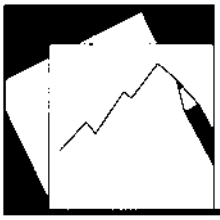
# Working Paper

INTERNATIONAL MONETARY FUND

WP/09/24



# IMF Working Paper

# Can Markets Compute Equilibria?

*Hunter Monroe*

INTERNATIONAL MONETARY FUND

© 2009 International Monetary Fund WP/09/24

**IMF Working Paper**

Western Hemisphere Department

# Can Markets Compute Equilibria?

**Prepared by Hunter Monroe**

Authorized for distribution by Paul Cashin

February 2009

## Abstract

**This Working Paper should not be reported as representing the views of the IMF.**
The views expressed in this Working Paper are those of the author(s) and do not necessarily represent those of the IMF or IMF policy. Working Papers describe research in progress by the author(s) and are published to elicit comments and to further debate.

Recent turmoil in financial and commodities markets has renewed questions regarding how well markets discover equilibrium prices, particularly when those markets are highly complex. A relatively new critique questions whether markets can realistically find equilibrium prices if computers cannot. For instance, in a simple exchange economy with Leontief preferences, the time required to compute equilibrium prices using the fastest known techniques is an exponential function of the number of goods. Furthermore, no efficient technique for this problem exists if a famous mathematical conjecture is correct. The conjecture states loosely that there are some problems for which finding an answer (i.e., an equilibrium price vector) is hard even though it is easy to check an answer (i.e., that a given price vector is an equilibrium). This paper provides a brief overview of computational complexity accessible to economists, and points out that the existence of computational problems with no best solution algorithm is relevant to this conjecture.

JEL Classification Numbers: C72, D01
Keywords: Nash equilibria, bounded rationality, computational complexity

Author's E-Mail Address: hmonroe@imf.org

Contents Page

## I. Introduction

Recent turmoil in financial and commodities markets has renewed questions regarding how well markets discover equilibrium prices.[1] A relatively new critique questions whether markets can realistically find equilibrium prices if computers cannot. The critique comes in two parts. First, in a simple exchange economy with Leontief preferences, the time required to compute equilibrium prices is an exponential function of the number of goods, using the fastest known techniques. This literature argues that "if your laptop cannot find it, neither can the market".[2] These are merely empirical statements; there may be an efficient technique that simply has not yet been discovered. Even so, the question remains how markets could outperform known programming techniques.

However, there is a more fundamental, theoretical objection: perhaps no efficient technique exists at all to compute equilibrium prices in certain settings. This conclusion would in fact follow from a famous mathematical conjecture, which states loosely that there are some problems for which finding an answer is hard even though it is easy to check an answer. For instance, it is easy to check that a given price vector is an equilibrium of the exchange economy above, but it is apparently hard to find an equilibrium price vector. Most mathematicians and computer scientists believe the conjecture is correct. This belief reflects in part the failure after several decades to discover an efficient technique to solve this class of problems. The Clay Mathematics Institute has set a US$1 million prize for a proof that the conjecture is correct or incorrect.

This paper provides a brief overview of computational complexity accessible to economists for an audience of economists, and is organized as follows. Section II reviews the evidence that computing equilibria in certain settings is inherently difficult to motivate an introduction to the key concepts. The presentation emphasizes accessibility to economists rather than technical precision.[3] With this background, Section III points out that the existence of computational problems with no best solution algorithm, a question which has not been a focus in the literature, is relevant to this conjecture.

## II. Is Computing Equilibria Difficult?

The section describes a simple exchange economy in which computing price equilibria appears to be inherently difficult. This motivates the introduction of key concepts used to specify exactly what it means for a computational problem to be difficult or easy, which provides a basis for what can be said despite the very limited understanding of which problems are hard.

---

[1]This is an extended version of Monroe (2008), in honor of the retirement of L. Richardson King from Davidson College. It would not have been prepared without encouragement from Bill Gasarch and Marius Zimand, and comments are also appreciated from an anonymous referee, Amir Ben-Amram, Paul Cashin, Neil Christensen, Bruno Codenotti, Padamja Khandelwal, Christos Papadimitriou, Cathy Pattillo, Nancy Wagner, and participants in seminars at Davidson College, the University of Maryland, and George Washington University and in the 2008 IEEE Conference on Computational Complexity. Remaining errors are my own.

[2]Kamal Jain, as quoted in Papadimitriou (2007).

[3]For a broader introduction to the theory of computational complexity, with precise definitions of the concepts used below, see Papadimitriou (1994) and Sipser (2001).

Consider the problem of finding an Arrow-Debreu equilibrium of an exchange economy with $n$ traders and $n$ goods where each trader $i$ has an initial endowment of one unit of good $i$. Traders have Leontief utility functions, where the desired proportion of trader $j$ for good $i$ is given by an $n \times n$ matrix $F$:[4]

$$u_j(x) = \min_{i:f_{ij}\neq 0} \left\{ \frac{x_i}{f_{ij}} \right\}. \tag{1}$$

A price vector $\pi$ is an equilibrium if the cost of one unit of utility is positive for each trader and the amount consumed of each good does not exceed the amount available:

$$\sum_k f_{kj}\pi_k > 0 \quad \text{for all } j \text{ and } \quad \frac{\sum_j f_{ij}\pi_j}{\sum_k f_{kj}\pi_k} \leq 1 \quad \text{for all } i. \tag{2}$$

No algorithm (i.e. computational technique) has been found for which the number of steps required for all $n \times n$ matrices $F$ to find an equilibrium $\pi$ is known to be at worst a polynomial function of $n$ for all matrices $F$. If one exists, we say the problem has worst-case polynomial time complexity, although the qualifier "worst-case" will be omitted below.

Whether a problem has an algorithm with polynomial time complexity is a widely-used indicator of its feasibility. This property is independent across a wide range of theoretical models of computers, such as Random Access Machines which can read and write at any point in memory like RAM in a desktop computers; Turing machines, which can read and write at a moving tape head on one or more one-dimensional tapes; and parallel computers, which have a fixed number of CPUs operating in parallel. The advantage of any one of these models over another is given by a polynomial function, so the property of polynomial time complexity is independent across these models.[5]

As a second example, consider the problem of finding a Nash equilibrium of a single stage, 2-player game, where each player has $n$ possible strategies. The payoffs of the two players under each possible combination of strategies are given by two $n \times n$ matrices. A special case is the Prisoner's Dilemma, where each player has $n = 2$ possible strategies (Table 1). A Nash equilibrium of such a game is one in which each player chooses a strategy which maximizes his payoff taking the other's strategy as given. Nash (1951) proved that at least one equilibrium exists with two or more players if one allows mixed strategies, in which a player chooses his strategy according to a probability distribution over the $n$ strategies. Finding the equilibrium in Table 1, which is (*Confess*, *Confess*), is straightforward.

| | *Deny* | *Confess* |
|---|---|---|
| *Deny* | -1,-1 | -10, 0 |
| *Confess* | 0,-10 | **- 5,-5** |

Table 1. Payoff Matrix for the Prisoner's Dilemma

The question arises: if there are many strategies for each player, is it computationally feasible to find a Nash equilibrium? Unfortunately, the well-known Lemke and Howson (1964) algorithm for

[4]The entries of $F$ might be all zeroes and ones.

[5]This is not known to be the case for quantum computers.

finding one Nash equilibrium requires a number of computational steps which is an exponential function of $n$ in the worst case, i.e. for certain pairs of $n \times n$ payoff matrices.[6] As the number of strategies becomes large, this algorithm is therefore infeasible. Again, no algorithm has been found for which the number of steps required for all pairs of $n \times n$ payoff matrices is known to be at worst a polynomial function of $n$.

Our understanding of computational complexity is highly limited: it is not known for a wide range of problems, including finding Arrow-Debreu or Nash equilibria, whether the problem can be solved in polynomial time. Nevertheless, the literature has succeeded in assessing many problems' difficulty relative to the difficulty of other problems. This has been accomplished by exhibiting methods, called reductions, for efficiently transforming an instance of one problem into an instance of another problem. For instance, the problem of finding a $n$-player Nash equilibrium for a fixed $n > 2$ is no harder than the two-player case;[7] the proof transforms an $n$-player game into a two-player game, and the solution to this two-player game is transformed back into a solution of the original $n$-player game.[8] Conversely, the two-player game is no harder than the $n$-player game—since we can add additional players with zero payoffs.

Thus, solving two-player and $n$-player games is equally hard. Both of these problems fall into a category of problems called $PPAD$ which rely upon fixed point theorems to prove the existence of the answer, in this case the existence of an equilibrium.[9] In fact, all problems in this category can be reduced to solving the two-player game, so this problem is hardest in category. A problem with this property of being hardest in category is called *complete*. Furthermore, finding an Arrow-Debreu equilibrium of the economy described above is at least as hard as finding a two-player Nash equilibrium (Chen and others 2007). Although it appears to be hard to find Nash equilibria, it is clearly easy to verify that a pair of mixed strategies is an equilibrium, using Nash's result that each pure strategy played with nonzero probability must be a best response to the other player's mixed strategy. Likewise, verifying that a price vector is an Arrow-Debreu equilibrium is easy using (2). In both cases, finding the answer seems to be harder than checking the answer.

This question is related to one of the greatest unsolved problems in mathematics, the $P$ versus $NP$ question. Problems in $P$ are defined as those which have a "yes" or "no" answer and have worst case polynomial time complexity. For example, checking that a pair of mixed strategies is a two-player Nash equilibrium is in $P$. Problems in $NP$ are those which have a "yes" or "no" answer, with a *certificate* for every "yes" answer that can be verified easily, i.e., the problem of verifying certificates is in $P$. For example, the problem of determining that for a given map that there is a round trip, or *Hamilton cycle*, passing through all cities once and only once is in $NP$ (Figure 1), because the list of cities in the order to be visited is an easily verifiable certificate (Figure 2). Determining that a Hamilton cycle exists is in fact at least as hard as any other $NP$ problem, that is, it is an $NP$-complete problem. Any other problem in $NP$ can be transformed into it in polynomial time. Another $NP$-complete problem is checking whether there are multiple Nash equilibria; the

[6] Savani and von Stengel (2005). See also McKelvey and McLennan (1996) and von Stengel (2002).

[7] See Chen and Deng (2006), improving a construction of Daskalakis and Papadimitriou (2005).

[8] Because as Nash pointed out there are 3-player games in which all equilibria are irrational, the literature examines multi-player equilibria in which each player's strategy is approximately optimal, to within $\epsilon$, given the strategies of the other players. Likewise, the literature examines Arrow-Debreu equilibria where markets clear within $\epsilon > 0$.

[9] For a precise definition of this category see Papadimitriou (1994).

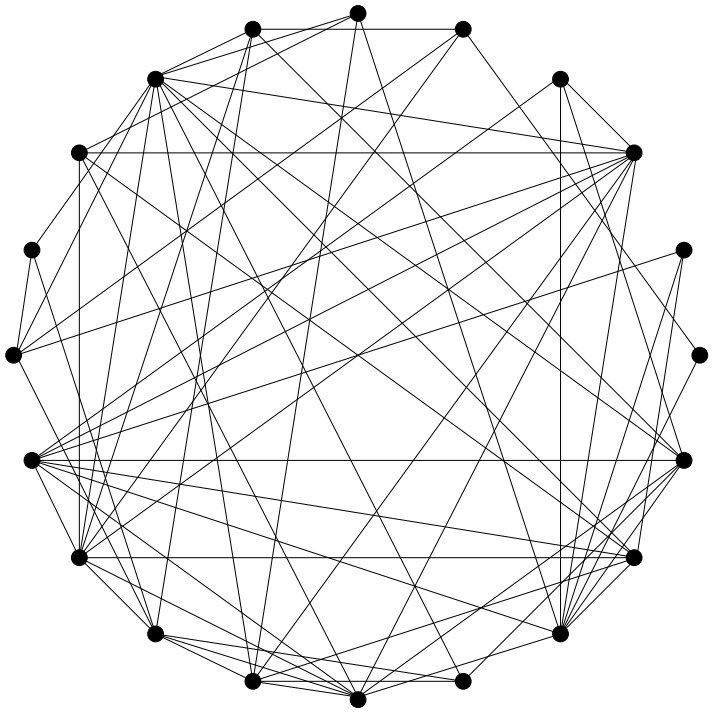


Figure 1. $NP$-complete: Is there a Hamilton Cycle?

certificate lists two equilibria. Surprisingly, these two apparently unrelated problems are equally hard: if either one can be solved in polynomial time, then so can the other.

Every problem in $P$ is also in $NP$, since the input itself can serve as the certificate—which is in $P$ by assumption. The $P$ versus $NP$ question asks whether the converse is true: are there problems in $NP$ which are not in $P$? For instance, is finding a Hamilton cycle (Figure 1) harder than verifying a Hamilton cycle (Figure 2)? This question's great significance stems from several sources. First, thousands of computational problems arising naturally in many industries and fields of research have been shown to be $NP$-complete (Garey and Johnson (1979)). This raises the stakes, because finding a fast algorithm for one would yield a fast algorithm for all the others, while showing that any one is hard implies that all are hard. Second, if $P = NP$, it would be easy to solve many important computational problems, but unfortunately also easy to break the encryption used for instance to send credit cards numbers over the Internet. Finally, one important $NP$-complete problem is that of determining whether a mathematical statement has a short proof, so the answer to the $P$ versus $NP$ question determines whether mathematical knowledge is harder to obtain than to verify. If $P = NP$, then finding mathematical theorems can be done easily by computers. It is generally believed that $P \neq NP$.

Consider what would be the implications $P \neq NP$ for the hardness of finding equilibria. If $P \neq PPAD$ then $P \neq NP$, but the converse is an open question. Therefore, it is possible that $P \neq NP$, but finding one Nash equilibrium, a $PPAD$-complete problem, is easy. However, $P \neq NP$ would nevertheless imply that the following problems are hard: is there a Nash equilibrium

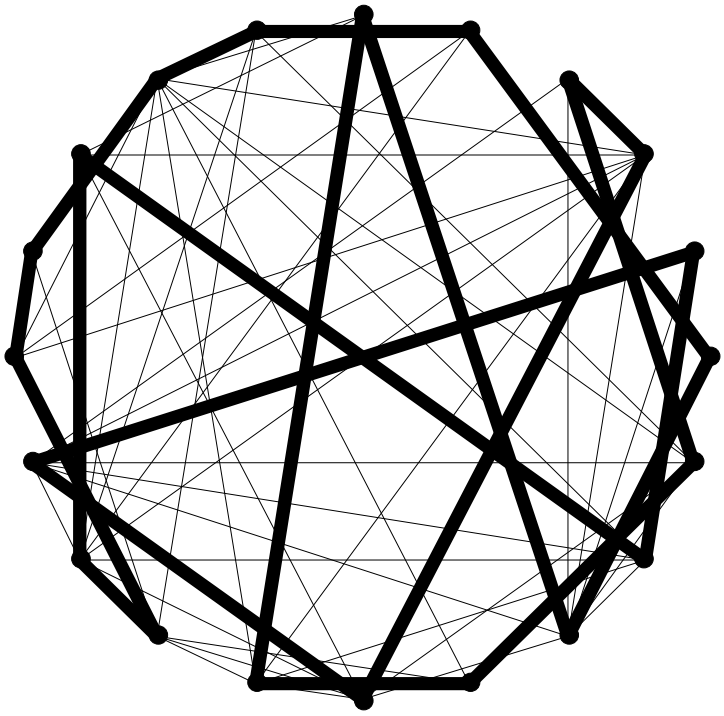


Figure 2. $P$: Is this a Hamilton Cycle?

achieving a given payoff; are there multiple equilibria; are all strategies in a specified subset given zero weight in some equilbrium, or all positive weight; and is there some Nash equilibrium in which at least a given number of strategies have positive weight, or no more than a given number.[10] Furthermore, the following problems related to Arrow-Debreu equilibria in the exchange economy described above would also be hard in the same sense: determining whether there is an equilibrium, and if there is one, whether there is more than one equilibrium and whether there is an equilibrium in which the prices of a given set of goods is positive (Codenotti and others 2006).

Papadimitriou (2007), who introduced the $PPAD$ class of problems, argues that efficient computability is an important modeling prerequisite, and that it is not meaningful to know that a market equilibrium exists if the participants cannot compute it: "If your laptop can't find it, neither can the market".[11] This point is valid even in the absence of a proof that finding Arrow-Debreu and Nash equilibria is hard, as it seems implausible that a market could do something which is beyond the reach of all known computer techniques. A proof that finding equilibria is inherently hard and not simply beyond known techniques would strengthen this argument.

The significance of such a proof would nevertheless require interpretation for several reasons. First, there are models in which finding an equilibrium is easy (Codenotti and others 2006). In practice, there are many markets which seem to clear, and the empirical challenge is determine which markets fail to clear and whether this failure can be attributed to computational difficulty. Second, concerns

[10] Gilboa and Zemel (1989). These problems are all $NP$-hard, or at least as hard as any problem in $NP$. In the two-player case, all of the above are $NP$-complete.

[11] Kamal Jain, as quoted in Papadimitriou (2007).

about the realism of assumptions made in economic modeling is nothing new: the classic proof that Arrow-Debreu equilibria exist assumes that there is a full set of state-contingent markets, while rational expectations models assume that agents can solve complex dynamic systems. Milton Friedman argued that an economic model should be judged not on the realism of its assumptions but its predictive success. Third, economists will still need to build models in any case, and it may not be tractable to build models and to design equilibrium concepts such that equilibria are efficiently computable.[12] An important objective of future research will be to determine whether there are large families of games and markets for which it is computational tractable to find equilibria. Finally, the important questions for policymakers—identifying markets which are "too" complex and designing mitigating policies—are not yet within reach.

With this background, the next section considers a special topic which has not been a focus in the complexity literature: whether there are natural problems with no best algorithm, and notes that this question is related to the $P$ versus $NP$ question.

## III. Are There Natural Problems with No Best Algorithm?

Some suspect that there is no best algorithm for integer multiplication or for matrix multiplication (MM).[13] This conjecture reflects in part the large number of clever cancellation tricks known for these problems—about 13 and 18 respectively.[14]

For instance, Strassen (1969) identified the following cancellation trick for fast matrix multiplication. Multiplying two $2 \times 2$ matrices using the "school" method requires eight multiplications of the form $a_{ij}b_{jk}$ where $i, j, k = 1, 2$:

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}. \tag{3}$$

However, Strassen's identity requires only seven multiplications:

$$\begin{array}{rcl} p_1 &=& (a_{12} + a_{22})(b_{21} + b_{12}) \\ p_2 &=& (a_{21} + a_{22})b_{11} \\ p_3 &=& a_{11}(b_{12} - b_{22}) \\ p_4 &=& a_{22}(b_{21} - b_{11}) \\ p_5 &=& (a_{11} - a_{12})b_{22} \\ p_6 &=& (a_{21} - a_{22})(b_{11} + b_{12}) \\ p_7 &=& (a_{12} - a_{22})(b_{21} + b_{22}) \end{array} \qquad \begin{array}{rcl} c_{11} &=& p_1 + p_4 - p_5 + p_7 \\ c_{12} &=& p_3 + p_5 \\ c_{21} &=& p_2 + p_5 \\ c_{22} &=& p_1 + p_3 - p_2 + p_6. \end{array} \tag{4}$$

To multiply $n \times n$ matrices for $n > 2$, assume $n = 2^k$ for some $k$ and subdivide each of the $n \times n$ matrices into four $\frac{n}{2} \times \frac{n}{2}$ matrices, and multiply these using the identity. This subdivision must be

[12] The problem of finding correlated equilibria is in $P$; see Gilboa and Zemel (1989).

[13] Schnorr and Stumpf (1975), Meyer and Fischer (1972), Blum (2007), and Christensen (1999).

[14] Bernstein (forthcoming) and Pan (1984).

repeated $k = log_2 n$ times. Therefore, the number of multiplications required is $7^{log_2 n} = n^{log_2 7}$, so this approach requires only $O(n^{log_2 7}) = O(n^{2.81})$ steps versus $O(n^{log_2 8}) = O(n^3)$ steps.[15]

The series of about 18 successive improvements to Strassen's identity described in Pan (1984) suggests that there is no best algorithm for MM. These improvements have reduced the exponent of MM from 3 in the school method, 2.81 using Strassen's identity, down to 2.376 using a different approach. In fact, there is no best MM identity of the form in equation (4), in which each term is linear in both the $a_{ij}$ and the $b_{kl}$ terms, to be called *bilinear*.[16]

Below, we identify several properties of problems that seem to imply that they have no best algorithm, and show that the existence of such problems is relevant to the $P$ versus $NP$ question. Say that a problem with no best algorithm has *speedup*. Section A defines superlinear speedup, and distinguishes it from the more widely known Blum speedup. Section B observes that if constructing identities such as Strassen's identity is hard, then $P \neq NP$. Section C notes that the property of some problems—that cancellation tricks provide a polynomial-time advantage—may imply that these problems have superlinear speedup. Section D notes that if there is more than polynomial speedup for the complement of any $NP$-complete problem, such as determining there is no Hamilton cycle, then $P \neq NP$. Section E points to parallels between the properties "has no best algorithm" and "has no algorithm at all".

## A. Superlinear vs. Blum Speedup

This section distinguishes the possible superlinear speedup of integer and matrix multiplication from the more familiar Blum speedup. Blum (1967) constructed an artificial problem with the following highly unusual property: for any algorithm for it requiring $T(n)$ steps, there is a better algorithm requiring at most $\log(T(n))$ steps for almost all inputs.[17] Integer and matrix multiplication clearly do not have this property: they have algorithms requiring $O(n^2)$ and $O(n^3)$ steps respectively, but there are no algorithms that require only $O(\log(n^2))$ and $O(\log(n^3))$ steps. These problems require a number of steps which is at least a linear function of $n$.

There are several distinctions between superlinear and Blum speedup. First, superlinear speedup relies on a much finer ranking of algorithms than Blum speedup. This ranking, formally a preorder,

---

[15] In "big-O" notation, $f(n) = O(g(n))$ if and only if there exist $N$ and $c > 0$ such that $f(n) < cg(n)$ for all $n > N$. This notation can be read informally as "grows more slowly than asymptotically". In this notation, the number of additions required drops out.

[16] Coppersmith and Winograd (1982), where identities may introduce an arbitrary unknown $\lambda$ which then cancels out. Also, if either of the two conjectures of Cohn and others (2005) are correct, then there is no best algorithm their type with exponent two.

[17] For a survey on Blum speedup, see van Emde Boas (1975). The preorder underlying Blum speedup is defined as follows. For a total recursive function $f$ and for time constructible total computable $R(\cdot)$ superlinear ($x = o(R(x))$), say that a program $\phi_i$ with abstract complexity measure $\Phi_i$ is *R-optimal* provided $\Phi_i(x) \leq R(\Phi_j(x))$ (for short, $\phi_i \leq_R \phi_j$) almost everywhere for each program $\phi_j$ for $f$. Equivalently, an $R$-optimal program $\phi_i$ is a least element in the preorder $\leq_R$.

used to define superlinear speedup is as follows.[18] For any two algorithms $A$ and $A'$, write $A \leq_O A'$ if there exists a constant $c$ such that for all inputs $x$,

$$time_A(x) \leq c \cdot time_{A'}(x) \tag{5}$$

where $time_A(x)$ is the number of steps required by $A$ on input $x$.[19] Say that a problem with no least element in this preorder has *superlinear speedup*.

Second, the function $\log$ for Blum speedup is defined in advance, while the constant $c$ in (5) is chosen after knowing $A$ and $A'$. Third, unlike problems with Blum speedup, a problem with superlinear speedup can have a well-defined lower bound, for instance, integer and matrix multiplication both require at least linear time. Finally, with Blum speedup there can be no automatic procedure for finding better algorithms (Blum (1971)), while there can be such a procedure for superlinear speedup. For instance, a better bilinear identity for $n \times n$ MM over a finite field, for instance, arithmetic modulo two, can always be found through exhaustive search of all possible identities for $n \times n$ MM.[20] Finding these identities appears to be merely difficult rather than impossible. The next section develops this idea.

### B. No Best Algorithm for Integer and Matrix Multiplication?

The nonexistence of a best identity for MM would seem inconsequential if there is a fast method to produce these, since there still would be an optimal algorithm which locates the best identity for each $n$ and then applies it. This section focuses on the possibility that these identities are hard to produce, and notes that this would imply $P \neq NP$. One reason to think so is that MM falls into a broader category of bilinear computations for which finding identities is $NP$-complete, i.e. hard to find if $P \neq NP$.[21]

An alternative method of expressing identities is to use the logical operations "AND", "OR", and "NOT" rather than "$\times$", "+", and "-" in seen in equation (4). Variables are Boolean, i.e., they can take only the values "TRUE" or "FALSE". Analogous to equation (4), there are input variables playing the role of $a_{ij}$ and $b_{kl}$, intermediate variables similar to the $p_i$, and output variables such as $c_{pq}$. There is no bilinearity assumption. These "identities" are known as *Boolean circuits*. For example, the Boolean circuit in Figure 3 has four inputs and outputs "TRUE" if at least two inputs are "TRUE".[22] Although their computing power might appear limited, they can in fact perform any computation in which the inputs and outputs are expressed in binary. For instance, the number 9 in

---

[18] Recall that a preorder "$\leq$" is a binary relation that is reflexive ($a \leq a$) and transitive ($a \leq b$ and $b \leq c$ implies $a \leq c$).

[19] An even finer-grained preorder $\leq_+$ uses an additive rather than multiplicative constant in (5), with a fixed alphabet for describing algorithms to avoid a trivial linear speedup (Seiferas (1990)).

[20] For arithmetic modulo two, $1 + 1 \equiv 0$, so there is an upper bound on possible coefficients in the MM identity, which makes the range of identities to be searched finite for each $n$.

[21] This is true over a finite field, such as arithmetic modulo two. A bilinear form is a set of bilinear functions in which all terms are linear in both $a_{ij}$ and $b_{kl}$. The number of multiplications needed to calculate bilinear forms over a finite field—for instance, seven in the case of $2 \times 2$ matrix multiplication—is $NP$-complete (Håstad (1990)).

[22] In Figure 3, the symbol $\wedge$ represents "AND", $\vee$ represents "OR", $x$ represents an input variable, and the output is shown at the top.

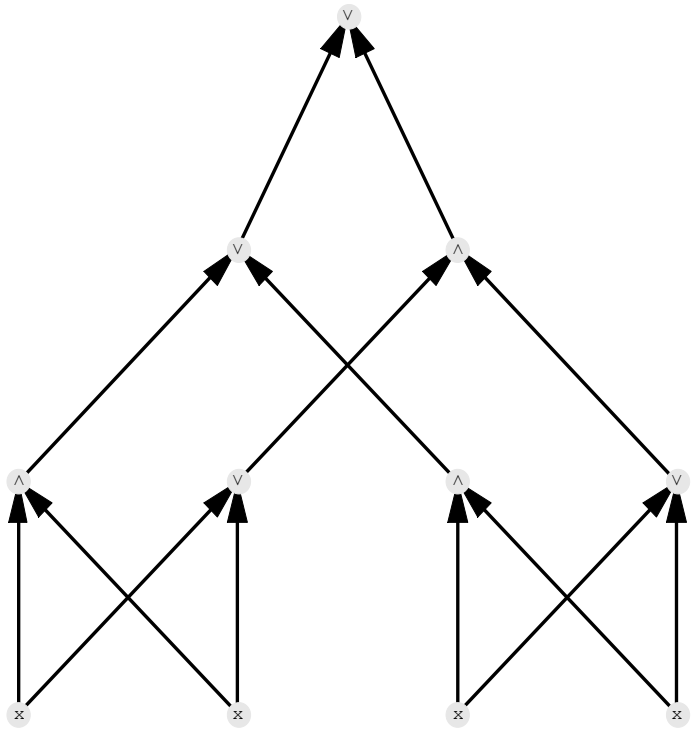



Figure 3. Boolean circuit: Are at least two inputs “TRUE”?

base 10 can be written as $1001$ in base two as input to a Boolean circuit for multiplication, where “1” stands for “TRUE” and “0” stands for “FALSE”. A given Boolean circuit has a fixed number of Boolean input variables, but there can be a different Boolean circuit specified for each number of inputs, for instance, a different multiplication circuit for each possible number of digits for the numbers being multiplied.

In fact, Boolean circuits are too powerful as a model of computing, as they can calculate the answer to problems which are not in fact computable. For instance, Turing (1936) showed that there can be no algorithm which takes a description of an algorithm as input and determines whether its halts at some point or runs on forever. This problem is called the halting problem. Some Boolean circuit family can solve this problem, known as the halting problem, because for any $n$ there is an $n$-input Boolean circuit producing every possible assignment of outputs to inputs. The problem is that picking out the correct family of Boolean circuits for the halting problem requires unlimited computational resources. Therefore, we consider a family of Boolean circuits only if some algorithm (say a Turing machine) can produce the circuit handling $n$ input variables within a polynomial number of steps in $n$. Such a family of circuits is called *$P$-uniform*.[23]

Suppose a problem has superlinear speedup among $P$-uniform families of circuits, measuring complexity by the number of gates rather than number of steps in (5). For such a problem, no family of circuits with the minimal number of AND, OR, and NOT operations for each input size $n$ is

[23]The literature typically uses $\log$-space uniformity, under which the arguments below go through but with weaker conclusions.

$P$-uniform, or the family would be optimal—under any reasonable preorder. Therefore, we can abstract from the preorder, and consider problems with no $P$-uniform family of minimal circuits. Such a problem in $P$ if one exists has minimal circuits that are harder to construct than to evaluate.[24]

We conjecture that neither integer nor matrix multiplication has a $P$-uniform family of minimal circuits. The conjecture implies that the circuit minimization problem is not in $P$, and therefore $P \neq NP$.[25]

### C. The Power of Cancellation

This section identifies a property of problems that appears to imply they do not have a $P$-uniform family of minimal circuits. The cancellation tricks used by fast integer and matrix multiplication algorithms give them a polynomial time advantage over the more familiar algorithms not employing cancellation taught in school. For instance, Strassen's identity provides an $n^3/n^{2.81} = n^{0.19}$ advantage over the school method for MM.

For this reason, circuits for the Boolean versions of these two problems which employ the NOT gates needed for cancellation (called nonmonotone circuits) have an advantage over circuits which do not employ NOT gates (called monotone circuits). In particular, nonmonotone circuits for Boolean convolution[26] can employ integer multiplication algorithms based on cancellation such as the discrete Fourier transform. Similarly, nonmonotone circuits for Boolean matrix multiplication can use MM algorithms based on cancellation, such as Strassen's identity.[27]

The two other problems known to have at least a polynomial gap between their monotone and nonmonotone circuit complexities display similar behavior to integer and matrix multiplication. The perfect matching problem has a large number of tricky algorithms, including one based on matrix inversion.[28] For Tardos' (1988) problem, the nonmonotone trick is the ellipsoid method, which also involves matrix operations. These examples suggest the conjecture that a problem has no $P$-uniform family of minimal circuits if and only if it has a polynomial or larger gap between its monotone and nonmonotone Boolean circuit complexity.[29]

Understanding which problems have a gap between their monotone and nonmonotone circuit size is important, given that monotone Boolean circuits certain $NP$-complete problems such as Hamilton

---

[24] At one point, the best-known circuits for integer division and iterated multiplication appeared to require more resources to construct than evaluate; see Hesse and others (2002).

[25] For a formal statement and proof, see Monroe (2008).

[26] Boolean convolution is a function used in multiplying two binary numbers. The convolution $(v_{2b-2}, \ldots, v_0)$ of two $b$ digit binary numbers $(x_{b-1}, \ldots, x_0)$ and $(y_{b-1}, \ldots, y_0)$ is given by $v_k = \sum_{i+j=k} x_i \cdot y_j$.

[27] Constructions of these nonmonotone circuits can be found in Wegener (1987). See also Boppana and Sipser (1990).

[28] See Mulmuley and others (1987). The perfect matching problem takes as input a list of men and women, and a list of pairs of men and women which are compatible. A perfect matching is a list of pairs all of which are compatible such that each man and each woman appears in exactly one pair.

[29] The fifth problem known to have a gap, Boolean sorting, has only a $\log n$ gap and appears to have easy to construct minimal circuits based on binary addition.

cycle require an exponential number of gates.[30] This result implies that these $NP$-complete problems have polynomial circuit size only if cancellation tricks are exponentially useful against them—as they are against Tardos' problem. Valiant (1992) observes that in nonmonotone circuits the computation and cancellation of unwanted terms plays a central role, and unfortunately it is not clear how to show that no such cancelled terms can exist for a certain problem.

### D. No Best Algorithm for $coNP$-Complete Problems?

Levin (1973) and Schnorr (1976) showed that there is an algorithm optimal up to a polynomial that correctly gives a "yes" answer for an $NP$-complete problem (for instance, "there is a Hamilton cycle").[31] Hartmanis asked whether there is also an optimal algorithm correctly giving a "yes" answer for the complement of an $NP$-complete problem (for instance, "there is no Hamilton cycle").[32] Call these *accepting* and *rejecting* algorithms respectively. A $coNP$-complete problem ("there is no Hamilton cycle") is the complement of an $NP$-complete problem. This section notes that if accepting any $coNP$-complete problem has more than polynomial speedup, then they all do, and $P \neq NP \neq coNP$.

This type of speedup relies on the following preorder over Turing machines which is coarser than $\leq_O$ defined in (5). For any two Turing machines $M$ and $M'$, write $M \leq_p M'$ if there exists a polynomial $p$ such that for all inputs $x$:

$$time_M(x) \leq p(|x|, time_{M'}(x)). \tag{6}$$

where $|x|$ is the length of input $x$. If there is a least element $M$ in this preorder, say that $M$ is *$p$-optimal* and otherwise that the problem has *superpolynomial speedup*. Krajíček and Pudlák (1989) show that $P = NP$ implies the existence of a $p$-optimal algorithm accepting propositional tautologies, which are $coNP$-complete.[33]

Any polynomial-time algorithm is $p$-optimal given the $|x|$ term, so no problem in $P$ has superpolynomial speedup. Therefore:

**Theorem 1.** *If some $coNP$-complete problem has superpolynomial speedup, then (1) they all do, and (2) $P \neq NP \neq coNP$.*

*Proof.* The assumption immediately implies that the $coNP$-complete problem is not in $P$, so part (2) follows. For part (1), any $p$-optimal algorithm for one must be reducible to a $p$-optimal algorithm for the other. *Q.E.D.*

---

[30] Razborov (1985) and Alon and Boppana (1987).

[31] See also Goldreich (2001), Ben-Amram (1997), and Messner (1999).

[32] As reported in Trakhtenbrot (1984).

[33] A propositional tautology is a Boolean formula which is correct for every possible assignment of values to its Boolean variables, for instance, $x \vee \neg x = 1$.

If $coNP$-complete problems have superpolynomial speedup, which is assumed in the remainder of this section, some interesting consequences would follow. First, because a $p$-optimal algorithm for both accepting and rejecting must be $p$-optimal for each, the problem of doing both for any $NP$-complete problem would have superpolynomial speedup. Therefore, the answer to Hartmanis' question—is there an optimal search algorithm that also rejects when there is no certificate—would be no. Second, superpolynomial speedup for accepting a $coNP$-complete problem would be one-sided given that accepting a $NP$-complete problem does not have superpolynomial speedup. However, for problems in $coNP \cap NP$, such as the property that a given integer is prime, the answer to Hartmanis' question is positive, since there are certificates for both accepting and rejecting.[34]

Finally, not only would we know that certificates for any $coNP$-complete problem are outside $P$, but that verifying certificates also has superpolynomial speedup. If a problem has a "swift checker", i.e., a verification algorithm which runs as fast as or faster than every algorithm for accepting, then Levin's certificate search algorithm, which runs every possible algorithm in parallel and then verifies the output, would be $p$-optimal.[35]

## E. No Best Algorithm Versus No Algorithm at All

This final section examines extreme versions of speedup to highlight some parallels between speedup and noncomputability. Speedup has been defined as the nonexistence of a fastest algorithm with respect to several successively coarser preorders: $\leq_+$, $\leq_O$, and $\leq_p$. The coarsest possible preorder places every algorithm for a problem into a single equivalence class, and all such algorithms are optimal in this preorder. A problem has speedup under the trivial preorder if and only if it has some algorithm which correctly answers "yes".[36] Thus, the property "has no fastest algorithm under $\leq_p$" can be seen as a finer-grained version of "has no algorithm at all". There is an apparent parallel between the known trivial speedup for the complement of the halting problem, for which there is no algorithm, and the conjectured superpolynomial speedup of the $coNP$-complete complement of bounded halting problem (Figure 4).[37]

---

[34]Ben-Amram (2007). A certificate than an integer is not prime is one of its factors. Pratt (1975) defined a certificate that an integer is a prime.

[35]See Ben-Amram (1997). For example, Hamilton cycle, which has a linear time certificate, has a swift checker w.r.t. $\leq_O$, and every $NP$-complete problem has a swift checker w.r.t. $\leq_p$. For the same reason, if MM has superlinear speedup on a RAM machine, on which Levin's algorithm is optimal up to a constant multiplicative factor plus the time required to verify, then MM verification would also have to have superlinear speedup, and no algorithm for MM verification could outperform every algorithm for MM computation and vice versa. Otherwise, MM would have a swift checker (see the previous footnote). Christensen (1999) identifies a problem with the property that the search, decision, and verification problems all have speedup, and any algorithm for one problem is outperformed by some algorithm for each of the other two problems. Therefore, if MM has superlinear speedup on a RAM, its deterministic and nondeterministic complexities coincide.

[36]Formally, a language has speedup in the trivial partial order if and only if it is not recursively enumerable (r.e.). See Rogers (1987). There is a nontrivial partial order which also shares this property, under which one algorithm is better than another if it accept more inputs correctly. That is, for a language $L$ and for Turing machines $M$ and $M'$ accepting languages $L_M$ and $L_{M'}$ respectively, write $M \leq_{re} M'$ if $L_M \subseteq L_{M'} \subseteq L$. Productive sets by definition have effective speedup under this partial order.

[37]The bounded halting problem is to determine, given a Turing machine M, a number of steps $n$, and a string $x$, whether there exists a string $y$ with length bounded by a polynomial in the length of $x$ such that $M$ halts within $n$ steps on the input $x, y$. Here, $y$ is the certificate. A natural proof of a circuit lower bounded for bounded halting might therefore not

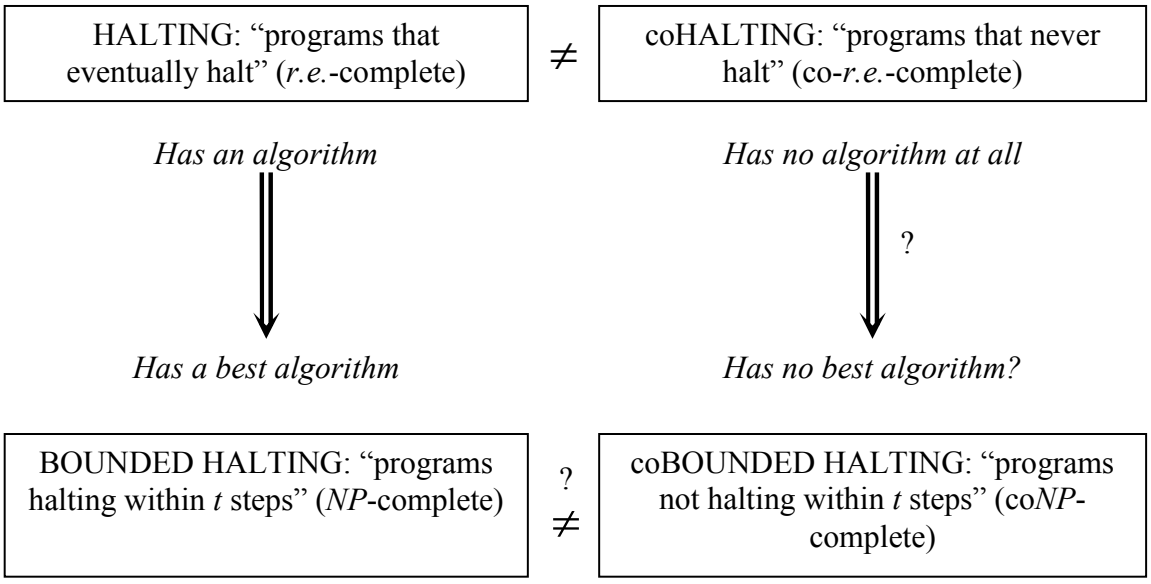



Figure 4. Is speedup inherited?

Under an extreme version of the property of having no $P$-uniform family of minimal circuits, a problem has minimal circuits that are not merely hard but are impossible to produce, i.e., the problem has no algorithm that both accepts and rejects. Integer multiplication is believed to have many slightly unruly properties, such as being one-way function,[38] which parallel the extremely unruly property that no algorithm can correctly identify provable statements in arithmetic if it includes multiplication.[39] In both cases above, describing a problem with no algorithm (e.g. halting, provable arithmetic statements) seems to require reference to a problem having algorithm and some form of speedup (e.g. bounded halting, integer multiplication).

The existence of natural problems with speedup would be consistent with the informal principle that most things are disorderly, i.e., most problems are nonrecursive, most real numbers are irrational, and most Boolean functions require Boolean circuits with $2^n/n$ gates. Calude and Zimand (1992) show that "most" partial recursive functions have Blum speedup.[40] However, their result does not concern speedup as defined above.

## IV. Conclusion

We have argued that speedup merits further investigation and has a bearing on important open problems. While there is a rich theory of orders over problems, based upon reductions, little is known regarding preorders over algorithms. It was a surprise when Turing (1936) showed that there

be able to use the fact that it is the bounded version of the nonrecursive halting problem, since that is proven using diagonalization. See Razborov and Rudich (1994).

[38]Informally, integer multiplication is one-way if it is harder to factor integers than to multiply them.

[39]Gödel's (1936) and Presburger (1929). Another possible parallel is that Gödel's speedup for verifying provable arithmetic statements might reflect a similar speedup for accepting; see the end of the previous section.

[40]The set of such functions is recursively Baire second category.

was no algorithm for a natural problem, identifying which Turing machines do not halt. Perhaps a similar surprise is in store if a natural problem is found to have speedup.

## REFERENCES


Alon, Noga, and Ravi Boppana, 1987, "The Monotone Circuit Complexity of Boolean Functions," *Combinatorica*, Vol. 7, pp. 1–22.

Ben-Amram, Amir, 1997, "The Existence of Optimal Programs," in *Computability and Complexity from a Programming Perspective*, Neil D. Jones, ed. (Cambridge, MA: MIT Press).

_____, 2007. Private communication.

Bernstein, Daniel, forthcoming, "Multidigit Multiplication for Mathematicians," *Advances in Applied Mathematics.*

Blum, Manuel, 1967, "A Machine-Independent Theory of the Complexity of Recursive Functions," *J. ACM*, Vol. 14, pp. 322–36.

_____, 1971, "On Effective Procedures for Speeding Up Algorithms," *J. ACM*, Vol. 18, pp. 290–305.

_____, 2007. Private communication.

Boppana, Ravi, and Michael Sipser, 1990, "The Complexity of Finite Functions," in *Handbook of Theoretical Computer Science: Volume A: Algorithms and Complexity*, J.van Leeuwen, ed. (Amsterdam: Elsevier), pp. 757–804.

Calude, Cristian, and Marius Zimand, 1992, "On Three Theorems in Abstract Complexity Theory: A Topological Glimpse," in "Abstracts of the 2nd International Colloquium on Words, Languages and Combinatorics," Kyoto, Japan, pp. 11–12.

Chen, Xi, and Xiaotie Deng, 2006, "Settling the Complexity of Two-player Nash Equilibrium," in "FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science," (Washington, DC: IEEE Computer Society), pp. 261–72.

_____, _____, and Shang hua Teng, 2007, "Settling the Complexity of Computing Two-player Nash Equilibria." arXiv.org:0704.1678.

Christensen, Niels H., 1999, "Levin, Blum and the Time Optimality of Programs," Master's thesis, (DIKU, University of Copehagen).

Codenotti, Bruno, Amin Saberi, Kasturi Varadarajan, and Yinyu Ye, 2006, "Leontief Economies Encode Nonzero Sum Two-player Games," in "SODA '06: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms," (New York, NY: ACM Press), pp. 659–67.

Cohn, Henry, Robert Kleinberg, Balazs Szegedy, and Christopher Umans, 2005, "Group-theoretic Algorithms for Matrix Multiplication," in "FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science," (Washington, DC: IEEE Computer Society), pp. 379–88.

Coppersmith, Don, and Shmuel Winograd, 1982, "On the Asymptotic Complexity of Matrix Multiplication," *SIAM J. Comput.*, Vol. 11, pp. 472–92.

Daskalakis, Konstantinos, and Christos H. Papadimitriou, 2005, "Three-Player Games Are Hard," *Electronic Colloquium on Computational Complexity (ECCC)*, No. 139.

Garey, M. R., and David S. Johnson, 1979, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (New York, NY: W. H. Freeman).

Gilboa, Itzhak, and Eitan Zemel, 1989, "Nash and Correlated Equilibria: Some Complexity Considerations," *Games and Economic Behavior*, Vol. 1, pp. 80–93.

Gödel, K., 1936, "Uber die Lange von Beweisen," *Ergebnisse eines Mathematischen Kolloquiums*, Vol. 24, pp. 745–50.

Goldreich, Oded, 2001, *Foundations of Cryptography*, Vol. Basic Tools (New York, NY: Cambridge University Press).

Håstad, Johan, 1990, "Tensor Rank is $NP$-complete," *J. Algorithms*, Vol. 11, pp. 644–54.

Hesse, William, Eric Allender, and David A. Mix Barrington, 2002, "Uniform Constant-Depth Threshold Circuits for Division and Iterated Multiplication," *J. Comput. Syst. Sci.*, Vol. 65, pp. 695–716.

Krajíček, Jan, and Pavel Pudlák, 1989, "Propositional Proof Systems, the Consistency of First Order Theories and the Complexity of Computations," *J. Symb. Log.*, Vol. 54, pp. 1063–79.

Lemke, C. E., and J. T. Howson Jr., 1964, "Equilibrium Points of Bimatrix Games," *Journal of the Society for Industrial and Applied Mathematics*, Vol. 12, No. 2, pp. 413–23.

Levin, Leonid A., 1973, "Universal Sequential Search Problems," *Problems of Information Transmission*, Vol. 9, pp. 265–66.

McKelvey, Richard D., and Andrew McLennan, 1996, "Computation of Equilibria in Finite Games," in *Handbook of Computational Economics*, Hans M. Amman, David A. Kendrick, and John Rust, eds. (Elsevier), pp. 87–142.

Messner, Jochen, 1999, "On Optimal Algorithms and Optimal Proof Systems," *Lecture Notes in Computer Science*, Vol. 1563, pp. 541–50.

Meyer, Albert R., and Patrick C. Fischer, 1972, "Computational Speed-Up by Effective Operators," *J. Symb. Log.*, Vol. 37, pp. 55–68.

Monroe, Hunter, 2008, "Are there Natural Problems with Speedup?," *Bulletin of the European Association for Theoretical Computer Science*, Vol. 94, pp. 212–20.

Mulmuley, Ketan, Umesh V. Vazirani, and Vijay V. Vazirani, 1987, "Matching is as Easy as Matrix Inversion," *Combinatorica*, Vol. 7, pp. 105–13.

Nash, J., 1951, "Non-Cooperative Games," *Annals Math.*, Vol. 54, pp. 286–95.

Pan, Victor, 1984, *How to Multiply Matrices Faster* (New York, NY: Springer-Verlag).

Papadimitriou, Christos H., 1994, *Computational Complexity* (Reading, MA: Addison-Wesley).

_____, 1994, "On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence," *J. Comput. Syst. Sci.*, Vol. 48, No. 3, pp. 498–532.

_____, 2007, *Equilibria and Complexity: What Now?* (Warwick University, UK). Presentation.

Pratt, Vaughan R., 1975, "Every Prime has a Succinct Certificate," *SIAM J. Comput.*, Vol. 4, pp. 214–20.

Presburger, M., 1929, "Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen in welchem die addition als einzige Operation hervorstritt," *Sprawozdanie z I Kongresu Matematikow Krajow Slowcanskich Warszawa*, pp. 92–101.

Razborov, Alexander A., 1985, "Lower Bounds on the Monotone Complexity of Some Boolean Functions," *Doklady Akademii Nauk SSSR*, Vol. 281, pp. 798–801. In Russian. English translation in *Soviet Mathematics Doklady*, 31:354–57, 1985.

_____, and Steven Rudich, 1994, "Natural Proofs," in "STOC '94: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing," (New York, NY: ACM Press) pp. 204–13.

Rogers, Jr. Hartley, 1987, *Theory of Recursive Functions and Effective Computability* (Cambridge, MA: MIT Press).

Savani, Rahul, and Bernhard von Stengel, 2004, "Exponentially Many Steps for Finding a Nash Equilibrium in a Bimatrix Game," in "FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science" (Washington, DC: IEEE Computer Society), pp. 258–67.

Schnorr, Claus-Peter, 1976, "Optimal Algorithms for Self-Reducible Problems," in "ICALP" pp. 322–37.

_____, and G. Stumpf, 1975, "A Characterization of Complexity Sequences," *Zeitschr. für Math. Logik und Grundlagen der Mathematik*, Vol. 21, pp. 47–56.

Seiferas, Joel I., 1990, "Machine-Independent Complexity Theory," in *Handbook of Theoretical Computer Science: Volume A: Algorithms and Complexity*, J.van Leeuwen, ed. (Amsterdam: Elsevier), pp. 163–86.

Sipser, Michael, 2001, *Introduction to the Theory of Computation* (Boston, MA: PWS Publishing Company).

Strassen, Volker, 1969, "Gaussian Elimination Is Not Optimal," *Numerische Mathematik*, Vol. 14, No. 3, pp. 354–56.

Tardos, Éva, 1988, "The Gap Between Monotone and Non-Monotone Circuit Complexity is Exponential," *Combinatorica*, Vol. 8, pp. 141–42.

Trakhtenbrot, Boris A., 1984, "A Survey of Russian Approaches to Perebor (Brute-force Search) Algorithms," *Annals of the History of Computing*, Vol. 6, pp. 384–400.

Turing, Alan M., 1936, "On Computable Numbers, with an Application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society*, Vol. 2, pp. 230–65.

Valiant, Leslie G., 1992, "Why is Boolean Complexity Theory Difficult?," in "Proceedings of the London Mathematical Society Symposium on Boolean Function Complexity" (New York, NY: Cambridge University Press), pp. 84–94.

van Emde Boas, Peter, 1975, "Ten Years of Speedup," in *MFCS*, Jirí Becvár, ed., Vol. 32 of *Lecture Notes in Computer Science* (New York, NY: Springer Verlag), pp. 13–29.

Wegener, Ingo, 1987, *The Complexity of Boolean Functions* (New York, NY: John Wiley & Sons, Inc.).