Testing monotonicity of Boolean functions

Bogdan Grechuk · 6 Sep 2026

https://functor.network/user/3333/entry/1966

One of the Best Paper Award winners at STOC 2026 is a paper by Mark Chen, Xi Chen, Hao Cui, William Pires, and Jonah Stockwell (Chen et al. 2026), which almost determines the query complexity of testing monotonicity of Boolean functions. The authors prove that even a fully adaptive randomized algorithm needs almost $\sqrt{n}$ queries. This improves the previous exponent $1/3$ to essentially the optimal exponent $1/2$.

Let us first formulate the problem. A Boolean function on n variables is a function

$$f : \{0,1\}^n \to \{0,1\}.$$

There are 2^n possible inputs, so the complete truth table of f contains 2^n values.

For

$$x = (x_1, \dots, x_n), \qquad y = (y_1, \dots, y_n) \in \{0,1\}^n,$$

write $x \le y$ if

$$x_i \le y_i \qquad \text{for every } i = 1, \dots, n.$$

We say that f is *monotone* if

$$x \le y \quad \Longrightarrow \quad f(x) \le f(y).$$

In other words, changing some coordinates of the input from 0 to 1 can never change the output from 1 to 0.

For example, for any fixed integer k, the function

$$f(x_1, \dots, x_n) = \begin{cases} 1, & x_1 + \cdots + x_n \ge k, \\ 0, & x_1 + \cdots + x_n < k \end{cases}$$

is monotone.

Suppose now that f is unknown, and that an algorithm may choose an input $x \in \{0,1\}^n$ and ask for the value $f(x)$. Such a request is called a *query*. The algorithm is not given the full truth table; it only sees the values at the points it queries.

If the goal were to determine exactly whether an arbitrary unknown function is monotone, one could not expect to do this after examining only a tiny part of its exponentially large truth table. Property testing asks a more flexible question: can we distinguish monotone functions from functions which are very far from every monotone function?

For two Boolean functions

$$f, g : \{0,1\}^n \to \{0,1\},$$

define their distance by

$$d(f,g) = \frac{1}{2^n} \left| \{x \in \{0,1\}^n : f(x) \neq g(x)\} \right|.$$

Thus $d(f,g)$ is the proportion of inputs on which the two functions have different values.

For a constant $\epsilon > 0$, we say that f is *ϵ-far from monotone* if

$$d(f,g) \geq \epsilon$$

for every monotone Boolean function g. Equivalently, one has to change the values of f at at least $\epsilon 2^n$ inputs in order to make it monotone.

A randomized algorithm is called an *ϵ-tester for monotonicity* if it satisfies the following two requirements:

1. if f is monotone, the algorithm outputs “monotone” with probability at least $2/3$;
2. if f is ϵ-far from monotone, the algorithm outputs “not monotone” with probability at least $2/3$.

The probability is over the random choices made by the algorithm. There is no requirement when f is non-monotone but is less than ϵ away from some monotone function.

The number of values of f requested by the algorithm is its *query complexity*. A tester is called *non-adaptive* if all points at which it queries f are selected before any answers are known. A tester is called *adaptive* if later queries are allowed to depend on the answers to earlier ones.

Adaptivity can potentially be very useful. After seeing a suspicious value of f, an adaptive algorithm may choose its next query nearby and try to locate a violation of monotonicity. A lower bound for adaptive testers is therefore substantially stronger than the same lower bound for non-adaptive testers.

The main theorem of Chen, Chen, Cui, Pires, and Stockwell (Chen et al. 2026) is the following.

Theorem. For every constant $c > 0$, there exist constants $\epsilon > 0$, $C > 0$, and n_0 such that, for every $n \geq n_0$, every randomized adaptive ϵ-tester for monotonicity of Boolean functions

$$f : \{0,1\}^n \to \{0,1\}$$

makes at least

$$Cn^{1/2-c}$$

queries.

Since c can be arbitrarily small, this says that the exponent in the lower bound can be made arbitrarily close to $1/2$. For example, by taking $c = 0.01$, one obtains a lower bound proportional to $n^{0.49}$, for some fixed value of ϵ. The constant ϵ is allowed to depend on c.

The history of the problem makes this theorem particularly striking. In 2015, Chen, De, Servedio, and Tan (Chen et al. 2015) had already proved an almost-$\sqrt{n}$ lower bound for *non-adaptive* testers: for every $c > 0$, some constant $\epsilon > 0$ forces at least a constant multiple of

$$n^{1/2-c}$$

queries. Thus the non-adaptive problem was already understood quite well. What remained unclear was whether adaptivity could provide a substantial advantage.

In 2017, Chen, Waingarten, and Xie (Chen et al. 2017) proved that an adaptive tester needs roughly $n^{1/3}$ queries, up to logarithmic factors. This was the best adaptive lower bound for several years.

On the other hand, Khot, Minzer, and Safra (Khot et al. 2018) constructed a *non-adaptive* tester using, for fixed ϵ, $\sqrt{n}$ queries multiplied only by a polylogarithmic factor in n. More precisely, their dependence on ϵ is proportional to $1/\epsilon^2$. In particular, for every fixed $\epsilon > 0$ and every $\delta > 0$,

their number of queries is at most

$$C_{\epsilon,\delta} n^{1/2+\delta}$$

for all sufficiently large n, where $C_{\epsilon,\delta}$ is a constant.

Before the new result, therefore, there was a genuine polynomial gap: the best adaptive lower bound had exponent $1/3$, whereas an upper bound with exponent arbitrarily close to $1/2$ was known. The new theorem closes this polynomial gap. The exponent of n in the query complexity is now known to be $1/2$.

There is an interesting contrast between n and 2^n here. An n-variable Boolean function has 2^n values, but testing monotonicity does not require anything remotely close to reading the full truth table. Roughly $\sqrt{n}$ queries suffice, apart from logarithmic factors, and the new theorem says that no randomized algorithm can reduce the exponent below $1/2$, even if it is allowed to choose every new query after seeing all previous answers.

References

Chen, Mark, Xi Chen, Hao Cui, William Pires, and Jonah Stockwell. 2026. “Boolean Function Monotonicity Testing Requires (Almost) $n^{1/2}$ Queries.” *Proceedings of the 58th Annual ACM Symposium on Theory of Computing*, 584–94. https://doi.org/10.1145/3798129.3800775.

Chen, Xi, Anindya De, Rocco A. Servedio, and Li-Yang Tan. 2015. “Boolean Function Monotonicity Testing Requires (Almost) $n^{1/2}$ Non-Adaptive Queries.” *Proceedings of the 47th Annual ACM Symposium on Theory of Computing*, 519–28. https://doi.org/10.1145/2746539.2746570.

Chen, Xi, Erik Waingarten, and Jinyu Xie. 2017. “Beyond Talagrand Functions: New Lower Bounds for Testing Monotonicity and Unateness.” *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, 523–36. https://doi.org/10.1145/3055399.3055461.

Khot, Subhash, Dor Minzer, and Muli Safra. 2018. “On Monotonicity Testing and Boolean Isoperimetric-Type Theorems.” *SIAM Journal on Computing* 47 (6): 2238–76. https://doi.org/10.1137/16M1065872.