INTRACTABILITY PROOFS AND THE COMPUTATIONAL COMPLEXITY OF BINARY QUADRATICS

by

L. Adleman and K. Manders

Memorandum No. UCB/ERL M78/30

1 June 1978

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

INTRACTABILITY PROOFS AND THE COMPUTATIONAL COMPLEXITY OF BINARY QUADRATICS

Leonard Adleman[*]
Massachusetts Institute of Technology

and

Kenneth Manders[**]
University of California, Berkeley

Abstract

The study "Intractability Proofs and the Computational Complexity of Binary Quadratics" is concerned with determination of computational tractability or intractability of problems in NP, i.e. solvable in nondeterministic polynomial time. This comprises a large variety of "almost" tractable decision problems, often important in applied areas such as Operations Research.

It is proposed to broaden the customary definition of computational tractability, solvability in deterministic polynomial time, to random-co-random recognisability in polynomial time. New methods are given for showing problems intractable, regardless of the definition used. These methods are potentially more easily and more broadly applicable than the sole previously available method, that of NP-completeness. They involve showing that any problem in NP can be reduced to the problem A being shown intractable by a nondeterministic polynomial time computation (showing, e.g. that $A \in NP \cap NP^c \Leftrightarrow NP = NP^c$); previously, only deterministic reductions were considered.

[*]Supported in part by Office of Naval Research Contract N00014-67-0204-0063.

[**]Supported by National Science Foundation Grant DCR72-03725-A02.

The methods are applied to prove intractable the problem of determining solvability in integers for equations of any of the forms:

(1) $$ax^2 + by = c$$

(2) $$x^2 - a^2y^2 = c$$

(3) $$x^2 + axy = c$$

(4) $$(ax + b)y = c$$

Only for (1) can this be shown by NP-completeness. The analysis of the other problems depends on detailed knowledge of the distribution of primes in arithmetic progressions.

Introduction

There is considerable practical interest to methods for determining whether a given problem is computationally tractable. Research strategy in developing a solution method for a problem will depend critically on one's assessment of tractability. The way to deal with a tractable problem is to find a good algorithm for solving the problem; to deal with a computationally intractable problem, one must return to the intended application and determine whether an approximate solution is adequate, or whether a heuristic method is likely to yield an acceptable solution with acceptable use of computational resources, for at least most instances of the problem which arise in the application. These are very different activities, and efficient allocation of research effort requires determination at an early stage of the strategy to be followed.

The first chapter of this study is devoted to the development of methods for intractability proofs which are applicable to problems on the verge of tractability -- problems about which one might genuninely be in doubt whether or not they are tractable. We motivate the theory of computational complexity in terms of which we define tractability. We then discuss methods for providing evidence for intractability; first the now "classical" method of NP-completeness, and then new methods based on generalized notions of polynomial time interreducibility of computational problems. The development is self-contained, but reference is made to proofs elsewhere that specific problems are indeed intractable in the senses defined.

The second chapter is devoted to application of the methods we have developed to the complexity classification of number theoretic problems.

Complexity classification of number theoretic problems is of interest both because of the insight it gives into number theoretic issues and because it provides complexity theory with insight from a deeply developed area of classical mathematics. This is especially important because of the present state of abstract complexity theory -- the apparent inadequacy of present techniques for proving lower bounds on complexity, as exemplified by the problem of showing that $NP \neq P$. For in the present situation, all we can show about intractability of a problem in NP is that it is about as hard as a large class of other problems in many different areas, none of which are known or believed to have feasible solutions. By including number theoretic problems in the classification, we provide significant evidence that the problems in the other areas which are believed intractable really are intractable; for we thereby show that even the deep results of centuries of number theoretic inquiry have not yielded feasible algorithms for computationally "equivalent" or related number theoretic problems.

The problems we classify are solvability problems for binary quadratic equations. Integer solvability problems for equations encompass equivalents of all decidable (and r.e.) problems, and hence are a universal normal form representation of such problems. The binary quadratics are of very special interest for our enterprise because their solvability problems cover the range of complexity levels to which our methods are relevant, and their solution is intimately related to the most fundamental and deeply studied number theoretic computational problems, such as prime factorization of numbers and determination of properties of quadratic number fields.

We will use the following notation: ω will denote the set of natural numbers $0,1,2,\ldots$; $\mathbb{Z}$ the integers $\ldots -2,-1,0,1,2,\ldots$. Sequences of natural numbers will be written as $\langle\cdot,\ldots,\cdot\rangle$, and this same notation will be used for the number encoding the sequence which is obtained by repeated use of a standard pairing function $\omega\times\omega\to\omega$. Similarly, all combinatorial objects which may serve as inputs for computation will be regarded as coded by natural numbers; thus every decision problem will be viewed as the problem of deciding membership in a subset of ω. The complement of a set $A\subseteq\omega$ in ω will be denoted by A^c; but for a class K of subsets of ω, such as NP, K^c will denote the class of complements of the sets in K.

1. Methods for Intractability Proofs

1.1 Conventions of Complexity Theory and Their Motivation

The initial problem of a theory of inherent computational complexity is to find a framework for classification of problems according to computational complexity which is at the same time independent enough of choice of computational model to be of general theoretical significance and sensitive enough to make the discriminations between complexity levels which are required in a given context. This requires a compromise. The most machine-independent theory of complexity in the literature is perhaps that of Blum [7], [13]. This theory gives general insights of considerable significance, but for practical evaluation of algorithms in common use considerably more attention must be paid to the influence of implementation, even down to the level of the study of time lags in individual hardware realizations [16]; this area is generally called concrete complexity.

The choice of conventions used in this study is intermediate between these extremes; it is based on the empirical observation that the running time requirements for "reasonable" models of computation in the literature are polynomially related; i.e. for any two such models M_1, M_2 there are polynomials $p(\cdot)$, $q(\cdot)$ such that

(i) M_1 requires less than t steps $\Rightarrow M_2$ requires less than $p(t)$ steps,

(ii) M_2 requires less than t steps $\Rightarrow M_1$ requires less than $q(t)$ steps,

for any given input. This entails that as long as we classify all problems with polynomially related complexity bounds together, the resulting complexity classification will be independent of the choice of any "reasonable" model. The resulting loss of resolution, while too great for certain practical applications, is much less than that in the Blum theory and yields an interesting complexity theory for many simple and important concrete problems.

The preceding discussion presupposes that we have resolved a difficulty: Any reasonable machine model can be programmed to deal with any (sufficiently small, in practice) finite set of inputs, for example by storing a table of relevant outputs in memory. This efficiency has little to do with the inherent complexity of "figuring out the answer" to a problem, and the conventions of our complexity theory must be chosen so that this phenomenon does not influence the complexity classification. The by now classical solution to this difficulty is to define a _problem_ as a function on an infinite set of inputs, each of which is a finite combinatorial object (such as numbers in binary representation); then we classify all problems together whose resource requirements differ on only a finite set of inputs.

This can be achieved by defining a running time bound as a function of input _size_, rather than of the input itself, and using only the asymptotic behavior of the bounds in complexity classification. The choice of input size as argument for complexity bounds has an important further methodological advantage. The resulting complexity theory deals uniformly with many types of inputs (numbers, graphs, polynomials); complexity expressed in terms of inputs or characteristic parameters of inputs (numbers, number of nodes, degree and coefficient size, ...) may be natural for a given type of input, but does not allow direct comparison of complexity with problems of other types. This universality will play an important role in our methods. We will use the symbol $|x|$ in this section to denote the length (size) of the input x. We thus arrive at the following classical definitions, which are given in terms of a Turing machine model, for the sake of rigor; by the above remarks the resulting complexity classification will not depend on this choice of model. For definitions and motivation of the notion of Turing machine, see Turing [31] or Minsky [26].

Definition. A Turing machine M runs in time $t: \omega \to \omega$ if and only if for any input x (of length $|x|$) M halts within $t(|x|)$ steps. M runs in polynomial time if and only if for some polynomial $p(\cdot)$, M runs in time $p(.)$. The class of sets decidable (by a Turing machine which runs) in polynomial time will be denoted by P.

1.2 Intractability

A theory of intractability requires a definition of tractability. We will consider two proposed definitions in this thesis, without deciding among them. Before doing so, we define the class of problems which will serve as the scope of our methods and focus them on problems "on the verge of" tractability.

Definition. A nondeterministic Turing machine (NDTM) is a Turing machine which may have more than one quintuple with any given state-input symbol pair. Thus on a given input to the machine, a number of computation paths are possible (one corresponding to each sequence of choices between quintuples with relevant state-input symbol pair), and these form a tree. The machine accepts an input x if and only if some computation path on that input terminates in a designated state, called the accepting state. A NDTM M accepts a set S if and only if S is exactly the set of inputs accepted by M. M runs in time $t: \omega \to \omega$ if and only if for every input x, all computation paths halt within $t(|x|)$ steps; M runs in polynomial time if M runs in time $p(\cdot)$ for some polynomial $p(\cdot)$; the class of sets accepted by an NDTM in polynomial time will be denoted by NP.

Our methods and results will be primarily concerned with problems in NP, and, derivatively, with problems in NP^c (i.e. with sets whose complement is in NP). This class contains a wide variety of concrete problems in many areas of application. It is a natural class in the sense that it is usually easy to recognize whether a problem is in NP. Problems in NP are indeed on the verge of intractability in that it is a priori conceivable that nontrivial methods might yield a feasible solution, even though a naive approach (such as deterministic simulation of all branches in the computation tree) leads to exponential explosion of resource requirements.

We next describe a more constrained version of nondeterministic computation which will serve in one of our definitions of tractability. If T is a finitely branching tree, an assignment of weights to T is a map $T: (0,1]$ from the nodes of T to real numbers, such that the root of T is assigned 1; the uniform assignment of weights is the assignment of weights such that if a node has weight w and has n distinct immediate successors, each successor node has weight w/n. The weight of a branch of T is the weight assigned to the lowest node in the branch.

Definition. An NDTM M accepts a set S randomly if and only if

(i) M accepts S, and

(ii) the accepting paths for any accepted input x (in the sense of (i)) have weight together of at least $\frac{1}{2}$ in the uniform assignment of weights to the computation tree of M on input x.

An NDTM runs in random time $t: \omega \to \omega$ if and only if it accepts randomly and runs in time t. We denote by R the class of sets accepted in random polynomial time.

Remarks. 1. Another formulation of accepting randomly is that any input for which there is some accepting computation path (in the NDTM sense) has in fact "at least half" of the paths in its computation tree accepting. Here "at least half" is expressed more precisely by the weight of accepting paths in uniform weighting as described above. The expression "at least half the paths" is literally accurate if all paths are of the same length and every path involves the same amount of branching at each level. In proving that algorithms are random we will often make use of this formulation.

2. The class random polynomial time (R) is not affected if we change the weight requirement from '$> \frac{1}{2}$' to '$> \varepsilon$', for $\varepsilon > 0$, or even to '$> \frac{1}{|x|^k}$' for finite k, and $|x|$ the length of the input x. This is shown by an easy simulation argument, comparable to the argument for the following proposition:

Proposition. Assume $S \in R \cap R^c$, R^c the class of sets whose complements are accepted in random polynomial time. Then for any $\varepsilon > 0$ there is a polynomial running time $p(\cdot)$ and a deterministic algorithm M with a random element such that

(i) M runs in time p;

(ii) for all but $1-\frac{1}{\varepsilon}$ of the possible histories of the random element of length $p(|x|)$, x any input,

$$M \text{ outputs '0' on input } x \text{ if } x \in S ,$$
$$M \text{ outputs '1' on input } x \text{ if } x \notin S ;$$

(iii) for at most $\frac{1}{\varepsilon}$ of the possible histories of the random element of length $p(|x|)$, x any input, M outputs '?'.

Proof. Let M_1 be a random polynomial time acceptor of S and M_2 be a random polynomial time acceptor of S^c. Consider the algorithm

> "On input x, simulate both M_1 and M_2, using the random element to choose a computational path of each. If both halt in a nonaccepting state, we start over, and repeat n times, where $\frac{1}{2^n} < \varepsilon$. If we still have not reached an accepting state, output '?'; otherwise output 0 if M_1 halted or 1 if M_2 halted."

This assignment is consistent because the sets accepted by M_1 and M_2 are disjoint. It is clear that for n repetitions of the simulation, i.e. each history tries out n paths in the trees of $M_1(x)$ and $M_2(x)$, at most $\frac{1}{2^n}$ of the histories fail to locate an accepting path of either M_1 or M_2. The entire simulation requires essentially $p = n\cdot(p_1+p_2)$ steps, where p_1, p_2 are the polynomial running times of M_1 and M_2.

We are now in a position to define tractability. A classical proposal is that

(Ti) a problem is tractable if and only if it is solvable in deterministic polynomial time, i.e. $S \in P$.

We wish to propose as an alternative explication:

(Tii) S is tractable if and only if $S \in R \cap R^c$.

The argument for this proposal rests on the Proposition and its proof. For assuming, for example on the basis of (Ti), that we can afford to spend polynomial time on problems of interest, we argue that we can also afford to risk a very small probability that we fail to come up with a solution in the given time. After all, we can keep trying if we really wish to find a solution. Note that the probability of failure does not depend on the input; that is, failure does <u>not</u> result because the algorithm is unable to deal with a particular, difficult, input in the given time.

There is a classical objection to (Ti), which is equally valid against (Tii): It is not obvious that one can afford polynomial time if the polynomial has large coefficients or large exponents. This is a genuine difficulty, which however has not become acute in practice: most concrete problems which have been found solvable in deterministic polynomial time have been found to have reasonably well-behaved running time bounds. A further advantage of (Ti) and (Tii) over modified versions restricting the size of running time polynomials is the theoretical convenience of implementation independence obtained by the polynomial closure of P and $R \cap R^c$, as explained at the beginning of this section. We will not consider further arguments for, against, or between (Ti) and (Tii) here, but rather develop the theory of intractability on the basis of both proposals.

In order to expose the theoretical problems underlying determination of (in-)tractability we consider the relationship between the complexity classes introduced so far:

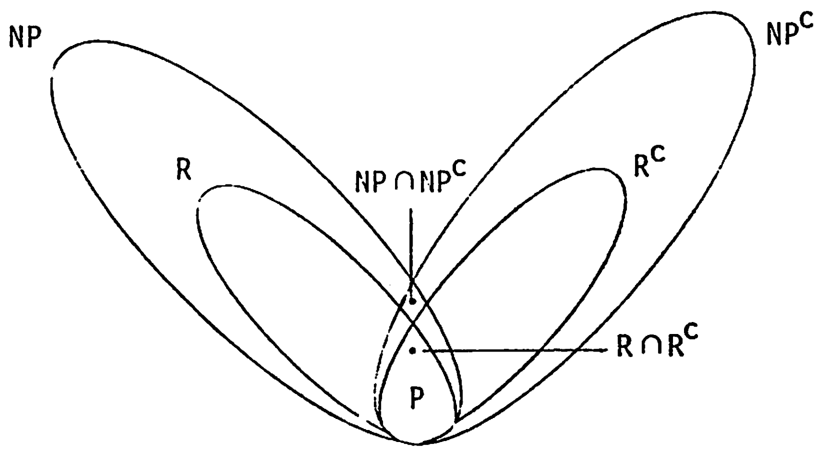

Diagram I.1

All the inclucions indicated in the diagram are easily proved from the definitions given above. On the other hand, it is a classical, apparently difficult

Open Problem. Is P distinct from NP (or equivalently, from NP^c)? ($NP \overset{?}{=} P$)-problem.

It will be important to the argument below that it is generally conjectured in complexity theory that all the inclusions indicated as strict in the diagram are indeed strict; e.g. $NP \supsetneq P$, and all intersections of classes depicted as nonempty are indeed nonempty.

We observe that on the one hand, proving that a tractable problem is tractable is in principle theoretically unproblematic, though in some cases perhaps formidable: one must produce (Ti) a deterministic algorithm, or (Tii) a pair of random algorithms; show the algorithm(s) correct, and prove polynomial running time bound(s). On the other hand, to show that there is a problem $S \in NP$ which is not tractable, i.e. either $S \notin P$ or $S \notin R \cap R^c$, entails solving the open ($NP \overset{?}{=} P$) problem. This problem is worth solving, but it is hardly fair to suggest that one's preliminary assessment of research strategy concerning an applied computational problem should include determination of a solution to this redoubtable theoretical issue. The theoretical challenge of intractability proofs then is to provide a methodology for demonstrating intractability in some weaker sense, not requiring solution of the NP-P problem.

1.3 Evidence for Intractability

The first method for providing evidence of intractability for problems in NP, and until the methods of this study the only available method, was provided by work of Cook [10] and Karp [15] and, independently, Levin [20]. Their now classical method, that of NP-completeness, will serve as a model for the other methods to be developed. We therefore present and analyze the method in some detail.

Let S and T be two sets. S is said to be polynomial time reducible to T if there is a (deterministic) polynomial time algorithm computing a function f_S such that for any x: $x \in S \Leftrightarrow f_S(x) \in T$. This condition clearly entails reducibility of the problem of accepting or determining membership in S to the analogous problem for T: to find out about the relationship between x and S, compute $f_S(x)$ and ask about the relationship between $f_S(x)$ and T.

Moreover, the restriction of the reduction algorithm to polynomial times guarantees that if solution of the problem S to be reduced requires much more than polynomial time, solution of the problem T must be essentially as hard as solution of the problem S; if all problems in a class are polynomial time reducible to a given problem, that problem must be essentially as hard as any problem in the class. In this spirit, we say that (the problem of accepting) the set S is _NP-hard_ if every set in NP is polynomial time reducible to S. If S is NP-hard and $S \in NP$, then S is _NP-complete_.

NP-complete sets do in fact exist. Let a _literal_ be a propositional (i.e. Boolean) variable or the negation of a propositional variable. Consider the propositional formulas ϕ of the form $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_k$; $C_i = \ell_{1i} \vee \ell_{2i} \vee \ell_{3i}$; ℓ_{ji} a literal for $1 \leq i \leq k$, $1 \leq j \leq 3$. The C's are called _3-clauses_ and ϕ is called a _3-formula_. A 3-formula is _satisfiable_ if there is an assignment of truth values (i.e. 'true' and 'false') to the propositional variables occurring in the 3-formula under which the 3-formula is true, i.e. at least one literal in each clause is true. Let _3-Sat_ be the set of satisfiable 3-formulas. The fundamental existence result for NP-complete sets is the

Theorem 1.3 (Cook [10]). 3-Sat is NP-complete.

As a further example, we show below in section 2.1 that

Theorem 1.3.1. The set $\{<a,b,c>: ax^2 + by = c \text{ solvable in } \omega\}$ is NP-complete.

For intractability proofs, the relevance of NP-completeness proofs is that NP-complete problems are the "hardest" problems in NP, or more formally:

Proposition 1.3.2. Let S be NP-complete. Then

(i) $S \in P \Leftrightarrow NP = P$ (+)

(ii) $S \in R \cap R^c \Leftrightarrow NP = R = R^c = NP^c$ (++)

Proof: (i) If $NP = P$, $S \in P$ is trivial; if $S \in P$, any $T \in NP$ has a deterministic polynomial time algorithm: "On input x, reduce T to S by computing $f_T(x)$, and then use the deterministic polynomial time algorithm for S to see if $f_T(x) \in S$. Output accordingly."

(ii) Again, $S \in NP = R = R^c \Rightarrow S \in R \cap R^c$ is trivial; and if $S \in R \cap R^c$ then by f_T for arbitrary $T \in NP$ we have a reduction showing as in (i) that $NP \subseteq R$ so $NP = R$; but also that $NP \subseteq R^c$, so $R \subseteq R^c$, and, by taking complements of members of R and R^c, $R^c \subseteq R$ so $NP = R = R^c$. By taking complements again $NP^c = R^c = R$. □

The process by which the method of NP-completeness provides evidence for intractability of a problem S can now be analyzed in two parts:

1. The proof that S is NP-complete. It is sufficient to reduce some known NP-complete problem to S; doing this requires insight into the particular nature of the problem S. For example, what may be required is knowledge of graph theory, if S belongs to that domain, or in the case of the examples in this study, of number theory.

2. We use insights of general complexity theory, namely the conjecture that the relevant inclusions in Diagram I.1 are strict, to

conclude from Proposition 1.3.2 that $S \notin P$ or $S \notin R \cap R^c$. These insights (which one hopes will eventually be superceded by rigorous proof) are based on experience in searching for efficient algorithms for problems in NP not known to be in P; thus the inference we make about S is based on insight gathered by research on algorithms in many areas besides the area of S. For example, the conjecture that $NP \cap NP^c \supsetneq P$ is based in part on the frustration of the search of several centuries for a method for determining the prime factorization of numbers, which has failed to yield anything better than exponential time methods (as we note in section 2.1 below, prime factorization is deterministic polynomial time computationally equivalent to a problem in $NP \cap NP^c$).

This unification -- the facility to bring insights from deeply developed areas of mathematics such as number theory to bear upon the complexity of problems in new or undeveloped areas -- is one of the principal strengths of complexity theory. The new methods for intractability proofs to be presented below are analogous to that of NP-completeness, and the analysis of 1. and 2. remains valid for these methods; the differences will be in the notions of reduction used and in different choices of inclusions in diagram II.1 for which the conjecture of strictness is relevant.

Before introducing the new methods, it is well to explain why there is a need for further methods. The method of NP-completeness has been used very widely and has been extremely successful. On the other hand, Ladner [17] has shown that if $NP \neq P$, there are problems in NP which are neither in P (or in fact in $NP \cap NP^c$) nor NP-complete. Thus it is in principle quite possible that some practical problems may be intractable but not NP-complete, and for these the method would not succeed in showing NP-completeness. A second motivation for stronger methods is to shift the burden of part 1 of the intractability proof, i.e., the part which must be done separately, to part 2. By allowing resources superior to deterministic polynomial time computation for reduction of problems, and hence broadening the class of complete problems, we will obtain potentially more general and more easily applicable methods.

The first new method, that of γ-completeness or nondeterministic NP-completeness, allows nondeterminism in the reduction algorithm. It may seem surprising that nondeterministic polynomial time reduction (as defined below) is not so powerful as to collapse all of NP into a single interreducibility class; that this is not so is due to the contrast of the (conjectured) nonclosure of NP under complementation with the symmetry of the condition (*) on reduction under complementation, given in the definition below:

<u>Definition 1.3.1</u>.

1) A <u>nondeterministic</u> <u>transducer</u> F is a nondeterministic Turing machine such that on every input x, at least one path halts with an output on its tape. Distinct computational paths on a given input may produce different outputs; we denote the set of outputs produced on input x by F(x).

2) Let A and B be sets. A is <u>γ-reducible</u> <u>to</u> B (notation $A \leq_\gamma B$) if and only if there is a nondeterministic transducer F which runs in polynomial time such that

$$\forall x \forall y \in F(x)[x \in A \Leftrightarrow y \in B] \quad . \qquad (*)$$

3) A set A is <u>γ-hard</u> if every set in NP is γ-reducible to A, and <u>γ-complete</u> if and only if it is γ-hard and in NP.

The utility of showing that a set is γ-complete for assessment of intractability can be expressed in a proposition analogous to Proposition 1.3.2.

<u>Proposition 1.3.3</u>. Let A be a γ-complete set. Then

$$A \in NP \cap NP^c \Leftrightarrow NP = NP \cap NP^c = NP^c . \qquad (+++)$$

<u>Proof</u>. The implication from right to left follows directly because $A \in NP$. To show the converse, assume A γ-complete and $A \in NP \cap NP^c$. Let S be an arbitrary set in NP, reducible to A by transducer F. To show $S \in NP^c$, consider the following nondeterministic polynomial time acceptor for S^c: "On input x, compute a value y of $F(x)$. As $A \in NP^c$, there is a nondeterministic polynomial time acceptor M for A^c. Run M on y, and accept x if M accepts y." Clearly this works. Hence $NP \subseteq NP^c$, and by taking complements, $NP^c \subseteq NP$ so $NP = NP \cap NP^c = NP^c$.

From this proposition we conclude that once we have understood a problem A sufficiently to show that A is γ-complete, the evidence gathered from other areas for the conjecture that the right hand side of (+++) is false is evidence that $A \notin NP \cap NP^c$. Thus A is bounded away from the classes of problems which are candidates for being tractable.

We give some applications of the new methods to problems not known to be NP-complete.

Theorem 1.3.4. The following sets are γ-complete.

1) $\{\langle a,c\rangle: (ax+1)y = c \text{ solvable in } \mathbb{Z}\}$

2) $\{\langle a,c\rangle: x^2-a^2y^2 = c \text{ solvable in } \mathbb{Z}\}$

3) $\{\langle p,q_1,\ldots,q_k\rangle$: p prime number, $q_1 \cdots q_k$ integer coefficient polynomials in sparse representation; such that, for some $n \in \omega$ $\prod_{i=1}^{k} q_i(n) \not\equiv 0$ modulo p$\}$ (Plaisted [29]).

The proofs of 1) and 2) are given in sections 2.4 and 2.3; 3) is shown in [29].

The next method we present, that of random completeness, is intermediate between that of γ-completeness and that of NP-completeness; it is obtained by requiring the nondeterministic reduction of Definition 1.3.1 to be a random nondeterministic computation.

Definition 1.3.2. 1) Let A and B be sets. A is random reducible to B (notation: $A \leq_R B$) if A is γ-reducible to B by a nondeterministic polynomial time transducer F such that for every input x, the computation paths of F on x which produce outputs have total weight $\geq \frac{1}{2}$ in the uniform weighting of the computation tree of F on X.

2) A set A is random hard if every set in NP is random reducible to A, and random complete if $A \in NP$ and A is random hard.

The characteristic property of random complete sets is:

Proposition 1.3.5. Let A be a random complete set. Then

(i) $A \in R \Leftrightarrow R = NP$

(ii) $A \in R^c \Leftrightarrow NP = R = R^c = NP^c$

Hence:

$$A \in R \cap R^c \Leftrightarrow NP = R = R^c = NP^c . \qquad (++)$$

Proof. If $A \in R$, a random algorithm for any set $S \in NP$ is obtained by composing the random reduction of S to A with a random polynomial time acceptor for A. If $A \in R^c$, i.e., $A^c \in R$, a random polynomial time algorithm for any set $S \in NP^c$ is obtained by composing the random reduction of S^c to A^c with a random acceptor for A^c. Hence $NP^c \subseteq R \subseteq NP$ so $NP = NP^c$ so $NP = R$.

As an application of this method, we show in section 2.3 the following result, which depends on an unproved conjecture from number theory, the Extended Riemann Hypothesis [25] (ERH):

Theorem 1.3.6 (ERH). The sets

$$\{<3^k,c>: x^2-(3^k)^2y^2 = c \text{ solvable in } \mathbb{Z}\}$$
$$\{<p,c>: (px+1)y = c \text{ solvable in } \mathbb{Z};\ p \text{ prime}\}$$

are random complete.

Finally, we present two new methods using a weaker notion of reduction instead of the many-one reduction of the form (*), unfaithful γ-completeness and unfaithful random completeness. This appears to be the first place in the literature that use is made of the type of reduction, unfaithful reduction, defined here. The key to these new, stronger methods is that weaker forms of reduction may not yield an efficient method for solving one problem using a solution of another, but still relate the complexity of the problems involved in a significant way.

Definition 1.3.3. Let A and B be sets. A is unfaithfully γ-reducible to B (notation: $A \le_{u\gamma} B$) if and only if there is a polynomial time nondeterministic transducer F such that

(i) $\forall x \forall y \in F(x) \quad x \in A \Rightarrow y \in B$

(ii) $\forall x \exists y \in F(x) \quad x \in A \Leftrightarrow y \in B$

A set $A \in NP$ is unfaithfully γ-complete if and only if every set in NP is unfaithfully γ-reducible to A.

The characteristic proposition in this case is

Proposition 1.3.7. If A is unfaithfully γ-complete, then

$$A \in NP \cap NP^c \Leftrightarrow NP = NP \cap NP^c = NP^c \qquad (+{+}+)$$

Proof. As in the proof of Proposition 1.3.3, we have $NP^c \subseteq NP$, by composing the γ-reduction of the complement S^c of an arbitrary set $S \in NP^c$ with a nondeterministic polynomial time acceptor for A^c.

Definition 1.3.4. Let A and B be sets. A is unfaithfully random reducible to B ($A \le_{uR} B$) if and only if A is unfaithfully γ-reducible to B by a transducer F such that for any input x, the computation paths of F on x which produce outputs y satisfying

$$x \in A \Leftrightarrow y \in B$$

have total weight $\geq \frac{1}{2}$ in the uniform weighting of the computation tree. $A \in NP$ is unfaithfully random complete if and only if every set in NP is unfaithfully random reducible to A.

We obtain the

Proposition 1.3.8. Let A be an unfaithfully random complete set. Then

$$A \in R \cap R^c \Leftrightarrow NP = R = R^c = NP^c \quad . \qquad (++)$$

Proof. Assume $A \in R^c$, $S \in NP^c$ arbitrary. Composing the random reduction of S^c to A^c with a random acceptor for A^c, we have $NP^c \subseteq R$, i.e. the result follows as in Proposition 1.3.5.

It is a remarkable fact about unfaithful reductions that completeness in the unfaithful sense gives us the same information about intractability as completeness in the corresponding faithful sense: the conclusions of Propositions 1.3.7 and 1.3.8 are the same as those of 1.3.3 and 1.3.5.

As an application, we will show in sections 2.3 and 2.4:

Theorem 1.3.9.

1) $\{<3^k,c>: x^2-(3^k)^2y^2 = c \text{ solvable in } \mathbb{Z}\}$ is unfaithfully random complete.

2) $\{<p,c>: (px+1)y = c \text{ nonzero solution in } \mathbb{Z}; \, p \text{ prime}\}$ is unfaithfully random complete.

Theorem 1.3.9 illustrates the types of advantages to be gained from use of the unfaithful reductions: We are able to dispense with the unproved hypothesis ERH used to show random completeness as well as with Ankeny's elaborate analysis of the least quadratic nonresidue [3] which underlies Miller's primality testing algorithm [25], used in the proof of Theorem 1.3.6. In other circumstances we may perhaps show unfaithful random completeness of a problem which even under the assumption of ERH we cannot show random complete. Thus the advantages are -- simplification of proofs, elimination of hypotheses, and more powerful intractability results than can be shown otherwise.

1.4 Conclusion and Open Problems

Four new methods for intractability proofs have been introduced. All of these methods are potentially of broader applicability than that of NP-completeness, as well as potentially easier to apply. Even problems which are NP-complete might more easily be shown unfaithfully γ-complete. It is thus with the binary quadratic solvability problems presented as examples: We cannot be really sure at this point that these problems are not NP-complete, but at present the only way known to show them intractable is to use our new methods. From a practical point of view, the intractability is what one wants to show, and preferably as efficiently as possible.

What can we say about the question whether (assuming $NP \neq NP^c$) NP-completeness and γ-completeness are really distinct? One has intuitive grounds to believe that nondeterministic reduction could indeed be properly more powerful than deterministic reduction (both in polynomial time): Nondeterministically one can "guess" a configuration related to the problem to which one is reducing, which is special in that it is a relatively small configuration with special properties, but such that no polynomial time method can deterministically locate such a configuration. For example, a small prime p in a given residue class (k modulo h) forms such a configuration, and this is a key fact used in the proofs of the theorems in this section. One can afford to write out the prime, and a proof of its primality, but no method is known to efficiently determine such a prime.

A variety of open problems are suggested by the methods of this section; we list some with most direct theoretical appeal:

<u>Open Problem 1</u>. Is every unfaithfully random hard set γ-hard? If not, does this at least hold for sets in NP (the complete sets)?

<u>Open Problem 2</u>. Assuming $NP \supsetneq R$, extend the methods of Ladner [17] to show that there are sets in NP which are not NP-complete nor in R.

<u>Open Problem 3</u>. Similarly, assuming $NP \neq NP^c$, show that there are sets in NP which are neither γ-complete nor in $NP \cap NP^c$.

<u>Open Problem 4</u>. If $NP \subsetneq R$, then there is no random complete set in R. But is there a γ-complete set in R? an unfaithfully γ-complete set? an unfaithfully random complete set?

<u>Open Problem 5</u>. One clearly has the inclusions

$$\text{NP-hard} \subseteq \text{random-hard} \begin{array}{l} \subseteq \text{unfaithfully random hard} \\ \subseteq \gamma\text{-hard} \end{array} \subseteq \text{unfaithfully } \gamma\text{-hard} .$$

Are all these inclusions strict? Are they strict for the corresponding complete sets? (Assume of course the relevant conjectures in Diagram I.1.)

None of the methods developed here applies directly to classification of complexity of problems in classes closed under complement such as $NP \cap NP^c$, in the sense that complete problems never lie in this class.

Open Problem 6. Is there an interesting sense in which problems in $NP \cap NP^c$ can be complete for $NP \cap NP^c$?

Another problem is suggested by the examples given, all of which are rather number theoretic.

Open Problem 7. Find natural non-number theoretic combinatorial problems which are unfaithfully γ-complete but not NP-complete.

The present authors have not looked in depth for such non-number-theoretic examples; it is conceivable that the graph isomorphism problem is a candidate.

2. The Solvability Problem for Binary Quadratics

2.1 Introduction

Many number theoretical questions are formulated or can be formulated as questions about the solvability of Diophantine equations (i.e., $p(x_1 \cdots x_n) = 0$ for a multivariable polynomial $p(x_1 \cdots x_n)$ with integer coefficients) in natural numbers or integers. This becomes clear upon examination of the section on Diophantine problems in any standard number theory text; it is in fact made mathematically precise by Matijasevic's theorem [22] that all recursively enumerable sets are Diophantine (i.e., the elements of the set S correspond to the parameter values a for which an appropriate Diophantine equation $p_S(a, x_1 \cdots x_n) = 0$ has a solution). It is therefore of fundamental importance to consider algorithms for deciding the solvability of Diophantine equations, as Hilbert stressed in asking for such algorithms in the 10th of his famous set of mathematical problems in 1900 [14]. One wishes to know (a) the complexity of deciding solvability for various classes of Diophantine equations, and especially (b) for which subclasses of Diophantine equations a feasible (i.e., deterministic polynomial time) algorithmic procedure to decide solvability exists.

It follows from Matijasevic's theorem that for some fixed n, the set of solvable n-variable Diophantine equations is nonrecursive. Much effort has been devoted to determining the minimum n for which this is true. The best published result is $n \leq 13$ [23]; Matijasevic has improved this to $n \leq 9$ [24]. In [23], it is conjectured that $n = 3$ may be possible, though this cannot be shown by present methods.

Diophantine equations in two variables form a classical topic in number theory. For degree 3 or greater, the solvability problem is generally quite difficult to approach; recent work of A. Baker [4] [5], based on his fundamental results on linear forms in logarithms, gives the first rather broadly applicable decision procedures for the solvability problem for some such types of equations. On the other hand, the solvability problem for binary quadratics encompasses a number of fundamental problems of number theory, such as primality testing, factorisation, and quadratic residuacity. Thus these equations have been intensively studied for the past several centuries, especially starting with Lagrange and Gauss [12]. We give an overview of complexity results for solvability problems for binary quadratics. (All coefficients are assumed positive.)

(a) $ax^2 + bx + c = dy$

Can be simplified to the question of quadratic residuacity, i.e.

(a)' $x^2 \equiv a'$ modulo b'

In P, using the quadratic reciprocity theorem for the Jacobi symbol.

(b) $(x+2)(y+2) = a$ (compositeness of a)

Clearly in NP; in NP^c by Pratt [30]; if ERH (see below) is true, in P by Miller [25].

(c) $(x+2)\big(y - (a+1)(b-x-3)\big) = a$

This equation is solvable if a has a nontrivial divisor ($\neq 1$, $\neq a$) below b; it is in $NP \cap NP^c$ and deterministic polynomial time equivalent to the problem of prime factorisation of numbers. See Miller [25].

(d) $x^2 - ay^2 = 1$

The Pell equation, known to be solvable if and only if a is not a square (in P). See [28].

(e) $x^2 - ay^2 = -1$

By significant forthcoming work of Lagurias [18], this problem is in $NP \cap NP^c$. Neither membership in NP nor in NP^c is easily shown.

(f) $x^2 - a^2y^2 = c$

Clearly in NP; in P if restricted to c with less than log log c prime factors. Unfaithfully random complete, even for $a = 3^k$. Given ERH, random complete. See section 2.3.

(g) $x \cdot (ax + by) = c$

Unfaithfully random complete, with ERH random complete, by same proof as for (f). See section 2.3.

(h) $(ax + b)y = c$

as (f). The intractability results hold even for

(i) a: power of 3; $b = 2$

(ii) a: prime; $b = 1$.

See section 2.4

(i) $ax^2 + by = c$

NP-complete by Manders-Adleman [21]. See section 2.2.

(j) $ax^2 + by^2 = c$

Clearly in NP. Otherwise: Open Problem. In P, if c prime. See Mordell [27].

(k) $x^2 - dy^2 = m$

Pell equation, more general from. Not even known to be in NP! This equation has considerable significance in algebraic number theory (see Borevich and Shafarevich [8]); its complexity constitutes the major open problem concerning binary quadratics.

Before proceeding to the proofs, we note the following conventions:

(a,b)	denotes the greatest common divisor of a and b; thus $(a,b) = 1$ means that a and b are relatively prime.
$a \mid b$	abbreviates 'a divides b'; $a \nmid b$ abbreviates 'a does not divide b'.
$[a]$	abbreviates the greatest integer less than b.
ERH	refers to the Extended Riemann Hypothesis, e.g. as defined in [25]. This is an as yet unproved numbertheoretic conjecture. We indicate that ERH is used in the proof of a statement by adding '(ERH)' to the statement.
$\lvert x\rvert$	denotes the absolute value of x.

2.2 The Complexity of $ax^2 + by = c$

If integer values for x and y are allowed, the solvability problem for this equation is equivalent to the question: "Is a^*c a quadratic residue modulo b?" where a^* is the inverse of a modulo b:

$$ax^2 \equiv c \Leftrightarrow x^2 \equiv a^*c \pmod{b} .$$

This problem is easily decided in polynomial time, using the quadratic reciprocity law for the Jacobi symbol [28]. Moreover, Adleman, Manders, and Miller [2] have shown, using ERH, that if b is prime or presented fully factored, some positive solution x of $x^2 \equiv a^*c \bmod b$ (if any exists) can be found in deterministic polynomial time. On the other hand, we shall see directly that for arbitrary b, even presented fully factored, and regardless of the assumption of ERH, the problem of finding the least positive solution to this congruence is NP-hard (in the sense that given an oracle for this problem, we can construct a deterministic polynomial time solution to an NP-complete problem.

If we consider only nonnegative integer solutions for x and y, then the size of x and y will play a role, over and above the quadratic residuacity of a^*c modulo b: a solution x must be a square root of a^*c modulo b satisfying $0 \leqslant x \leqslant (c/a)^{1/2}$. Thus the "naive" approach to this solvability problem is perhaps, given the factorisation of b, to determine (ERH) the square roots of a^*c modulo each maximal prime power factor of b, and use the Chinese Remainder Theorem to reconstruct all square roots of a^*c modulo b to accept the input $<a,b,c>$ if one is encountered which is below the size bound. Unfortunately, b may have almost $\log c$ distinct prime factors, giving $2^{\log b} = b$ solutions; hence this method may be expected to require exponential time on infinitely many inputs. Moreover, no efficient method is known to choose a set of residues modulo prime power factors of b which will yield a small square root modulo b after the Chinese Remainder process.

We will show below that this solvability problem is indeed intractable:

Theorem 2.2.1. The following sets are NP-complete:

(i) $\{<a,b,c>: ax^2+by = c \text{ solvable in } \omega\}$

(ii) $\{<a,b,c>: x^2 \equiv a \text{ modulo } b,\ 0 < x < c \text{ solvable in } \omega\}$

This theorem was shown in Manders and Adleman [21]. We now give a shorter proof which more clearly exposes the crucial step in the construction. This theorem is the only known NP-completeness result for a class of binary quadratics. From the analysis given above, it is clear that the ability to find the least positive square root modulo a number will yield a polynomial time solution to the NP-complete problem (ii), as was promised above.

Proof. The proof is by reduction of the set partition problem:

Proposition (Karp [15]). The set partition problem: "Given a finite list of positive integers $a_1,\ldots,a_n$, not necessarily distinct, is there a binary partition of the list such that the two classes have equal sum?" is NP-complete.

Notation. The sums over the two classes of a binary partition of the list will be written as $\sum a_i$, $\sum' a_i$; i.e. we suppress reference to the index sets in order to simplify notation.

We first give the reduction algorithm:

(A) "On input $a_1,\ldots,a_n$, compute:

$p_1,\ldots,p_n$ first n odd primes;

k minimal such that $2^{k-1} > \sum_{i=1}^{n} a_i$;

θ_i, $i = 1,\ldots,n$ minimal positive integers such that

$\theta_i \equiv a_i$ modulo 2^{k-1},

$\theta_j \equiv 0$ modulo $\prod_{j\neq i} p_j^k$,

$\theta_i \not\equiv 0$ modulo p_i;

$K = \prod_{i=1}^{n} p_i^k$, $H = \sum_{i=1}^{n} \theta_i$;

u: inverse of 2^{k-1} modulo K

Output:

(i) $2^{2k-2}x^2 + Ky = H^2$, i.e. $<2^{2k-2},K,H^2>$;

(ii) $x^2 \equiv u^2H^2$ modulo K, $0 < x < H/2^{k-1}$, i.e. $<u^2H^2,K,[H2^{-k+1}]>$."

A runs in polynomial time: We can afford to sieve for the first n odd primes. For each i, a minimal positive solution λ to the first two conditions on θ_i can be found by the Chinese Remainder Theorem; then at least one of λ and $\lambda + 2^{k-1} \prod_{j \neq i} p_j^k$ will not be divisible by p_i (as the difference between the two is not divisible by p_i) and thus will satisfy the third condition on θ_i as well as the first two. All other computation is routinely polynomial time.

The crucial point in the reduction is the use of the θ_i, K, and H as explained in the following key lemma:

<u>Lemma 2.2.2</u> (Manders-Adleman [21]). Let θ_i, $i = 1,\ldots,n$, K, and H be as defined in the algorithm. The general solution in integers of the system

(1) $0 \leq |z| \leq H$,

(2) $(H+z)(H-z) \equiv 0$ modulo K,

is given by

(3) $z = \sum_{i=1}^{n} \alpha_i\theta_i$, $\alpha_i \in \{-1,+1\}$, $i = 1,\ldots,n$.

<u>Proof</u>. First assume that z is of the form (3). Then $|z| < \sum\theta_i = H$. By definition of H and θ_i, $H \equiv \theta_i$ modulo p_i^{n+1}, so

$$H \equiv \theta_i \equiv z \text{ modulo } p_i^{n+1} , \quad \alpha_i = 1 ;$$
$$H \equiv \theta_i \equiv -z \text{ modulo } p_i^{n+1} , \quad \alpha_i = -1 ;$$

so that z also satisfies (2).

Next assume that z satisfies (1), (2). Then for each i

$$p_i^{n+1} \mid (H+z)(H-z) .$$

On the other hand, for each i, p_i divides at most one of $H+z$, $H-z$. For otherwise $p_i \mid (H+z)+(H-z) = 2H$, and p prime > 2, so $p \mid H = \sum\theta_i$. But by definition of θ_i, $p_i \mid \theta_{i'}$ for all $i' \neq i$. Then also $p_i \mid \theta_i$, contradicting the definition of θ_i.

Thus for each i, $p_i^{n+1} \mid H+z$ or $p_i^{n+1} \mid H-z$, but not both.

Define

$$\alpha_i = 1 \ , \ \text{if } p_i^{n+1} \mid H-z \ ;$$
$$\alpha_i = -1 \ , \ \text{if } p_i^{n+1} \mid H+z \ ;$$
$$z' = \sum\alpha_i\theta_i \ .$$

Then for each j, $z' \equiv \alpha_i\theta_i \equiv \alpha_i H \equiv z$ modulo p_i^{n+1}, so

$$z' \equiv z \text{ modulo } K, \text{ and}$$

$$\left.\begin{array}{l} -H \leqslant z' \leqslant H \\ -H \leqslant z \leqslant H \end{array}\right. \Rightarrow |z'-z| \leqslant 2H < K,$$

where $2H < K$ because

$$0 < \theta_i < 2\cdot 2^{k-1}\cdot K/p_i^k \ ,$$
$$2H < 4nK(2^{k-1}/3^{k-1})\cdot \Pi p_i^{-1} < K \ , \quad \text{as} \quad 4n < \Pi p_i \ .$$

Thus $z = z'$ and hence z is of the form (3). □

We now prove correctness of the algorithm:

$$\sum a_i = \sum{}' a_i \Leftrightarrow \sum_{i=1}^n \alpha_i a_i = 0, \ \alpha_i \in \{-1,+1\}$$
$$(\text{since } 2^{k-1} > |\sum_{i=1}^n x a_i|)$$
$$\Leftrightarrow \sum_{i=1}^n \alpha_i a_i \equiv 0 \text{ modulo } 2^{k-1}$$
$$\Leftrightarrow \sum_{i=1}^n \alpha_i \theta_i \equiv 0 \text{ modulo } 2^{k-1} \text{ (by the lemma)}$$
$$\Leftrightarrow \text{(I)} \quad z \in \mathbf{Z}, \ 0 < |z| < H$$
$$\text{(II)} \quad (z+H)(z-H) \equiv 0 \text{ mod } K$$
$$\text{(III)} \quad z \equiv 0 \text{ modulo } 2^{k-1}$$

$\Leftrightarrow$ (requiring $z \in \omega$ without loss of generality, as (I)-(III) is symmetric in z; and converting (III) into $z = 2^{k-1}x$, $x \in \omega$)

(I)' $x \in \omega$, $0 < 2^{k-1}x < H$

(II)' $2^{2k-2}x^2 - H^2 \equiv 0$ modulo K

$\Leftrightarrow$

(i) $x, y \in \omega$; $2^{2k-2}x^2 + yK = H^2$

for clearly (i) $\Rightarrow$ (I)-'(II)'; conversely, if (II)' is satisfied, the equation has a solution with integral y and $x \in \omega$; but any such solution with negative y must have x violating (I)', and any such solution with $y \in \omega$ must have x satisfying (I)'. Hence the restriction $y \in \omega$ is equivalent to (I)'.

(ii) $x^2 \equiv H^2 \cdot (2^{2k-2})^{-1}$ modulo K, $0 < x < H/2^{k-1}$. $\square$

2.3 <u>The Complexity of $x^2 - a^2y^2 = c$ and $x^2 + axy = c$</u>

Rewriting the equation as $(x+ay)(x-ay) = c$, we see that it is solvable if and only if c can be written as a product of two factors congruent modulo 2a. Thus this solvability problem is in NP -- "Guess" factors f_1, f_2 and check that $f_1 \equiv f_2$ modulo 2a and $f_1 \cdot f_2 = c$. Deterministically, we can solve the problem of prime factorisation of c and testing all binary partitions of the factors of c; the difficulty is that infinitely many c have about $\log c/\log\log c \cong |c|$ distinct prime factors, so that we get about $2^{|c|}$ distinct partitions, requiring almost exponential time for the complete test. The intractability proof given below provides evidence that non-trivial methods of solution would not be expected to improve this to polynomial time.

The proofs are by reduction of the set partition problem (defined in section 2.2).

Theorem 2.3.1. The set $\{\langle 3^k, c\rangle : x^2-(3^k)^2y^2 = c \text{ solvable in } \mathbb{Z}\}$ is

(i) unfaithfully random complete,

(ii) (ERH) random complete.

Proof (in five parts).

(I) Consider the following chain of equivalences, for an arbitrary list of integers $a_1,\ldots,a_n$:

$$\sum a_i = \sum' a_i$$

$$\Leftrightarrow \ (1)$$

$$\sum a_i \equiv \sum' a_i \text{ modulo } 3^k, \quad 3^k > \sum|a_i| + \sum'|a_i|$$

$$\Leftrightarrow \ (2)$$

$$\Pi 4^{a_i} \equiv \Pi' 4^{a_i} \text{ modulo } 3^{k+1}$$

$$\Leftrightarrow \ (3)$$

$$\Pi h_i \equiv \Pi' h_i \text{ modulo } 3^{k+1}, \quad \text{where } h_i \text{ prime, } h_i \equiv 4^{a_i} \text{ mod } 3^{k+1}$$

$$\Leftrightarrow \ (4)$$

$$(x+3^{k+1}y)(x-3^{k+1}y) = c, \quad \text{for } x, y, c \text{ given by}$$

$$c = (\Pi h_i)\cdot(\Pi' h_i)$$

$$x + 3^{k+1}y = \Pi h_i, \quad x - 3^{k+1}y = \Pi' h_i$$

The equivalence (1) follows because the modulus 3^k is chosen so large as to exceed the largest possible difference between $\sum a_i$ and $\sum' a_i$. (2) follows by a technical lemma.

Lemma. For $k \geqslant 0$, $a \equiv b$ modulo $3^k \Leftrightarrow 4^a \equiv 4^b$ modulo 3^{k+1}.

Proof. We have $4^a \equiv 4^b$ modulo $3^{k+1} \Leftrightarrow 4^{a-b} \equiv 1$ modulo 3^{k+1}; the latter congruence holds if and only if the order $o_k(4)$ of 4 modulo 3^{k+1} divided a-b, i.e. $a \equiv b$ modulo $o_k(4)$. To show that $o_k(4) = 3^k$ for $k \geqslant 0$, we show by induction for $k \geqslant 0$ that

$$4^{3^k} = 1 + y_k \cdot 3^{k+1} \ ; \quad 3 \nmid y_k \qquad (*)$$

For from this clearly $4^{3^k} \equiv 1$ modulo 3^{k+1}, so $o_k(4) \mid 3^k$; but also for $k \geqslant 1$

$$4^{3^{k-1}} = 1 + y_{k-1} \cdot 3^k \not\equiv 1 \text{ modulo } 3^{k+1}, \quad \text{as } 3 \nmid y_{k-1} \ .$$

This gives $o_k(4) = 3^k$ for $k \geqslant 1$; but also the order $o_0(4)$ of 4 modulo 3 is clearly $3^0 = 1$.

Now to show (*), for $k = 0$ we have $4^{3^0} = 1 + y_0 \cdot 3$, $y_0 = 1$ satisfying (*). Assuming (*) for $n = k \geqslant 0$, we find for n+1:

$$\begin{aligned} 4^{3^{n+1}} &= (4^{3^n})^3 = (1 + y_n \cdot 3^{n+1})^3 = 1 + 3y_n \cdot 3^{n+1} + 3y_n^2 \cdot 3^{2n+2} + y_n^3 \cdot 3^{3n+3} \\ &= 1 + y_{n+1} \cdot 3^{n+2} \ , \end{aligned}$$

where

$$y_{n+1} = y_n + y_n^2 \cdot 3^{n+1} + y_n^3 \cdot 3^{2n+1} = y_n(1+3y')$$

so that indeed $3 \nmid y_{n+1}$ because $3 \nmid y_n$. $\square$ (Lemma)

The equivalence (3) is direct from the congruence condition on the h_i; for (4) note that if $\Pi h_i \equiv \Pi' h_i$ modulo 3^{k+1}, then Πh_i and $\Pi' h_i$ differ by <u>twice</u> 3^{k+1} because all h_i are prime > 3^{k+1} and hence Πh_i, $\Pi' h_i$ are both odd. Thus these can indeed be written in terms of integers x, y as desired. Conversely, any two factors of c solving the equation are products of h_i's because the h_i are the prime factors of c; the factorisation is thus of the form Πh_i, $\Pi' h_i$

as desired to show the existence of an equal sum partition of the a_i.

(II) We see from the equivalences that, given any finite list of integers a_i, the list has an equal sum partition if and only if the associated equation for suitable c, k has a solution in integers; moreover, c and k depend on the list of a_i as a whole rather than on the partition considered in the equivalences. To obtain our reductions, we must thus find a method for computing c and k within the allowed resource bounds. For the unfaithful reduction, we use the following nondeterministic algorithm:

(A1) "On input $<a_1,\ldots,a_n>$, set k minimal with $3^k > \sum|a_i| + \sum'|a_i|$ (and also exceeding a constant k_0 as explained in the proof; this non-asymptotic effect is irrelevant to the running time). Choose, for each i, $6kn$ distinct x congruent to 4^{a_i} modulo 3^{k+1}, each less than 3^{3k+3}; from these, set h_i = the least x_i such that $2^{x_i-1} \equiv 1$ modulo x_i, or, if no such x_i is found, halt without output. Set $c = \prod_{\text{all } i} h_i$; output $<3^{k+1},c>$."

For the random reduction, we use the algorithm (A2) obtained from (A1) by replacing the Fermat test "$2^{x_i-1} \equiv 1$ modulo x_i" by a test for primality of x_i, using Miller's algorithm [25]. We now show that these algorithms have the desired properties. It is directly clear from the programs of (A1) and (A2), and, respectively, the time analysis for evaluating a^b modulo m (see Lehmer [19]) and the analysis of Miller's algorithm (see Miller [25]) given (ERH), that the algorithms run in nondeterministic polynomial time. It remains to analyze the density of successful paths in the algorithms. This requires number theoretic analysis of the density of primes in short initial segments of arithmetic progressions.

(III) In the following, $\phi(D)$ is the number of positive integers less than D and relatively prime to D; for ℓ relatively prime to D, $\pi(x,D,\ell)$ is the number of primes congruent to ℓ modulo D and less than x, $\lg(x)$ is the logarithm base e of x, and

$$\ell i(x) = \int_z^x \frac{dt}{\lg t} .$$

The major number-theoretic result needed in the proof is

<u>Proposition</u> (Barban, Linnik, and Tshudakov [6]). Let $p \geqslant 3$ be prime, $D = p^n$ $(n=1,2,\ldots)$, $\varepsilon > 0$ arbitrarily small, M arbitrarily large, and ℓ relatively prime to D. Then, for any $x \geqslant D^{8/3+\varepsilon}$

$$\pi(x,D,\ell) = \frac{1}{\phi(D)} \ell i(x)\left(1 + O(\log(x))^{-M}\right)$$

where the constant implied in $O(\cdot)$ depends only on ε and M.

Applying this we find

<u>Lemma 2.3.2</u>. There are absolute positive constants c, n_0 such that for p prime $\geqslant 3$, $n \geqslant n_0$, $D = p^n$, $(\ell,D) = 1$, the density of primes in the class up to D^3 satisfies

$$\frac{\pi(D^3,D,\ell)}{D^2} > \frac{1}{6 \lg D} .$$

Proof. In the proposition, choose $M = 1$, $\varepsilon = \frac{1}{3}$. Let c be the constant then implied in $O(\cdot)$. Choose n_0 minimal such that $D^3 = p^{3n_0} \geqslant \max\{u, e^{2c}\}$, where u is the solution to $\ell i(u) = u/\lg(u)$. (This is unique and for $x > u$, $\ell i(x) > x/\lg(x)$). Thus we find for $n > n_0$, $D = p^n$, $x = D^3$:

$$\frac{\pi(D^3, D, \ell)}{D^2} \geqslant \frac{\ell i(D^3)}{D^2 \phi(D)} \cdot \frac{1}{2} > \frac{1}{2 \lg(D^3)} = \frac{1}{6 \lg(D)} . \qquad \square$$

(IV) At this point we first conclude the proof that (A2) is a correct random reduction algorithm. Correctness is in this case clear from the program of (A2) and the equivalences considered above; we can be sure that at least half of the paths pick a prime congruent to 4^{a_i} modulo 3^{k+1} (for each i) if for each i, the proportion of failing guess sequences is at most $1/2^n$ (n is the number of a_i's). Thus we want to choose the length f of test sequences for each a_i minimal such that

$$\left(1 - \frac{1}{6 \lg D}\right)^f < 2^{-n}$$

$$-n \lg 2 > f \lg\left(1 - \frac{1}{6 \lg D}\right)$$

$$f > \frac{n \lg 2}{\lg\left(1 - \frac{1}{6 \lg D}\right)}$$

and for $D = 3^{k+1}$, the right hand side is monotone increasing asymptotic to $6nk \lg 3 \lg 2$. As $\lg 3 \lg 2 > 1$, asymptotically $f > 6nk$ suffices, and one can verify that this suffices for $k \geqslant 3$. Thus (A2) does indeed give a random reduction.

(V) To show that (A1) gives an unfaithful random reduction (ERH), we must verify that (a) (A1) gives an unfaithful γ-reduction, in which (b) "at least $\frac{1}{2}$" of the paths are faithful. To show (a), note that the only condition in the construction of the equivalences of part (I) of the proof which might fail to be realized by (A1) is the primality of h_i for one or more h_i on a given output-producing path. But if some of the h_i are not prime, the equivalences (1)-(3) still hold, as well as the implication from left to right in (4). This shows that condition (i) in the definition of unfaithful γ-reduction is satisfied; condition (ii) is also clearly satisfied as certainly (by the argument of part (4) for (A2), for example) some paths on any input produce prime h_i for each i.

To show that at least half the paths of (A1) on any given input are faithful, we show that at least half the paths produce prime h_i for each i. Clearly the number of sequences (for fixed i) of length f for which the Fermat test base 2 yields a positive result for a nonprime is to the number of such sequences for which the test produces a prime as the number of pseudoprimes (nonprime solutions to the Fermat test base 2) is to the number of primes, in the set from which the choices are made. The number of primes is at least that given by Lemma 2.3.2; for the total number of pseudoprimes (and a fortiori the number of pseudoprimes in the residue class in question) we have

Proposition (Erdös [11]). Let $p(x)$ be the number of pseudoprimes less than x.

$$p(x) < x \exp(-c(\lg x \cdot \lg \lg x)^{1/2}) ,$$

for some positive constant c.

This result is to be compared to ($x = D^3$)

$$\pi(D^3, D, \ell) > x \exp - (\lg \lg x + c')$$

and as even $\exp(-c(\lg x)^{1/2})$ is eventually vanishing compared to $\exp - \lg \lg x$, it is clear that there are essentially no pseudoprimes, and that the estimate of f given above in (4) again shows that the choice of f made in (A1) is asymptotically adequate. We regard the algorithm as modified to have larger f for the small values of k for which the asymptotic situation is not realised (if any); this does not affect the asymptotic running time. This completes the proof. □

In the equation $x^2 - a^2y^2 = c$, we choose $a = 3^{k+1}$ in Theorem 2.3.1 for two reasons. By choosing 'a' a power of a fixed prime, we found an easily computable (namely constant 4) element of large order modulo a by the construction of Lemma 2.3.2. This allowed us to make the crucial exponentiation transition of equivalence (2), from an additive partition problem to a multiplicative partition problem, without the resulting modulus being too large. The second reason was in order to apply the result of Barban, Linnik, and Tshudakov. For arbitrary moduli the same result can be shown, but only assuming (ERH).

From Theorem 2.3.1 we of course directly have the

Corollary 2.3.3. The set

$$\{\langle a,c\rangle : x^2 - a^2y^2 = c \text{ solvable in integers}\}$$

is (i) unfaithfully random complete;

(ii) (ERH) random complete.

The arguments used above give us results about a slightly different form of binary quadratic as well. Consider the line from our sequence of equivalences in the proof of Theorem 2.3.1:

$$\Pi h_i \equiv \Pi' h_i \text{ modulo } 3^{k+1} .$$

Clearly we can also set $x = \Pi h_i$, $\Pi' h_i = x + 3^{k+1}y$, $c = \Pi_{i=1}^{n} h_i$, to get

Corollary 2.3.4. The set

$$\{\langle a,c\rangle : x(x+ay) = c \text{ solvable in } \mathbb{Z}\}$$

is (i) unfaithfully random complete;

(ii) (ERH) random complete;

even if we restrict a to be of the form 3^k, $k \geqslant 1$.

2.4 The Complexity of $(ax+b)y = c$

The solvability problem for $(ax+b)y = c$ is again a multiplicative partition problem with a congruence constraint: "Does c have a factor congruent to b modulo a?" Naive analysis yields an algorithm requiring exponential running time for infinitely many c, namely the c which are highly composite. Below we demonstrate the intractability of the problem. In fact our analysis of a number of special cases, all intractable, will pinpoint origins of the complexity of the problem; the results suggest that compositeness of c is indeed by itself sufficient to make the problem hard. On the other hand the naive analysis shows that compositeness of c is necessary to make the problem hard. Clearly the problem restricted to c with no more than

$k \log\log c$ prime factors (not necessarily distinct), for any fixed k, is in P.

As in the preceding section, the reduction involves exponentiation, to convert an additive partition problem to the desired multiplicative one. This time we reduce knapsack:

<u>Proposition</u> (Karp [15]). The <u>knapsack problem</u>: "Given a list $a_1 \cdots a_n$, $b \in \mathbb{Z}$, not necessarily distinct, is there $S \subseteq \{a_1 \cdots a_n\}$: $\sum_S a_i = b$?" is NP-complete. (In fact the proof in [15] shows NP-completeness for the problem restricted to $a_1 \cdots a_n > 0$; we will use knapsack in this form.)

In this section, we shall prove:

<u>Theorem 2.4.1</u>. The set

$$\{<3^k, c>: (3^k x + 2)y = c \text{ solvable in } \mathbb{Z}\}$$

is (i) unfaithfully random complete,

(ii) (ERH) random complete.

<u>Remark</u>. The method of the proof works for p^k instead of 3^k, for any prime $p \geqslant 3$.

<u>Theorem 2.4.2</u>. The set

$$\{<p, c>: p \text{ prime}, (px+1)y = c \text{ has nonzero solution in } \mathbb{Z}\}$$

is (i) unfaithfully random complete,

(ii) (ERH) random complete.

Proof of Theorem 2.4.1. First note that we can reduce the knapsack problem to the case where $b = 1$, for

$$\sum a_i = b \Leftrightarrow \sum 2a_i + 2b - 1 = 1 \quad \text{(summation over some subset of } a_1 \cdots a_n)$$

so we set $a_i \leftarrow 2a_i$, $n \leftarrow n+1$, $a_n \leftarrow 2b-1$. Now we have a series of equivalences similar to those given before:

$$\sum a_i = 1 \quad \text{(summation over a subset of } a_1 \cdots a_n)$$

$$\Leftrightarrow (1)$$

$$\sum a_i \equiv 1 \text{ modulo } 2\cdot 3^k, \quad 2\cdot 3^k > 1 + \sum_{i=1}^n |a_i|$$

$$\Leftrightarrow (2)$$

$$\Pi 2^{a_i} \equiv 2 \text{ modulo } 3^{k+1}$$

$$\Leftrightarrow (3) \quad \text{setting } h_i \equiv 2^{a_i}, \quad h_i \text{ prime}, \quad i = 1,\ldots,n$$

$$\Pi h_i \equiv 2 \text{ modulo } 3^{k+1}$$

$$\Leftrightarrow (4) \quad \text{setting } c = \Pi_{i=1}^n h_i, \quad x = (\Pi h_i - 2)/3^{k+1}, \quad y = c/\Pi h_i$$

$$(3^{k+1}x + 2)y = c$$

It will now be clear how the reduction algorithms are to be chosen; the analysis is parallel to that in the proof of Theorem 2.3.1 and will be omitted. □

Proof of Theorem 2.4.2. We first reduce the knapsack problem to the multisack problem: "Given integers $a_1 \cdots a_n$, b; $a_i > 0$; does a nonempty subset of $a_1 \cdots a_n$ sum to an integer multiple of b?" In fact we show that the special case: $b = 3^k$, $k > 0$, is NP-complete. For the reduction, given knapsack input $a_1 \cdots a_n$, b, find

k minimal > 0 such that

$$3^k > |b| + \sum_{i=1}^{n} |a_i| .$$

Output the multisack problem $a_1 \cdots a_{n+1}$, 3^k, where $a_{n+1} = 3^k - b$. It is easily verified that this is a valid (and obviously deterministic polynomial time) reduction.

We next give a random reduction of 3^k-multisack to the following problem: "Given $a_1 \cdots a_n \in \mathbb{Z}$, p prime, a of order (exactly) 3^k modulo p; does a subset S of $a_1 \cdots a_n$ satisfy $\prod_S a^{a_i} \equiv 1$ modulo p?" As $\prod_S a^{a_i} = a^{(\sum_S a_i)}$, it is clear that if we can find p, a as specified, then

$$\exists \lambda \in \mathbb{Z}\ \exists S \subseteq \{a_1 \cdots a_n\}\ \ S \neq \emptyset,\ \ \sum_S a_i = \lambda \cdot 3^k$$

$$\Leftrightarrow$$

$$\exists S \subseteq \{a_1 \cdots a_n\}\ \ S \neq \emptyset,\ \ \prod_S a^{a_i} \equiv 1 \text{ modulo } p .$$

Our algorithm generating p, a will be justified by the following:

<u>Proposition</u> (Brillhart, Lehmer, and Selfridge [9]). Let $N = F_1R_1 + 1$, $2|F_1$, $(F_1,R_1) = 1$, $R_1 = 2F_1s + r$, $1 \leqslant r < 2F_1$. Assume that (i) $N < (F_1+1)(2F_1^2 + (r-1)F_1 + 1)$,

(ii) for each prime $p_i | F_1$ $\exists a_i$: $a_i^{N-1} \equiv 1$ modulo N, $(a_i^{(N-1)/p_i} - 1, N) = 1$.

Then N is prime if and only if $s = 0$ or $r^2 - 8s$ is not a square.

Consider the following nondeterministic algorithm computing values of p, a:

```
(B) "On input $3^k$, guess $0 \leq \ell < 2k$ and $1 \leq f$ such that
    $(3^\ell \cdot f) < (3^k)^2$, $3 \nmid f$. Set
        $p = 1 + 2f3^{\ell+k}$,
        $F_1 = 2 \cdot 3^{\ell+k}$, $R_1 = f = 2F_1 s + r$, $1 \leq r < 2F$,
        if $s = 0$ and $r^2 - 8s$ is a square, halt without output;
        guess $1 < g < p$; verify that
            $g^{p-1} \equiv 1$ modulo p
            $(g^{(p-1)/2} - 1, p) = (g^{(p-1)/3} - 1, p) = 1$
        Set $a = g^{2f3^\ell}$ modulo p, $0 < a < p$.
        Output p, a."
```

Correctness and running time of (B): It is clear by inspection that we can implement the algorithm (B) in nondeterministic polynomial time. It will follow that any output p, a has p a prime once we verify condition (i) of the Proposition. We have indeed

$$p = 1 + 2f \cdot 3^{k+\ell} < 1 + 2 \cdot 3^{3k} < 16 \cdot 3^{3k} < (2 \cdot 3^k + 1)(8 \cdot 3^{2k} - 2 \cdot 3^k + 1)$$

where the last term is the minimum possible value (for $r = 0$, $\ell = 0$) of the bound in (i) of the Proposition. So any p output is prime.

Let o( ) represent the order modulo p. By $g^{p-1} \equiv 1 \bmod p$, $o(g) \mid p-1$. By $(g^{(p-1)/3} - 1, p) = 1$, $o(g) \nmid 3^{k+\ell-1}$, so $o(g) = h \cdot 3^{k+\ell}$, $3 \nmid h$. As $3 \nmid 2f$, $o(a) = o(g^{2f3^\ell}) = 3^k$, as desired.

To determine the density of successful paths of (B), we note that every prime congruent to 1 modulo 3^k and less than 3^{3k} is of the form

$$1 + 2 \cdot 3^k \cdot 3^\ell \cdot f\ , \quad 3^\ell f < (3^k)^2\ .$$

Moreover, for any such prime p, "almost all" g with $1 < g < p$

(namely $f \cdot 2 \cdot 3^{k+\ell-1} = (p-1)/3$ out of $p-2$) will have $2 \cdot 3^{k+\ell} \mid o(g)$, and hence satisfy all the Fermat conditions, that is, for any such prime, 1/3 of all paths will be successful. As there is a 1-1 correspondence between p and $\langle f, 3^{\ell} \rangle$, each number will be guessed exactly once as a candidate for p.

It follows that the successful paths will have at least 1/3 of the weight given by the density bound of Lemma 2.3.2; thus (B) is not strictly random according to our definition, but by polynomial repeated application as analyzed in the proof of Theorem 2.3.1, an algorithm with weight > 1/2 of successful paths will be obtained. □

To complete the reductions of Theorem 2.4.2, one must now compute h_i prime, $h_i \equiv a^{a_i}$ modulo p, for each i,

$$c = \Pi h_i$$

and output $\langle p, c \rangle$. The analysis of this nondeterministic random computation (without ERH merely unfaithful) is entirely analogous to earlier proofs and will be omitted here. We merely note that the difficulty in the application of Lemma 2.3.2, that $D = p$ must exceed a constant minimum value, is here resolved by choosing $3^k > 3^{n_0}$ for sufficiently large n_0, in the reduction to 3^k-multisack. As before, this does not affect the asymptotic behavior of our reduction. □

References

[1] Adleman, L. and Manders, K., Reducibility, Randomness and Intractability, Proc. 9th Annual ACM Symp. on Theory of Computing, 1977, pp. 151-163.

[2] Adleman, L., Manders, K., and Miller, G.L., On Taking Roots in Finite Fields, Proc. 18th Annual Symp. on Foundations of Computer Science (IEEE), 1977, pp. 175-178.

[3] Ankeny, N., The Least Quadratic Nonresidue, Ann. of Math. 55 (1952), pp. 65-72.

[4] Baker, A., Contributions to the Theory of Diophantine Equations. I. On the Representation of Integers by Binary Forms, Philos. Trans. Roy. Soc. London Ser. A 263 (1967/68), pp. 173-191.

[5] ________, Transcendental Number Theory, Cambridge University Press, 1975.

[6] Barban, M., Linnik, Yu., and Tshudakov, N., On Prime Numbers in an Arithmetic Progression with a Prime-Power Difference, Acta Arithmetica 9 (1964), p. 375-390.

[7] Blum, M., A Machine-Independent Theory of the Complexity of Recursive Functions, J. ACM 14 (1967), pp. 322-336.

[8] Borevich, Z. and Shafarevich, I., Number Theory, Academic Press, 1966.

[9] Brillhart, J., Lehmer, D.H., and Selfridge, J., New Primality Criteria and Factorisations of $2^m \pm 1$, Math. Comp. 29 (1975), pp. 620-647.

[10] Cook, S., The Complexity of Theorem-Proving Procedures, Proc. 3rd Annual ACM Symp. on Theory of Computing, 1971, pp. 151-158.

[11] Erdös, P., On Pseudoprimes and Carmichael Numbers, Publ. Math. Debrecen 4 (1956), pp. 201-206.

[12] Gauss, K., Disquisitiones Arithmeticae. English Translation: Yale University Press, 1966.

[13] Hartmanis, J., and Hopcroft, J., An Overview of the Theory of Computational Complexity, JACM 18 (1971), pp. 444-475.

[14] Hilbert, D., Mathematische Probleme: Vortrag gehalten auf dem internationalen Mathematiker-Kongress in Paris, 1900. Nachrichten Akad. Wiss. Göttingen, Math.-Phys. Kl. (1900), pp. 253-297.

[15] Karp, R., Reducibility Among Combinatorial Problems. In: Complexity of Computer Computation, eds. Miller, R.N. and Thatcher, J., Plenum Press, 1972, pp. 85-104.

[16] Kung, H., private communication.

[17] Ladner, R., On the Structure of Polynomial Time Reducibility, J. ACM 22 (1975), pp. 155-

[18] Lagurias, J., to appear.

[19] Lehmer, D.H., Computer Technology Applied to the Theory of Numbers. Studies in Number Theory, M.A.A., 1969, pp. 117-151.

[20] Levin, P.A., Universal Sorting Problems (Russian). Problemi Peredaci Informacii IX (1973), pp. 115-116.

[21] Manders, K., and Adleman, L., NP-complete Decision Problems for Binary Quadratics, J. Comp. Sys. Sci. 15 (1978).

[22] Matijasevic, Y., Enumerable Sets are Diophantine (Russian), Dokl. Akad. Nauk SSSR 191 (1970), pp. 279-282.

[23] Matijasevic, Y. and Robinson, J., Reduction of an Arbitrary Diophantine Equation to One in 13 Unknowns, Acta Arithmetica 27 (1975), pp. 521-553.

[24] Matijasevic, Y., to appear.

[25] Miller, G.L., Riemann's Hypothesis and Tests for Primality, J. Comp. Sys. Sci. 13 (1976), pp. 300-317.

[26] Minsky, M., Computation: Finite and Infinite Machines, Prentice-Hall, 1967.

[27] Mordell, L., Diophantine Equations, Pure and Applied Mathematics, Vol. 30, Academic Press, 1969.

[28] Niven, I., and Zuckerman, H., An Introduction to the Theory of Numbers, J. Wiley, 1960, 1966.

[29] Plaisted, D., Some Polynomial and Integer Divisibility Problems are NP-hard, Proc. 17th Annual Symp. on Foundations of Computer Science (IEEE), 1976, pp. 264-267.

[30] Pratt, V., Every Prime has a Succinct Certificate, SIAM J. Comput. 4 (1975), pp. 214-220.

[31] Turing, A., On Computable Numbers, with an Application to the Entscheidungsproblem; Proc. London Math. Soc. Ser. 2 42 (1936), pp. 230-265.