# Self-Witnessing Polynomial-Time Complexity and Prime Factorization

Michael R. Fellows
Department of Computer Science
University of Victoria
Victoria, B.C. V8W 2Y2 Canada

Neal Koblitz
Department of Mathematics
University of Washington
Seattle, WA 98195 USA

**Abstract**

For a number of computational search problems, the existence of a polynomial-time algorithm for the problem implies that a polynomial-time algorithm for the problem is constructively known. Some instances of such *self-witnessing polynomial-time complexity* are presented. Our main result demonstrates this property for the problem of computing the prime factorization of a positive integer, based on a lemma which shows that a certificate for primality or compositeness can be constructed for a positive integer $p$ in deterministic polynomial time given a complete factorization of $p-1$.

## 1. Introduction

In a brief note Levin observed that if there exists a polynomial-time algorithm which finds nontrivial factors of a composite number, then in fact we constructively know such an algorithm (by a diagonalization) [Le]. Our main result extends this observation.

Levin's idea has subsequently found a number of uses. For example, combined with self-reduction, it is used in [FL] to constructivize most of the known concrete complexity consequences of the Graph Minor Theorem, and to show that the potential "nightmare" [Jo] of a nonconstructive proof of $P = NP$ cannot happen by means of the GMT and the usual kinds of completeness reductions. (The "nightmare" would be a proof of $P = NP$ that does not provide us a means of constructing polynomial-time algorithms for the many well-known problems in $NP$.) Levin's idea plays a role also in the theorem of Feigenbaum, Lipton and Mahaney that if any set in $NP$ has an almost everywhere invulnerable generator, then we constructively know one (for Satisfiability) [FLM].

We began the present study by considering the question of for what computational problems $X$ could we prove a theorem of the form: If there exists a polynomial-time algorithm for $X$ then we constructively know one. We propose to call such a problem *$P$-time self-witnessing*.

Although there is obviously an important distinction to be made between the situation where we constructively know a fast algorithm for a computational problem, and the situation where we know only that a fast algorithm exists, it is not a distinction that can be formalized in classical mathematical logic. Thus to define $P$-time self-witnessing it is necessary to make reference to some conceptual primitive of mathematical constructivism. (See Beeson [Be] for a comprehensive account of constructivist foundations of mathematics.)

In the following definition we refer, as a primitive, to the constructivist notion of the idealized collective mathematician $U$ (see [MRR]).

*Definition.* A computational problem $X$ given by an input–output specification is $P$-time self-witnessing if $U$ possesses a theorem of the form: $A$ is an algorithm such that if there exists any algorithm $B$ and polynomial $q$ such that $B$ solves $X$ in time bounded by $q$, then there exists a polynomial $q'$ such that $A$ solves $X$ in time bounded by $q'$.

The idea of employing classical arguments to prove the existence or correctness of constructively known algorithms is not new. An example of this, as well as an entertaining historical discussion of the relationship of the constructivist and classical mathematics of algorithms, can be found in [St].

Note that according to the above definition, if $X$ is $P$-time self-witnessing, then we constructively know a polynomial-time algorithm for $X$, but perhaps do not constructively know a polynomial bound on the running time of the algorithm (although such a bound exists). Also note that the "diagonalization" algorithms that we discuss are not practical.

In the next section we show that the problem of computing a prime factorization is $P$-time self-witnessing. In Section 3 we discuss a combinatorial instance of this phenomenon. Section 4 describes an important open problem concerning this notion.

## 2. Prime Factorization

This section first describes an unconditional deterministic polynomial time algorithm that, given an odd integer $p$ and a list of all prime factors of $p-1$, provides either a certificate of primality or a certificate of compositeness for $p$. We show how this can be used to demonstrate that the problem of computing a prime factorization is $P$-time self-witnessing.

Suppose that $p$ is a positive odd integer that is claimed to be prime, and we are given a complete list of the prime factors of $p-1$. Pratt [P] showed that primality can then be

certified in probabilistic polynomial time: namely, as soon as a random selection of integers $1 < j < p$ happens upon a generator of $(\mathbf{Z}/p\mathbf{Z})^*$, it will be easy to show, using the list of prime factors of $p-1$, that $j$ has exact order $p-1$ modulo $p$, and so $p$ must be prime. If the Riemann hypothesis is true for quadratic number fields, then even without knowing the factorization of $p-1$, Miller [M] found a deterministic polynomial time algorithm to certify primality. The lemma that follows is unconditional, i.e., it does not depend on any unproven conjecture.

*Lemma 1.* Given an odd integer $p > 1$ and a list $\{q_i\}_{i=1}^r$ of the prime factors of $p-1$, one can certify primality in deterministic polynomial time. More precisely, the deterministic polynomial time algorithm described below will, when given $p$ and the prime factors of $p-1$ as input, produce either a certificate of primality or a certificate of compositeness.

*Proof.* For each positive integer $j \leq \log^3 p$, find the exact order of $j$ modulo $p$ as follows. We may suppose that $j^{p-1} \equiv 1 \; mod \; p$, since otherwise $j$ provides a certificate that $p$ is composite. Set $h_j = p-1$, and then successively for the different $q_i$:

$$\text{if} \qquad \text{g.c.d.}(j^{h_j/q_i} - 1, p) \begin{cases} = 1, & \text{do nothing;} \\ = p, & \text{change } h_j \text{ to } h_j/q_i; \\ \neq 1, p, & \text{stop: you have a nontrivial factor of } p. \end{cases}$$

Continue until $h_j$ cannot be replaced by $h_j/q_i$ for any $i = 1, \cdots, r$. In polynomial time this gives the exact order $h_j$ of $j$ modulo $p$. Note that for any prime $l|p$ the same $h_j$ is also the exact order of $j$ modulo $l$, since otherwise we would have found a nontrivial factor of $p$ (i.e., the third case of the above g.c.d. would have occurred). In particular, this means that $h_j | l-1$ for any prime $l|p$. Finally, set $h = \text{l.c.m.}(h_j | 1 < j \leq \log^3 p)$.

If $h \geq \sqrt{p}$, then we know that $p$ is prime, because $h | l-1$ for any prime $l|p$.

If $h < \sqrt{p}$, then we claim that we know that $p$ is composite. Since $j^h \equiv 1 \; mod \; p$ for $1 \leq j \leq \log^3 p$, the same congruence holds for any $1 \leq j < p$ which is $\log^3 p$-smooth (this means that it has no prime factor greater than $\log^3 p$). If we show that the number of such $j$ is $\geq \sqrt{p}$, then it will follow that the polynomial $X^h - 1$ has $> h$ roots in $\mathbf{Z}/p\mathbf{Z}$, and so $p$ is composite. But it is known that

$$\Psi(x, \log^c x) = \#\{1 \leq j < x | j \text{ is } \log^c x - \text{smooth}\} \sim x^{1 - c^{-1} + o(1)},$$

where the $o(1)$ term is effective. We need only set $c = 3$ here to complete the proof of our claim. More precisely, the above smoothness estimate implies that

$$\text{for some effectively computable } p_0, \qquad \Psi(p, \log^3 p) \geq p^{1/2} \qquad \text{for} \qquad p \geq p_0.$$

Note that the smoothness result we needed is a very weak one, and it can easily be proved in the way described in §2 of [deB], where it is called an "almost trivial lower bound."

Thus, we obtain either a certificate of primality or a certificate of compositeness. □

*Remark.* Eric Bach and Jeff Shallit have offered the following observations and concrete estimates (e-mail communication):

(1) $p_0 = 10^{20}$ is sufficient;

(2) 3 in the exponent of $\log p$ can be replaced by $2 + \epsilon$ for any slowly growing $\epsilon$; for example, one can take $\epsilon = (\log\log\log p)/(\log\log p)$ and $p_0 = 10^{36}$.

*Corollary.* In deterministic polynomial time, given an oracle for factoring, one can compute a certificate of either primality or compositeness for arbitrary positive integer input. □

We now have the following extension of Levin's observation.

*Theorem 1.* The problem of computing a prime factorization is $P$-time self-witnessing.

*Proof.* Let $n$ denote the input positive integer to the following algorithm. We use a universal subroutine to generate all possible Turing machines. At the $i^{th}$ stage we generate $M_i$, and execute each of the separate computations $M_1, ..., M_i$ on the input $n$ for one further step. (This process is commonly termed diagonalization.) A computation $M_j$ is *finished* if it halts, offering purported prime factorizations $n = \prod p_j^{\alpha_j}$, $p_j - 1 = \prod q_k^{\beta_k}$ for each purported prime factor $p_j$, a purported prime factorization of each $q_k - 1$, and so on, ...

The output of a finished computation thus consists of $O(\log^2 n)$ such purported factorizations. A computation $M_j$ is *successful* if its output is certified correct (i.e., it is determined that each purported prime is in fact a prime) by the algorithm of Lemma 1. If there is any polynomial-time algorithm which correctly computes prime factorizations, then there is a computation $M_j$ which will finish in polynomial-time with all of the correct factorizations. We apply the algorithm of Lemma 1 to the output of any computation which finishes, and we halt when we find output certified by Lemma 1 to be a correct prime factorization. The total time spent by our algorithm will be roughly proportional to the square of the time spent by the successful computation. □

## 3. A Combinatorial Example

At the present time the main source of nontrivial combinatorial examples of self-witnessing computational complexity of which we are aware is the Graph Minor Theorem [RS2]. The problem $k$-Fold Planar Cover Search is that of finding either a $k$-fold planar cover of a graph $G$, or determining that no such planar cover exists. (For example, the reader can easily show that $K_5$ has a 2-fold planar cover.) Because for each $k$ the graphs having such a cover form a minor order lower ideal, we know that there exists a finite obstruction set (particular to $k$) $H_1, ..., H_m$ such that $G$ has a $k$-fold planar cover if and only if none of the

$H_i$ for $i = 1, ..., m$ is a minor of $G$. Thus, the decision problem of determining whether a $k$-fold planar cover of $G$ exists, can be solved by testing whether $H_i$ is a minor of $G$ for $i = 1, ..., m$. Each such test can be accomplished in time $O(n^3)$, where $n$ is the number of vertices of $G$, by the (constructively known) algorithm of [RS1]. This yields that for each fixed $k$ there exists an $O(n^3)$ algorithm for the decision problem. It is not known whether the obstruction sets can be effectively computed.

We also do not know whether there exists a polynomial-time algorithm for the search problem. The constructivization technique of [FL] based on self-reduction, which would reduce the search problem to the decision problem, is not known to apply. However, we have the following.

*Theorem 2.* $k$-Fold Planar Cover Search is $P$-time self-witnessing.

*Proof.* We diagonalize on all Turing machines as in the proof of Theorem 1; if output purporting to be a planar cover of $G$ is produced, then we check it. (Checking is easily accomplished in polynomial time.) If $G$ has a $k$-fold planar cover, this first process will halt in polynomial time if there is a $P$-time algorithm for the problem. We alternate computational steps between the first process and a second process which searches for evidence proving that $G$ has no $k$-fold planar cover.

The second process consists of generating and exhaustively analyzing all finite graphs to identify obstructions. A graph $H$ is an obstruction if and only if (1) $H$ does not have a $k$-fold planar cover, and (2) every proper minor of $H$ has a $k$-fold planar cover. The exhaustive analysis simply consists of generating combinatorially all $k$-fold covers, and attempting to embed them in the plane. Whenever an obstruction $H$ is found, we test whether $H$ is a minor of $G$ (in $O(n^3)$ time). Note that since the size of the obstruction set is a fixed constant, we make at most a constant number of these tests. The second process must halt in time $O(n^3)$ if $G$ does not have a $k$-fold planar cover. Thus in either case we will halt in time bounded by a fixed (but possibly unknown) polynomial. □

## 4. Remarks

With respect to the significance of nonconstructive methods in computational complexity, the following problem appears central.

*Open Problem.* Can we prove some familiar $NP$-hard problem to be $P$-time self-witnessing?

If we could, then the possibility of a nonconstructive proof of $P = NP$ would be moot: *any* proof of $P = NP$ would yield constructively known $P$-time algorithms for the hosts of natural problems in $NP$. Furthermore, we presently lack any means of arguing that an $NP$-hard problem (such as computing the chromatic number of a graph) cannot be, or is not likely to be, $P$-time self-witnessing. For example, such an outcome would not cause a

collapse of the polynomial hierarchy, simply because the latter is defined classically.

It is also to be remarked that this investigation engages in an unusual (although we believe well-motivated) blend of classical and constructive mathematics. This could be of interest from the point of view of mathematical logic.

## References


[Be] M. J. Beeson, *Foundations of Constructive Mathematics*, Springer, Berlin, 1985.

[deB] N. G. de Bruijn, On the number of positive integers $\leq x$ and free of prime factors $> y$, *Indag. Math.* **28** (1966), 239-247.

[FL] M. R. Fellows and M. A. Langston, "On Search, Decision and the Efficiency of Polynomial-Time Algorithms," *Proceedings of the ACM Symposium on Theory of Computing* (1989), 501-512.

[FLM] J. Feigenbaum, R. J. Lipton and S. R. Mahaney, "A Completeness Theorem for Almost Everywhere Invulnerable Generators," Technical Memorandum, AT&T Bell Laboratories, February 1989.

[Jo] D. S. Johnson, "The $NP$-Completeness Column: An Ongoing Guide," Column 19, *J. Algorithms* 8 (1987), 285-303.

[Le] L. A. Levin, "Universal Enumeration Problems," (Russian) *Problemy Peredachi Informatsii*, **IX** (1972), 115-116.

[M] G. L. Miller, Riemann's hypothesis and a test for primality, *J. Comput. System Sci.* **13** (1976), 300-317.

[MRR] R. Mines, F. Richman and W. Ruitenburg, *A Course in Constructive Algebra*, Springer-Verlag, New York, 1988.

[P] V. R. Pratt, Every prime has a succinct certificate, *SIAM J. Comput.* **4** (1975), 214-220.

[RS1] N. Robertson and P. D. Seymour, "Graph Minors XIII. Disjoint Paths," to appear.

[RS2] N. Robertson and P. D. Seymour, "Graph Minors XVI. Wagner's Conjecture," to appear.

[St] I. Stewart, *The Problems of Mathematics*, Oxford Univ. Press, 1987.