CSC 2421: Approximation Algorithms	**Winter 2003**

Lecture 10: March 18th

Lecturer: Charles Rackoff *Scribes: Travis Gagie and Al-Mustansir Bin Mukhles*

Today's lecture will cover:

1. the Merkle-Hellman public-key cryptosystem;
2. simple hardness of approximation results, with examples CHROMATIC-NUMBER and TSP;
3. MAX-3-SAT;
4. an introduction to probabilistically checkable proofs;
5. the PCP Theorem; and
6. gap-introducing reductions to MAX-3-SAT.

Further material can be found in the text and online at:
`http://www.nada.kth.se/~viggo/problemlist/compendium.html`.

1 The Merkle-Hellman Public-Key Cryptosystem

In the last lecture, we saw that, in the average case, the L^3 algorithm can be used to solve low-density instances of SUBSET-SUM. In this section, we describe a simplified version of a public-key cryptosystem proposed by Merkle and Hellman in 1978 [6] that is based on KNAPSACK. It was originally broken by Shamir [7] in 1982. In 1983, Adleman [1] realized that it can be broken using the L^3 algorithm. There have been numerous alleged improvements to the scheme proposed, almost all of which have also been broken.

To use the Merkle-Hellman scheme, we first generate a public key and a private key. To do this, we:

1. choose n random $2n$-bit numbers $s_1, \ldots, s_n$, with $(2^i - 1)2^n \leq s_i < 2^i 2^n$;
2. choose a random m, with $2^{2n+1} < m < 2^{2n+2}$;
3. choose a random $w < m$, with w relatively prime to m; and
4. compute $a_1, \ldots, a_n$, where $a_i = (s_i \cdot w) \bmod m$.

Our public key is $\langle a_1, \ldots, a_n \rangle$. Our private key is $\langle s_1, \ldots, s_n, m, w \rangle$.

Notice that $s_1, \ldots, s_n$ is super-increasing. That is, $s_i > \sum_{j=1}^{i-1} s_j$. Also, notice that $m > \sum_{i=1}^{n} s_i$.

Suppose that we are given $s_1, \ldots, s_n$ and $\sum_{i=1}^{n} x_i \cdot s_i$, where $x_1, \ldots, x_n$ is an unknown binary string. Since $s_1, \ldots, s_n$ is super-increasing, we can efficiently compute $x_1, \ldots, x_n$, using the following algorithm:

$t \leftarrow \sum_{i=1}^{n} x_i s_i$
for $i = n \ldots 1$ **do**
  if $t \geq s_i$ **then**
    $x_i = 1$
    $t \leftarrow t - s_i$
  else
    $x_i = 0$
  end if
end for

1.1 Encryption

Suppose that someone wants to send us an encrypted message. To encrypt an n-bit string $x_1, \ldots, x_n$ using our public key $\langle a_1, \ldots, a_n \rangle$, that person computes $e = \sum_{i=1}^{n} x_i \cdot a_i$.

In practice, this scheme would not be used to encrypt long messages. Instead, it would be used to encrypt session keys to be used with a private-key cryptosystem. Thus, it is reasonable to consider encrypting strings of only n bits.

1.2 Decryption

To decrypt e using our private key $\langle s_1, \ldots, s_n, m, w\rangle$, we compute

$$\begin{aligned}
& \left(\frac{e}{w}\right) \bmod m \\
= \;& \left(\frac{\sum_{i=1}^{n} x_i \cdot a_i}{w}\right) \bmod m \\
= \;& \left(\frac{\sum_{i=1}^{n} x_i((s_i \cdot w) \bmod m)}{w}\right) \bmod m \\
= \;& \left(\sum_{i=1}^{n} x_i \cdot s_i\right) \bmod m.
\end{aligned}$$

Since

$$\sum_{i=1}^{n} x_i \cdot s_i \leq \sum_{i=1}^{n} s_i < m,$$

we have that

$$\left(\frac{e}{w}\right) \bmod m = \left(\sum_{i=1}^{n} x_i \cdot s_i\right) \bmod m = \sum_{i=1}^{n} x_i \cdot s_i.$$

As noted above, we can efficiently compute $x_1, \ldots, x_n$ from $s_1, \ldots, s_n$ and $\sum_{i=1}^{n} x_i \cdot s_i$.

1.3 Motivation

The motivation behind this scheme is to "disguise" a set of numbers, $s_1, \ldots, s_n$, with respect to which SUBSET-SUM is easy, as a set of numbers, $a_1, \ldots, a_n$, with respect to which SUBSET-SUM is (supposedly) hard. This is done by multiplying each s_i by the randomly chosen number w, so that the result, a_i, "looks" like a randomly chosen number mod$m$. As noted at the beginning of this section, such attempts have not been successful. We mention this scheme since breaking it is another application of the L^3 algorithm.

2 Simple Hardness of Approximation Results

In this section, we define gap-introducing reductions. These can be used to prove hardness of approximation under certain assumptions, e.g, that P $\neq$ NP. Then, we prove several simple hardness results. Finally, we define Max-3-Sat, as an introduction to the more complicated proof techniques we will discuss in the last two sections.

By our convention, approximation ratios are greater than 1. Thus, for upper bounds, closer to 1 is smaller and better and, for lower bounds, further from 1 is larger and better. We only consider approximation algorithms that are guaranteed to return a feasible answer. This means that, for minimization problems, our estimate will be greater than or equal to the optimum and, for maximization problems, our estimate will be less than or equal to the optimum.

Suppose that L is an NP-complete language and Π is a minimization problem with objective function f. Suppose that there exists a polytime-computable function g that maps yes-instances of L to S_1 and maps no-instances of L to S_2, where S_1 and S_2 are subsets of instances of Π. Suppose that there exists a polytime-computable function h such that, for every yes-instance x of L, we have $f(g(x) \leq h(g(x))$ but, for every no-instance x of L, we have that $f(g(x)) > \alpha \cdot h(g(x))$. This is illustrated in Figure 1.

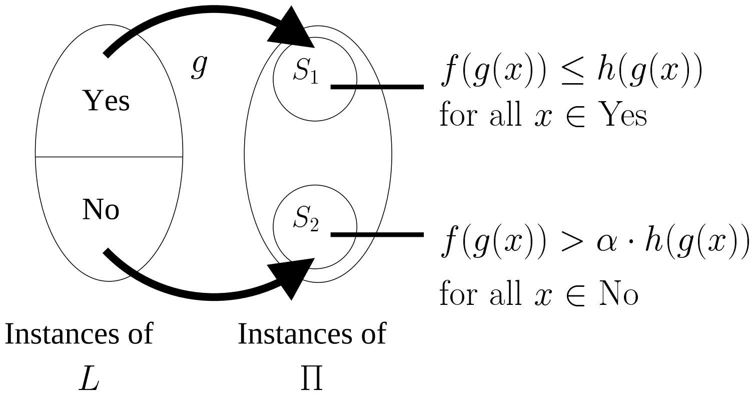


Figure 1: A gap-introducing reduction from L to Π.

Theorem 1. *If there exists a polytime algorithm that approximates* Π *to within a factor of* α*, then* $\mathrm{P} = \mathrm{NP}$.

Proof. If there exists a polytime algorithm that approximates Π to within a factor of α, then there exists a polytime algorithm for determining membership in L:

1. for an instance x of L, estimate $f(g(x))$;
2. accept if and only if the estimate is less than or equal to $\alpha \cdot h(g(x))$.

Thus, P = NP. □

The function g is called a *gap-introducing reduction* from L to Π. In some sense, α is the "size of the gap" introduced by the reduction g.[1] To make this idea more concrete, we present a simple example.

The *chromatic number* of a graph G is the least integer c such that the vertices G can be coloured with c colours such that no two adjacent vertices assigned the same colour.

Theorem 2. *If* P $\neq$ NP, *then no polytime algorithm approximates* Chromatic-Number *to within a factor strictly less than* $4/3$.

Proof. This theorem follows from the fact that 3-Colourability is NP-complete and the existence of the gap-introducing reduction shown in Figure 2. □

There is a similarly trivial gap-introducing reduction from HamCycle to TSP. Given a graph G, we construct a graph G' as follows:

1. let G' be the complete graph on the same vertices as G;
2. assign each edge of G' that is in G cost 0; and
3. assign each edge of G' that is not in G cost 1.

Notice that, if G is Hamiltonian, then the minimum cost of a traveling salesperson's tour of G' is 0. Otherwise, it is at least 1. Thus, no polytime algorithm approximates TSP to within a finite factor, unless P = NP.

[1] In fact, α can be a function. We have omitted consideration of several cases, e.g., when Π is a maximization problem instead of a minimization problem. The text contains a more rigourous exposition; we have only introduced the flavour.

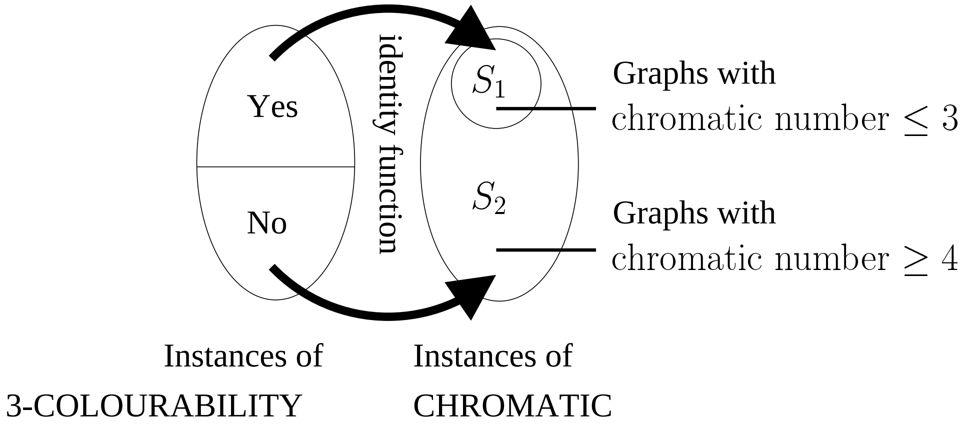


Figure 2: A gap-introducing reduction from 3-Colourability to Chromatic-Number.

3 Max-3-Sat

Max-3-Sat is defined as the following maximization problem: given a set[2] of m 3-clauses[3], determine how many of those clauses can be simultaneously satisfied.[4] The corresponding search problem is to find a truth assignment satisfying as many of the 3-clauses as possible.

3.1 Upper Bounds

It is interesting to note that the general optimization problem Max-Sat does not behave like the decision problem Sat. For example, Max-2-Sat is NP-hard to solve optimally even though 2-Sat is known to be in P.

When all of the 3-clauses contain exactly three distinct literals, for any constant $\epsilon > 0$, there is a simple randomized algorithm that simultaneously satisfies $\left(\frac{7}{8} - \epsilon\right) m$ of the 3-clauses with high probability.

This algorithm is based on the fact that, since each 3-clause contains exactly three distinct literals, each of the clauses is satisfied by at least $\frac{7}{8}$ of the possible truth assignments. Thus,

[2]An instance of Max-3-Sat can be viewed as either a set of clauses or as a 3-CNF formula.

[3]Disjunctions containing at most three literals.

[4]Satisfied by the same truth assignment.

by randomly sampling polynomially many truth assignments, with high probability we satisfy at least $\left(\frac{7}{8} - \epsilon\right) m$ of the 3-clauses. This algorithm can be derandomized using the method of conditional expectation, as described in Chapter 16 of the text. This also improves the bound by removing the $-\epsilon$.

Karloff and Zwick [5] give a randomized semi-definite programming algorithm for the search problem corresponding to Max-3-Sat. They prove that the expected number of clauses simultaneously satisfied is at least $\frac{7}{8}$ of optimal when all of the 3-clauses can be simultaneously satisfied. They also conjecture that their algorithm performs this well on arbitrary instances of the search problem for Max-3-Sat. There is a remark on Zwick's webpage that this conjecture has been confirmed[5], at:

`http://www.cs.tau.ac.il/~zwick/my-online-papers.html#approximation`.

3.2 Lower Bounds

There is a trivial gap-introducing reduction from 3-Sat to Max-3-Sat: a 3-CNF formula with m clauses is satisfiable if and only if more than a $\frac{m-1}{m}$ fraction of its clauses can be simultaneously satisfied.

This implies that, if $\mathrm{P} \neq \mathrm{NP}$, then no polytime algorithm can approximate Max-3-Sat to within a factor strictly less than $\frac{m}{m-1}$. We want to prove a stronger result, specifically:

If $\mathrm{P} \neq \mathrm{NP}$*, then, for some constant* $\epsilon > 0$*, no polytime algorithm can approximate* Max-3-Sat *to within a factor of* $1 + \epsilon$.

The claim above is in fact true. Håstad [4] has proven that no polytime algorithm can approximate Max-3-Sat to within a factor of $\frac{8}{7} - \epsilon$, for any $\epsilon > 0$. By the above algorithm, this bound is nearly optimal.

Finding gap-introducing reductions that introduce constant gaps is an obvious approach to proving the desired hardness result for Max-3-Sat.

Suppose that, for each $L \in \mathrm{NP}$, there exists a polytime-computable function g from instances of L to instances of Max-3-Sat such that:

- $y \in L$ implies that all of the 3-clauses in $g(y)$ are simultaneously satisfiable; and

[5] Possibly this refers to a later paper by Zwick [8].

- $y \notin L$ implies that fewer than 99% of the 3-clauses in $g(y)$ are simultaneously satisfiable.

This implies a lower bound of $\frac{100}{99}$ on the hardness of approximability of MAX-3-SAT, unless P = NP. More interestingly, it implies the existence of a miraculous proof system for each language in NP.

There is a standard proof system for $L \in$ NP: a deterministic polytime verifier takes as input an instance y and an alleged proof π that $y \in L$, reads all of π, and accepts if and only if π is a valid proof.

We now give an alternate proof system for $L \in$ NP. Define a proof that $y \in L$ to be a truth assignment satisfying $g(y)$, where g is the gap-introducing reduction from L to MAX-3-SAT. Define a probabilistic polytime verifier V to:

- take as input an instance y and an alleged proof π that $y \in L$; and
- accept if and only if π satisfies a 3-clause chosen uniformly at random.

Notice that V only looks at 3 bits of π. Also,

- if $y \in L$, then the 3-clauses in $g(y)$ are all simultaneously satisfiable, so there exists a proof π such that $V(y, \pi)$ accepts with probability 1; and
- if $y \notin L$, then, for every truth assignment π, π satisfies fewer than 99% of the clauses in $g(y)$, so $V(y, \pi)$ accepts with probability less than 0.99.

The probability that V accepts a yes-instance is called the *completeness* and the probability that V accepts a no-instance is called the *soundness*.[6]

Somewhat surprisingly, under the assumptions that we have made, it is possible to achieve a constant separation between completeness and soundness while only checking 3 bits of the proof. We can increase this separation by replacing V with a verifier V' that repeatedly simulates V and accepts if and only if it every simulation accepts.

For example, if V' simulates V 100 times, then:

- if $y \in L$, then there exists a proof π such that $V'(y, \pi)$ accepts with probability 1; and

[6]This is counter-intuitive. Increased soundness means that the verifier is *more* likely to make a mistake.

- if $y \notin L$, then, for every proof π, $V'(y, \pi)$ accepts with probability less than $(0.99)^{100} \approx \frac{1}{e} < \frac{1}{2}$.

Notice that V' checks at most 100 times more bits than V, that is, 300 bits. Also, V' makes 100 times more random choices of bits of the proof than V does.

So we see that a strong kind of gap-producing reduction as hypothesized above yields a miraculous kind of proof system for NP languages. We will now see that these proof systems actually exist, and that they can be used to achieve the desired reductions.

4 Probabilistically Checkable Proofs

A language L has probabilistically checkable proofs (PCP's) if it is always possible to probabilistically check a proof of membership in L by reading a bounded amount of the proof.

More formally, the class

$$\mathrm{PCP}_{c,s}\left(r(n), b(n)\right)$$

is the class of languages L for which there exists a probabilistic polytime verifier V such that:

- V takes as a "normal" input a string y and runs in time polynomial in $n = |y|$;
- V takes as a "special" input an alleged proof π that $y \in L$;
- V uses $r(n)$ random bits;
- V examines at most $b(n)$ bits of π;
- if $y \in L$, then, for some π, $V(y, \pi)$ accepts with probability at least c; and
- if $y \notin L$, then, for all π, $V(y, \pi)$ accepts with probability less than s.

Here, c is the completeness parameter and s is the soundness parameter.

By convention, there are restrictions on the parameters c, s, $r(n)$, and $b(n)$:

- $0 \leq s < c \leq 1$;

- $r(n)$ and $b(n)$ are at most polynomial in n; and
- proofs are at most $2^{r(n)}$ bits long.

The first restriction, that completeness be greater than soundness, guarantees that a verifier must be more likely to accept a yes-instance than a no-instance. The second restriction is because verifiers are polytime: if $r(n)$ or $b(n)$ were super-polynomial, then a verifier would not have time to read $r(n)$ random bits or $b(n)$ bits of the proof. The third restriction guarantees that a verifier has enough random bits to choose a bit of the proof uniformly at random.

Lemma 3. $\mathrm{PCP}_{c,s}\left(O(\log n), n^{O(1)}\right) \subseteq \mathrm{NP}$.

Proof. Let L be a language in $\mathrm{PCP}_{c,s}\left(O(\log n), n^{O(1)}\right)$ via verifier V. To show that $L \in \mathrm{NP}$, we construct a polytime non-deterministic Turing machine M that accepts L.

Given a string y, M guesses an alleged proof π that $y \in L$. M then simulates $V(y, \pi)$ on all possible choices of random bits. Notice that, since V is polytime and uses $O(\log n)$ random bits, this takes at most polynomial time. If at least a c fraction of these simulations accept, then M accepts. Otherwise, M rejects.

If $y \in L$, then there exists a proof π such that $V(y, \pi)$ accepts with probability at least c. Therefore, at least a c fraction of the simulations accept, so M accepts. If $y \notin L$, then, for every proof π, $V(\alpha, \pi)$ accepts with probability less than $s < c$. Therefore, less than a c fraction of the simulations accept, so M rejects. □

Knowing Lemma 3, it is easy to see that for every $1 \geq c > s \geq 0$

$$\mathrm{NP} = \mathrm{PCP}_{c,s}\left(O(\log n), n^{O(1)}\right).$$

That is, if we can check the whole proof, then we do not need random bits[7] and we can be certain of our decision.

In the next section, we state the PCP Theorem. This says that, for all languages in NP, by increasing the soundness parameter to 1/2 and increasing the number of random bits available to $O(\log n)$, we can restrict the number of bits of the proof we check to $O(1)$.

[7] The third parameter is larger than 0 only because we restricted proofs to be at most exponential in the number of random bits available. Recall that the standard definition of NP allows polynomial-length proofs, so we need to allow $O(\log n)$ random bits.

5 The PCP Theorem

One of the most interesting and least obvious of the results in computer science is the following:

Theorem 4 (PCP Theorem). $\mathrm{NP} = \mathrm{PCP}_{1,1/2}\,(O(\log n), O(1))$.

Note that since every NP language can be Karp reduced to, say SAT, and since SAT$\in$ $\mathrm{PCP}_{1,1/2}(O(\log n), k)$ for some fixed k, we have that for some fixed k, $\mathrm{NP} = \mathrm{PCP}_{1,1/2}\,(O(\log n), k)$.

One direction of Theorem 4, that $\mathrm{PCP}_{1,1/2}\,(O(\log n), O(1)) \subseteq \mathrm{NP}$, follows from Lemma 3.

The other direction, that $\mathrm{NP} \subseteq \mathrm{PCP}_{1,1/2}\,(O(\log n), O(1))$, was proven by Arora, Lund, Motwani, Sudan, and Szegedy [2]. It is an amazing result. It seems deeper than its current applications, and it probably has fundamental implications that have not yet been realized.

Corollary 5. *For any constant s, $1 > s > 0$, $\mathrm{NP} = \mathrm{PCP}_{1,s}\,(O(\log n), O(1))$.*

Proof. Suppose that $L \in \mathrm{NP}$. Then, for some verifier V, $L \in \mathrm{PCP}_{1,1/2}\,(O(\log n), O(1))$ via V.

Let V' be the following verifier: given a string y and an alleged proof π that $y \in L$, V' simulates $-\log_2 s$ executions of $V(y,\pi)$, using new random bits each time, and accepts if all simulations accept.

If $y \in L$, then there exists a proof π such that $V(y,\pi)$ accepts for every choice of random bits. Therefore, $V'(y,\pi)$ simulations all accept, and $V'(y,\pi)$ accepts.

If $y \notin L$, then, for every proof π, $V(y,\pi)$ accepts for fewer than half of the choices of random bits. Therefore, each of $V'(y,\pi)$'s simulations accept with probability less than 1/2. Thus, $V'(y,\pi)$ accepts with probability less than

$$\left(\frac{1}{2}\right)^{-\log_2 s} = s.$$

V' uses $-\log_2 s$ times more random bits than V and examines $-\log_2 s$ times more bits of the proof that V. However, since s is a constant, this does not affect our asymptotic notation. $\square$

6 Gap-Introducing Reductions to Max-3-Sat

We can now use Theorem 4 to finish what we started in Subsection 3.2, that is, proving a hardness of approximation result for Max-3-Sat.

Theorem 6. *If* $\mathrm{P} \neq \mathrm{NP}$*, then, for some constant* $\epsilon > 0$*, no polytime algorithm can approximate* Max-3-Sat *to within a factor of* $1 + \epsilon$*.*

Proof. Let L be an NP-complete language. By Theorem 4, for some constants d and k and some verifier V, $L \in \mathrm{PCP}_{1,1/2}\,(d \log n, k)$ via V.

Suppose that we want to determine membership of a string y in L, with $|y| = n$. Furthermore, suppose that alleged proofs for V on y are of length at most u, where u is polynomial in n.

Let $r_1, \ldots, r_{n^d}$ be the n^d binary strings of length $d \log n$; these are the possible choices that V might make for its random bits. First, we construct n^d boolean formulas, which may not be in 3-CNF, that correspond to $r_1, \ldots, r_{n^d}$. These formulas, together, will contain variables $x_1, \ldots, x_u$, corresponding to the bits of a proof. Then, we transform these formulas into sets of 3-clauses. Finally, we consider the union of these sets.

For any particular choice of random bits, V can be considered to be evaluating a boolean decision tree of height k. This tree can be encoded as a boolean formula of size at most $k2^k$ that contains at most 2^k variables, which are a subset of $\{x_1, \ldots, x_u\}$.

For $1 \leq i \leq n^d$, let F_i be such a boolean formula with the following property: for any proof π, $V(y, \pi)$ accepts with random bits r_i if and only if F_i is satisfied by the truth assignment mapping the bits of π onto $x_1, \ldots, x_u$.

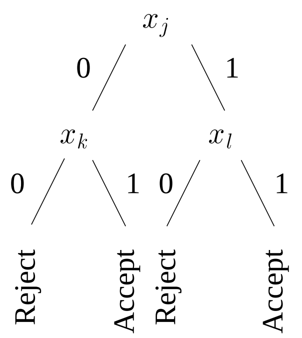
For example, suppose that $k = 2$. Fix a choice of random bits r. Suppose that, given a proof π, $V(y, \pi)$ on r first examines $\pi(j)$, where $\pi(j)$ denotes the jth bit of π. If $\pi(j) = 1$, then V examines $\pi(k)$. If $\pi(k) = 1$, then V accepts. If $\pi(j) = 0$, then V examines $\pi(l)$. If $\pi(l) = 1$, then V accepts. In all other cases, V rejects. This corresponds to the boolean decision tree shown below.

This tree can be encoded as the formula

$$(\neg x_j \wedge x_k) \vee (x_j \wedge x_l).$$

Since k is a constant and V is polytime, each F_i can be computed in polynomial time.[8]

[8]Notice that the F_i's are *not* tableaux for V on y—using the construction illustrated here, they are DNF-



Notice that, if $y \in L$, then there exists a proof π such that $V(y, \pi)$ accepts for every choice of random bits, so all of $F_1, \ldots, F_{n^d}$ can be simultaneously satisfied. If $y \notin L$, then, for all proofs π, $V(y, \pi)$ accepts for fewer than half of $r_1, \ldots, r_{n^d}$, so fewer than half of $F_1, \ldots, F_{n^d}$ can be simultaneously satisfied.

Although the F_i's are boolean formulas, they may not be 3-CNF formulas. We use the standard Karp-reduction [3] from SAT to 3-SAT to obtain sets of 3-clauses $F'_1, \ldots, F'_{n^d}$ such that:

- the number of clauses in F'_i is at most 3 times the size of F_i;
- F'_i contains a superset of the variables in F_i;
- any truth assignment to the variables in F_i that satisfies F_i can be extended to satisfy F'_i;
- any truth assingment that satisfies F'_i satisfies F_i; and
- the variables that we introduce for F'_i are not in $\{x_1, \ldots, x_u\}$ and are not introduced in any other F'_j.

Using this reduction, the number of clauses in F'_i is at most 3 times the size of F_i. Therefore, each of $F'_1, \ldots, F'_{n^d}$ contains at most $3k2^k$ clauses. Consider

$$\mathcal{F} = \bigcup_i F'_i,$$

formulas. Each term in an F_i contains k literals, and represents a branch leading to an accepting leaf in the corresponding boolean decision tree.

which contains at most $n^d 3k2^k$ clauses.

Suppose that $y \in L$. Then there exists a proof π such that $V(y, \pi)$ accepts for every choice of random bits. Thus, there exists a truth assignment that satisfies all of $F_1, \ldots, F_{n^d}$. Furthermore, there exist an extension of that truth assignment that satisfies all of $F'_1, \ldots, F'_{n^d}$, and so satisfies all of $\mathcal{F}$.

Now suppose that $y \notin L$. Then, for all proofs π, $V(y, \pi)$ accepts for fewer than half of the choices of random bits. Thus, every truth assignment satisfies fewer than half of $F_1, \ldots, F_{n^d}$. Furthermore, for every truth assignment τ, for fewer than half of the F'_i's, τ satisfies all of F'_i. Therefore, every truth assignment falsifies more than $n^d/2$ clauses in $\mathcal{F}$, which is at least a

$$\frac{n^d}{2n^d 3k2^k} = \frac{1}{6k2^k}$$

fraction of the clauses in $\mathcal{F}$. Since k is a constant, this factor is also a constant.

Therefore, there exists a constant factor,

$$\frac{1}{1 - \frac{1}{6k2^k}} = 1 + \frac{1}{6k2^k - 1},$$

such that, if there exists a polytime algorithm that approximates Max-3-Sat to within that factor, then there exists a polytime algorithm to decide membership in L. Since L is NP-complete, this would imply that P = NP. $\square$

References

[1] L. M. Adleman. On breaking generalized Knapsack public-key cryptosystems. In *ACM Symposium on Theory of Computation*, pages 402–412, 1983.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *IEEE Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* MIT Press/McGraw-Hill, 2nd edition, 2001.

[4] J. Håstad. Some optimal inapproximability results. In *ACM Symposium on Theory of Computation*, pages 1–10, 1997.

[5] H. Karloff and U. Zwick. A 7/8-approximation algorithm for MAX 3SAT? In *IEEE Symposium on Foundations of Computer Science*, pages 406–415, 1997.

[6] R. Merkle and M. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, 24:525–530, 1978.

[7] A. Shamir. A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem. In *IEEE Symposium on Foundations of Computer Science*, pages 145–152, 1982.

[8] U. Zwick. Computer assisted proofs of optimal approximability results. In *ACM Symposium on Discrete Algorithms*, pages 496–505, 2002.