), and obtains a formula of depth $5.3 \log_2 n$ (rather than $8.13 \log_2 n$).

Acknowledgments. Thanks to Alina Arbitman for her comments and suggestions regarding the original 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] G. Cohen, I. Damgard, Y. Ishai, J. Kolker, P. Miltersen, R. Raz, and R. Rothblum. Efficient Multiparty Protocols via Log-Depth Threshold Formulae. In the *33rd CRYPTO*, Part 2, Lecture Notes in Computer Science (Vol. 8043), Springer, pages 185–202, 2013.

[6] Actually, we have $\Pr[MAJ(x) = F_\ell(R(x))] > 1 - 2^{-n-\Omega(n)}$, because we can use $\ell_3 = (2.71 - c_1) \cdot \log_2 n - O(1) = \Omega(\log n)$, where $c_1 \approx 1/\log_2(1.5) \approx 1.70951129$. (Alternatively, note that the analysis of the last ℓ_3 iterations actually yields an error probability of $3^{2^{\ell_3} - 1} \cdot (0.1)^{2^{\ell_3}} < (0.3)^{2^{\ell_3}} < 2^{-1.7n}$. Furthermore, 0.1 can be replaced by any positive constant.)

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

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

[4] 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.

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