Optimization of Bootstrapping in Circuits

Fabrice Benhamouda[*] Tancrède Lepoint[†] Claire Mathieu[‡] Hang Zhou[§]

Abstract

In 2009, Gentry proposed the first Fully Homomorphic Encryption (FHE) scheme, an extremely powerful cryptographic primitive that enables to perform computations, i.e., to evaluate circuits, on encrypted data without decrypting them first. This has many applications, in particular in cloud computing.

In all currently known FHE schemes, encryptions are associated to some (non-negative integer) noise level, and at each evaluation of an AND gate, the noise level increases. This is problematic because decryption can only work if the noise level stays below some maximum level L at every gate of the circuit. To ensure that property, it is possible to perform an operation called *bootstrapping* to reduce the noise level. However, bootstrapping is time-consuming and has been identified as a critical operation. This motivates a new problem in discrete optimization, that of choosing where in the circuit to perform bootstrapping operations so as to control the noise level; the goal is to minimize the number of bootstrappings in circuits.

In this paper, we formally define the *bootstrap problem*, we design a polynomial-time L-approximation algorithm using a novel method of rounding of a linear program, and we show a matching hardness result: $(L-\varepsilon)$-inapproximability for any $\varepsilon > 0$.

1 Introduction

Imagine evaluating a circuit with noise: at each gate the noise level may increase due to the computation. Now, imagine that you can occasionally perform a computationally expensive operation on the output of a gate (called *bootstrapping*) to reduce the noise level. Given a circuit, at which gates should you apply the bootstrapping operation to the output of the gate so that the maximum noise level remains within a certain tolerance level? We want to minimize the number of bootstrappings.

For example, if the noise level at an input gate equals 0 and the noise level at a gate with two direct predecessors u and v equals $\max(\text{noiselevel}(u), \text{noiselevel}(v)) + 1$, then in the circuit in Fig. 1, it is possible to maintain a maximum noise level of at most $L = 3$ by doing 2 bootstrappings; that is optimal.

1.1 Motivation This problem arises in cryptography in the context of *fully homomorphic encryption (FHE)* [2,4–6,8,10–15,17,18,26,33]. A fully homomorphic encryption scheme enables one to encrypt bits and keep them confidential, while allowing anyone who is given an encryption $E(a)$ of a bit a and an encryption $E(b)$ of a bit b to publicly compute $E(\mathsf{not}\, a)$, $E(a \,\mathsf{xor}\, b)$, and $E(a \,\mathsf{and}\, b)$. Such a scheme makes it possible to securely compute any binary circuit over encrypted bits. This primitive has tremendous potential for applications, the canonical one being to the problem of outsourcing computation to a remote server without compromising one's privacy. Concrete application examples include biometric identification, statistics over encrypted data [3, 27], machine learning [20], and private genomic analyses [24].

All existing instantiations of FHE follow the same blueprint [13]: ciphertexts (i.e., encryptions of bits) contain some "noise" that grows during the circuit evaluation. To ensure correctness at decryption time, one has to regularly perform *bootstrapping* operations on the ciphertexts whose aim is to lessen the noise.[1] Unfortunately, such operations are very expensive in practice (see, e.g., [8, 12, 16, 22, 28, 31]), hence the question:

What is a minimum set of ciphertexts to be bootstrapped in order to correctly evaluate the circuit?

This is called the *bootstrap problem*.

In all efficient implementations of FHE schemes [4, 5, 9, 12, 17, 21, 22, 26, 31], non-linear gates (and) introduce much more additional noise than linear gates (not and xor), hence we use a simplified model where evaluation of linear gates do not increase the noise; see, e.g., [1, 7, 21, 25, 30].

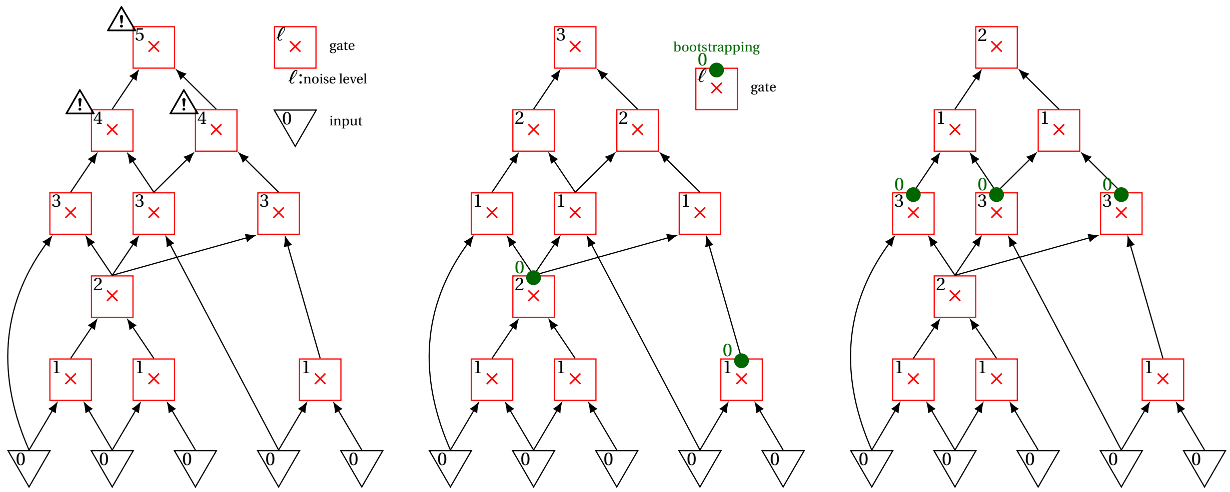
[*]IBM Research. fabrice.benhamouda@ens.fr. Work done while the author was a student at ENS, CNRS, INRIA, and PSL Research University, Paris, France. Research supported in part by the CFM Foundation and the French FUI Project FUI AAP 17 CRYPTOCOMP.

[†]SRI International, New York, USA. tancrede.lepoint@sri.com.

[‡]ENS, CNRS, and PSL Research University, Paris, France. cmathieu@di.ens.fr.

[§]Max-Planck-Institut für Informatik, Saarbrücken, Germany. hzhou@mpi-inf.mpg.de. Research supported in part by the Lise Meitner Award Fellowship.

[1]An upper bound on the admissible noise is given as part of the parameters of the FHE scheme.



(a) Original circuit: without bootstrapping, the noise of the output of the three top gates is above $L = 3$ (warning sign).

(b) Optimal bootstrapping strategy: the noise of the output of all gates is below $L = 3$.

(c) Naive boostrapping strategy: the noise of the output of all gates is below $L = 3$ but the strategy requires 3 bootstrappings (non-optimal).

Figure 1: Example of circuit and of two bootstrapping strategies

Formally, each ciphertext has an associated "noise level" $\ell \in \mathbb{Z}_{\geq 0}$; evaluating a not gate over a ciphertext does not change its noise level; evaluating an xor gate yields a ciphertext whose noise level $\max(\cdot,\cdot)$ is the maximum noise level of its inputs; on the other hand, evaluating a non-linear gates (and) yields a ciphertext with increased noise level $\max(\cdot,\cdot)+1$ (noise behaviors of other FHE schemes are discussed later).

To ensure that the circuit evaluation is correct, the FHE scheme has a parameter $L \geqslant 1$, which is independent of the circuit size, and requires that all ciphertexts must have their noise levels less than or equal to L at all gates of the circuit being evaluated. This requires performing a bootstrapping operation on the output of some gates of the circuit; a bootstrapping operation reduces the noise level of the ciphertext to 0.

The first instantiations of FHE were for $L = 1$ [8, 10, 11, 16]. Most of them merely perform a bootstrapping operation right after (see, e.g., [16]) or right before (see, e.g., [17]) each and gate evaluation. However, this can be computationally wasteful since fewer bootstrappings may be sufficient to evaluate the whole circuit when positioned more carefully [25, 30]; see also Fig. 1 (in this figure, square gates correspond to and gates).

Lepoint and Paillier [25] modeled the problem of constructively computing the exact minimum number of bootstrappings for any $L \geqslant 1$ based on Boolean satisfiability. They associated a Boolean to each ciphertext during the circuit evaluation. The Boolean is equal to $\mathtt{true}$ when the ciphertext should be bootstrapped. Using the logic circuit and the noise level constraints, the authors constructed a Boolean monotone predicate ϕ which captures the correctness of the circuit evaluation. They then described a heuristic method to recover the smallest prime implicant of ϕ, which directly yields the minimum number of bootstrappings and the ciphertexts to be bootstrapped. However no complexity analysis nor hardness result were claimed in [25].[2]

Later, Paindavoine and Vialla [30] showed that, for $L = 1$, the bootstrap problem can be solved in polynomial time by a reduction to (s,t)-min-cut; and that, for $L \geqslant 2$, the bootstrap problem is NP-hard by a reduction from the vertex cover problem.[3] They also provided experimental results on real-world circuits (namely integer addition, integer multiplication, and some cryptographic primitives), based on mixed integer linear programming.[4]

1.2 Problem Formulation

In graph theory terms, the bootstrap problem can be formulated as

[2] Lepoint and Paillier only indicated that for Boolean monotone predicates, finding the size of the smallest prime implicant is known to be #P-complete [19, Section 6].

[3] We note that in the terminology of [30], $l_{max} = L+1$ since the minimum noise level in their model is 1.

[4] Their linear programming relaxation is different from the one in this paper.

follows: the input is a positive integer L and a directed acyclic graph (DAG) $G = (V, E)$ whose vertices all have indegree 0 or 2, with colors on the vertices: vertices of indegree 0 are white and vertices of indegree 2 are either blue or red. G may have parallel edges. A *feasible solution* is a subset $S \subseteq V$ of *marked* vertices such that $\max_{u \in V} \ell(u) \leqslant L$, where the function $\ell(\cdot)$ is computed recursively as follows:[5]

$$\ell(v) = \begin{cases} 0 & \text{if } v \text{ is white,} \\ \max\limits_{(u,v)\in E} \ell(u) \cdot \mathbb{1}_{V\setminus S}(u) & \text{if } v \text{ is blue,} \\ \max\limits_{(u,v)\in E} \ell(u) \cdot \mathbb{1}_{V\setminus S}(u) \; + \; 1 & \text{if } v \text{ is red.} \end{cases}$$

The goal of the bootstrap problem is to find a feasible solution S of minimum cardinality.

If $S = V$, then $\ell(v) \leq 1$ for every vertex v, so this solution is always feasible. If $S = \emptyset$, then $\ell(v)$ is the maximum number of red vertices on any path ending at v, so this solution is feasible if and only if there does not exist a path in G containing $L+1$ red vertices.

In terms of the problem we have been discussing, the DAG is a binary or an arithmetic circuit, the white vertices are the input variables, the blue vertices are the xor (or addition) gates, the red vertices are the and (or multiplication) gates, S is the set of ciphertexts that are bootstrapped during the computation, $\ell(\cdot)$ is the noise level, and L is the maximum allowed noise level.[6]

A feasible solution $S' \subseteq V$ is an *α-approximate solution* (for $\alpha \geqslant 1$) if $|S'| \leqslant \alpha \cdot \mathrm{OPT}$, where OPT denotes the minimum cardinality of a feasible solution.

1.3 Results

We characterize the complexity of the bootstrap problem by providing a polynomial-time L-approximation algorithm (Theorem 1.1) and showing that, assuming the Unique Games Conjecture, L is the best achievable approximation factor (Theorem 1.2).

Theorem 1.1. (approximation algorithm) *Let $L \geqslant 1$ be an integer parameter. There is a deterministic polynomial-time approximation algorithm for the bootstrap problem within approximation factor L.*

The proof of Theorem 1.1 is in Section 2.

Theorem 1.2. (hardness of approximation) *Let $L \geqslant 2$ be an integer parameter. For any $\epsilon > 0$, it is NP-hard to approximate the bootstrap problem within a factor of $L - \epsilon$, assuming the Unique Games Conjecture.*

The proof of Theorem 1.2 is in Section 3.

To design the approximation algorithm used in Theorem 1.2, we first observe that a set of marked vertices is a feasible solution if and only if, for every path $p = v_1 \dots v_k$ that starts and ends at red vertices and that traverses $L + 1$ red vertices (including endpoints), at least one vertex among $v_1, \dots, v_{k-1}$ is marked. Such path p is called an *interesting path*.

Based on this observation, our algorithm starts by solving a linear program relaxation with one constraint for each interesting path and obtains a value $x_v \in [0, 1]$, for every $v \in V$, indicating whether the vertex v should be bootstrapped. The challenging part of the algorithm is the rounding.

In a naive attempt to do the rounding, we define $\delta(u, v)$ as the u-to-v distance in the metric induced by $\{x_v\}$. In order that every interesting path from u to v contains a marked vertex, we choose a value $t \in [0, \delta(u, v)]$ (randomly or according to some rules such as in the *region growing technique* [34]), and then mark a vertex $w \in V$ if and only if $\delta(u, w) \leqslant t \leqslant \delta(u, w) + x_w$. This approach does not yield a good approximation because $\delta(u, v)$ might be very small or even zero (see Fig. 2), as there might exist short non-interesting u-to-v paths. Hence the major difference between the bootstrap problem and classical cut problems (e.g., min-cut, multi-cut, multi-terminal cut): in the bootstrap problem, for each pair of vertices (u, v), we only want to "cut" the interesting u-to-v paths, but the non-interesting u-to-v paths may remain.

Another attempt is to apply *iterative rounding* [23, 34]. However this does not seem to help for the bootstrap problem, mainly because the family of interesting paths is not closed under union, intersection, and difference.

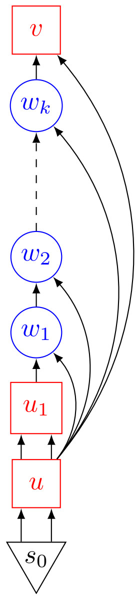
Our approach to rounding is instead to separate paths according to the number of red vertices which they traverse and define a function f_i with respect to all paths that traverses exactly i red vertices. This is a main idea of the algorithm. We perform a rounding for each function separately and obtain L sets of marked vertices. By taking the union of these sets, we obtain a feasible solution of cardinality at most $L \cdot \mathrm{OPT}$.

Remark 1.1. *When $L = 1$, the output of algorithm in Theorem 1.1 is optimal.*

To prove the lower bound of Theorem 1.2, we first look at a related problem called the *DAG Vertex*

[5] The indicator function $\mathbb{1}_{V\setminus S}$ has value 1 on $V \setminus S$ and 0 on S.

[6] Without loss of generality, we assume there are no not gates, since they do not influence the noise level.



Graph with $4 + k$ vertices. $L = 2$.
The only interesting path is: $uu_1w_1w_2 \dots w_kv$.
Circle vertices are blue, square vertices are red, and the triangle vertex is white.

Suppose that the fractional solution obtained from the linear program is the following:

$$\begin{cases} x_{w_i} = \frac{1}{k} & \text{for } i = 1, \dots, k; \\ x_u = x_{u_1} = x_v = 0. \end{cases}$$

Then $\delta(u, v) = 0$.

Figure 2: Example of circuit for which naive rounding does not work

Deletion (DVD) problem [29,32]. In the DVD problem, we are given a directed acyclic graph H and an integer $L \geqslant 2$ and we want to delete the minimum number of vertices so that the resulting graph has no path containing L vertices. Svensson [32] showed that approximating DVD within an $L - \epsilon$ factor is NP-hard, assuming the Unique Games Conjecture.

To show the UG-hardness of approximating the bootstrap problem, we provide an approximation-preserving reduction from the DVD problem to the bootstrap problem.

1.4 Discussion of the Model

Necessity of bootstrappings. To date, the bootstrapping paradigm is the only known way of obtaining an unbounded FHE scheme, i.e., one that can homomorphically evaluate any efficient function using constant-size keys and ciphertexts. Therefore, to exploit the full potential of fully homomorphic encryption, one must resort to bootstrapping.

Noise levels. The bootstrap problem is a simplification of the way in which FHE schemes behave, since in practice noise grows in a more complex manner. Indeed, all encryption procedures in fully homomorphic encryption schemes consist of adding a short noise to the encoding of the message (a bit or more generally an integer). Since the noise is added to the encoding, and computing an xor gate homomorphically essentially corresponds to adding the ciphertexts and thus adding the corresponding noises, on a logarithmic scale the amount of noise remains approximately as large as the maximum input noise (up to one bit). On the other hand, computing an and gate requires a multiplication of the ciphertexts, and makes the noise growth noticeably larger [21]. This is why the cryptographic community introduced the simplified model of Section 1.1 and started building circuits for which the noise does not increase too much in this model [1,7].

In this paper, we say that the noise level of a ciphertext is in $[0, L]$, where L is the parameter of the FHE scheme. In previous works, the level was either in $[1, \ell_{\max}]$ [25,30], or in $[9, \ell_{\max}]$ [21],[7] where $\ell_{\max} = L + 1$. This is equivalent: $\ell = 0$ should not be interpreted to mean that a ciphertext is noise-free, but that the amount of noise the ciphertext contains results from a bootstrapping operation.

Other noise behaviors. The noise model in Section 1.1 corresponds to the family of FHE schemes that are the most efficient in practice, but there exist other families of FHE schemes.

One family corresponds to the first implementations that were proposed [8,11,16]. Therein, non-linear

[7] The 9 comes from the fact that a "fresh" ciphertext (i.e., an unprocessed encryption of a bit) is said to have noise level 1, and that after a bootstrapping, the resulting ciphertext has a noise level 9.

gates behave as $\cdot + \cdot$ for the noise levels.[8] Hence, the noise growth is exponential with the multiplicative depth of the circuit and these schemes will never be used in practice. In addition, to get reasonable parameters, the proof-of-concept implementations set $L = 1$ (compared to, e.g., $L = 41$ in the HElib implementation [22]) and in this particular case, the noise model is actually equivalent to the one we are considering (when $L = 1$).

Another family is the one of the GSW scheme [6, 18]. Variants of this scheme have been implemented [9, 12]. These implementations have a faster wall-clock time for bootstrapping than [8,22], but do not support large plaintext spaces nor vector plaintexts, hence have larger amortized per-bit timing. The noise behavior is slightly different there: it is asymmetric (i.e., the order of the inputs matters). Modeling the noise behavior for these schemes, and extending our results within this new model is left as an interesting open problem.

Computing model. In the bootstrap problem, we minimize the total number of bootstrappings (i.e., marked vertices), thus accounting for classical sequential complexity. We could also consider a parallel computing model, where doing any number of bootstrappings in parallel cost the same as doing one bootstrapping. This might be relevant in some Cloud-based scenarios where the user encrypting the data has an unbounded amount of money and only want to minimize the time to get the result of the circuit evaluation over the encrypted data. However, the financial cost would basically be proportional to the total number of bootstrappings. Furthermore, we remark that this parallel version of the bootstrap problem has a trivial solution: topologically sorting the DAG and greedily marking the vertices with noise level greater than L.

1.5 Other Related Work The DVD problem (see Section 1.3) was introduced by Paik, Reddy, and Sahni in [29] in the context of certain VLSI design and communication problems. Svensson showed that the DVD problem is UG-hard [32]. His work was mainly motivated by the classical *Discrete Time–Cost Tradeoff Problem* in the completely different setting of *Project Scheduling*.

[8] For this model, the lowest noise level needs to be set to 1 instead of 0, and the maximum allowed noise is $L + 1$ instead of L.

2 Approximation Algorithm

To prove Theorem 1.1, we give a randomized algorithm (Algorithm 1) in Section 2.1, analyze it in Section 2.2, and derandomize it in Section 2.3.

2.1 Algorithm For a path $p = v_1 \dots v_k$, the vertex v_k is called the *final vertex* of p and the vertices $v_1, \dots, v_{k-1}$ are called the *non-final vertices* of p.

The following fact is used throughout the paper.

FACT 2.1. *A set of marked vertices is a feasible solution if and only if every path that starts and ends at a red vertex and that contains exactly $L + 1$ red vertices (including endpoints) has a non-final vertex that is marked.*

Proof. We first observe that a set S of marked vertices is a feasible solution if and only if, for all *red* vertices $u \in V$, the noise level $\ell(u)$ with respect to S is at most L. This is because a white vertex has noise level 0 and a blue vertex has noise level not exceeding those of its direct predecessors. We further observe that, for each red vertex $u \in V$, the level $\ell(u)$ is the maximum number of red vertices on *any* v-to-u path (for some red vertex v) that does not contain any marked vertices (except u if appropriate). Thus $\ell(u) \leqslant L$ if and only if every path that starts at a red vertex and ends at u and that contains exactly $L + 1$ red vertices (including endpoints) has a non-final vertex that is marked. Therefore, all *red* vertices $u \in V$ are such that $\ell(u) \leqslant L$ if and only if every path that starts and ends at a red vertex and that contains exactly $L + 1$ red vertices (including endpoints) has a non-final vertex that is marked. □

This fact leads to the definition of *interesting* paths.

DEFINITION 2.1. (INTERESTING PATH) *A path in G is called* interesting *if it starts and ends at red vertices, and traverses exactly $L + 1$ red vertices (including endpoints). For a given vertex $v \in V$ and a given level $i \in \{1, \dots, L + 1\}$, a path in G is called (v, i)-*interesting *if it starts at a red vertex, ends at v, and traverses exactly i red vertices (including endpoints, if appropriate).*

We associate to each vertex $v \in V$ a non-negative weight x_v. In our algorithm, these weights come from a solution of a linear program (LP). These weights induce a metric. More formally, we define the following notion of length.

DEFINITION 2.2. (LENGTH) *Let p be a path in G. We define the* length $\mathsf{len}(p)$ *of p as the sum of* *the weights x_v of all the non-final vertices v of p. We define $f_i(v)$ as the minimum length of a (v,i)-interesting path.*[9]

We remark that, for any red vertex $v \in V$, a $(v, L+1)$-interesting path is an interesting path.

To compute $\{f_i(v)\}_{v,i}$, we use the following dynamic program, which proceeds in phases corresponding to $i = 1, \ldots, L+1$.

- For the base case $i = 1$:

$$f_1(v) = \begin{cases} \infty & \text{if } v \text{ is white,} \\ 0 & \text{if } v \text{ is red,} \\ \min\limits_{u \text{ red}} \delta(u,v) & \text{if } v \text{ is blue.} \end{cases}$$

- For $i \in \{2, \ldots, L+1\}$:

$$f_i(v) = \begin{cases} \infty & \text{if } v \text{ is white,} \\ \min\limits_{(u,v)\in E} (f_{i-1}(u) + x_u) & \text{if } v \text{ is red,} \\ \min\limits_{u \text{ red}} (f_i(u) + \delta(u,v)) & \text{if } v \text{ is blue.} \end{cases}$$

2.2 Analysis We now prove that the output of Algorithm 1 is a feasible solution (correctness property) and has cardinality at most $L \cdot \mathrm{OPT}$ (approximation factor L). We then show that Algorithm 1 runs in polynomial time.

Analysis of Correctness. Consider an interesting path $p = v_1 \ldots v_k$.

LEMMA 2.1. *Let $p = v_1 \ldots v_k$ be an interesting path. For every $j \in \{1, \ldots, k\}$, let $i_j \in \mathbb{N}$ denote the number of red vertices on the subpath $v_1 \ldots v_j$ of p. Then, for any $t \in [0,1]$, there exists $j \in \{1, \ldots, k-1\}$ such that $t \in [f(v_j, i_j), f(v_j, i_j) + x_{v_j}]$.*

Applying Lemma 2.1, and using the fact that $i_j \in \{1, \ldots, L\}$ for every $j \in \{1, \ldots, k-1\}$, we see that the algorithm marks at least one non-final vertex of p, and so by Fact 2.1 the output is a feasible solution, proving correctness.

Proof. (Proof of Lemma 2.1) By definition of interesting paths, the sequence $\{i_j\}_j$ is non-decreasing, $i_1 = 1$, $i_{k-1} = L$, and $i_k = L+1$. It is sufficient to show that the interval $[0,1]$ is contained in the union of the intervals $[f(v_j, i_j), f(v_j, i_j) + x_{v_j}]$ over all $j \in \{1, \ldots, k-1\}$, which is a direct consequence of the three following properties (see Fig. 3):

1. $f(v_1, i_1) = 0$;
2. for every $j \in \{1, \ldots, k-1\}$, $f(v_{j+1}, i_{j+1}) \leqslant f(v_j, i_j) + x_{v_j}$;
3. $f(v_k, i_k) \geqslant 1$.

The first property follows directly from the definition of f since v_1 is red and $i_1 = 1$.

To show the second property, for any $j \in \{1, \ldots, k-1\}$, consider a (v_j, i_j)-interesting path p' that achieves the length $f(v_j, i_j)$. We observe that the concatenation of p' and v_{j+1} is a (v_{j+1}, i_{j+1})-interesting path and it has length $f(v_j, i_j) + x_{v_j}$. From the definition of $f(v_{j+1}, i_{j+1})$, we have $f(v_{j+1}, i_{j+1}) \leqslant f(v_j, i_j) + x_{v_j}$.

To show the third property, consider a (v_k, i_k)-interesting path p' that achieves the length $f(v_k, i_k)$. Then p' is an interesting path since v_k is red and $i_k = L+1$. (p' may differ from p though.) Therefore, the constraint on p' in the LP implies that $\mathsf{len}(p') \geqslant 1$. Hence $f(v_k, i_k) = \mathsf{len}(p') \geqslant 1$.

This concludes the proof. □

Analysis of Quality of Approximation. The expected value of the output is the expected number of marked vertices, $\sum_{v \in V} \Pr(v \text{ marked})$. Let $v \in V$. By the algorithm and a union bound:

$$\begin{aligned} &\Pr(v \text{ marked}) \\ &= \Pr(\exists i \in \{1, \ldots, L\} \ : \ t \in [f_i(v), f_i(v) + x_v]) \\ &\leqslant \sum_i \Pr(t \in [f_i(v), f_i(v) + x_v]). \end{aligned}$$

For t uniformly random in $[0,1]$, the probability that $t \in [f_i(v), f_i(v) + x_v]$ is at most x_v. Thus $\Pr(v \text{ marked}) \leqslant L x_v$ and the expected value of the output is at most $L(\sum_{v \in V} x_v)$. Since the linear program is a relaxation of the problem, $\sum_v x_v$ is less than or equal to the optimum value of the bootstrap problem, proving that the output is an L-approximation.

Analysis of Running Time. Clearly, computing $\{f_i(v)\}_{v,i}$ takes polynomial time. Next we show that the LP in Step 1 of the algorithm can be solved in polynomial time (regardless of an exponential number of constraints). To that end, it is well known (see, e.g., [34]) that a polynomial-time *separation oracle*[10] for this LP suffices.

[9] $f_i(v) := \infty$ if there is no (v,i)-interesting path.

[10] A separation oracle takes as input a supposedly feasible solution to the linear program, and either verifies that it is indeed a feasible solution to the linear program or, if it is infeasible, produces a violated constraint.

Algorithm 1 Approximation algorithm for the bootstrap problem

1: Solve the following LP relaxation, where we have one variable x_v for each vertex v, representing whether there is a bootstrapping on v; and one constraint for each interesting path p.

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v & \\ \text{s.t.} \quad & \sum_{\substack{\text{non-final} \\ \text{vertex } v \text{ of } p}} x_v \geqslant 1 & \forall \text{ interesting path } p \\ & 0 \leq x_v \leq 1 & \forall v \in V \end{aligned}$$

2: For every red vertex u and blue vertex v, compute

$$\delta(u,v) = \min\left\{x_u + x_{v_2} + \cdots + x_{v_{k-1}} : \text{ path } p = uv_2 \ldots v_{k-1}v \text{ such that } v_2, \ldots, v_{k-1} \text{ are blue}\right\},$$

using a classical shortest path algorithm. By convention $\delta(u,v) := \infty$ if no such path exists.

3: For every vertex v and integer $i \in \{1, \ldots, L+1\}$, compute

$$f_i(v) = \min\left\{x_{v_1} + x_{v_2} + \cdots + x_{v_{k-1}} : \text{ path } p = v_1v_2 \ldots v_{k-1}v \text{ is } (v,i)\text{-interesting}\right\}$$

using the side table δ and a dynamic program (see Section 2.1). By convention $f_i(v) := \infty$ if no such path exists.

4: Rounding: Pick a uniformly random value $t \in [0,1]$; A vertex v is marked if and only if there exists $i \in \{1, \ldots, L\}$ s.t. $t \in [f_i(v), f_i(v) + x_v]$.

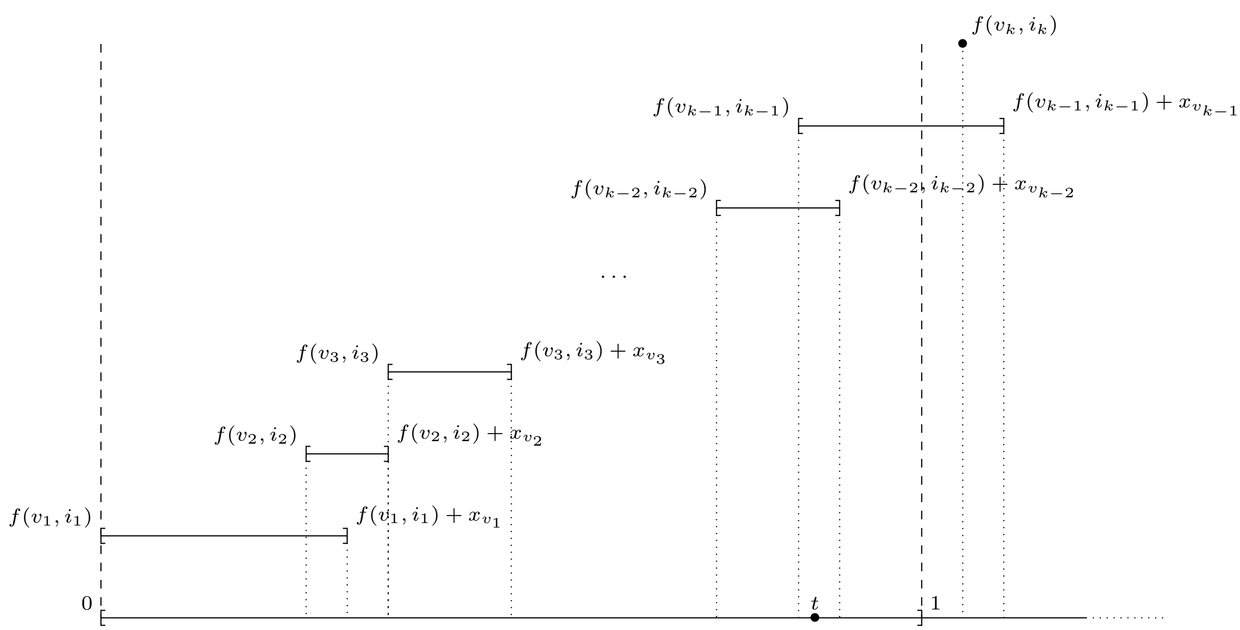


Figure 3: Illustration for the proof of Lemma 2.1

To check whether an oracle input $\{x_v\}_v$ is a feasible solution to the LP, we compute $\{f_i(v)\}_{v,i}$ with respect to $\{x_v\}_v$ using the same dynamic program as before, in polynomial time. From the definition of f, we have that $\{x_v\}_v$ is a feasible solution if and only if $f_{L+1}(v) \geqslant 1$ for every red vertex $v \in V$. Suppose there is some red vertex $v \in V$ with $f_{L+1}(v) < 1$. Then there must be an interesting path p with final vertex v such that the constraint on p is violated. It is easy to enrich the dynamic program in a standard manner, so that we obtain the entire path p. Thus we complete the description of the polynomial-time separation oracle.

Therefore, the overall running time of Algorithm 1 is polynomial.

2.3 Derandomization Algorithm 1 can be easily derandomized: $\{f_i(v)\}_{v,i} \cup \{f_i(v) + x_v\}_{v,i}$ contains at most $2|V| \cdot L$ different values, so they separate the $[0, 1]$ interval into at most $2|V| \cdot L + 1$ sub-intervals. We can enumerate one value t for each sub-interval, compute a feasible solution with respect to each value t, and finally return the best solution among them.

3 Hardness of Approximation

In this section, we prove Theorem 1.2. First, we recall the definition of the DAG Vertex Deletion (DVD) problem (see Section 1.3). In Lemma 3.2, we reduce the DVD problem to the bootstrap problem. The hardness of the bootstrap problem then follows from the hardness of the DVD problem [32].

Lemma 3.1. (Adapted from [32, Theorem 1.1]) *Let $L \geqslant 2$ be an integer parameter. For any $\epsilon > 0$, it is NP-hard to approximate the DVD problem within a factor of $L - \epsilon$, assuming the Unique Games Conjecture.*

Lemma 3.2. *There is an approximation-preserving reduction from the DVD problem to the bootstrap problem.*

Theorem 1.2 follows immediately from Lemmas 3.1 and 3.2. In the rest of the section, we prove Lemma 3.2. The proof is elementary but delicate, mainly because in the bootstrap problem, vertices have indegree at most 2, while in the DVD problem, vertices may have arbitrary indegree.

Consider a DVD instance with the DAG $H = (V_H, E_H)$ and the integer parameter $L \geqslant 2$. As a warm-up, let us first suppose that all vertices in V_H have indegree at most 2. We construct a DAG $G = (V, E)$ for the bootstrap problem:

- We create a new vertex set V'_H (which can be viewed as a clone of V_H): for every $v \in V_H$, V'_H contains a new vertex v'. We also create a new vertex s_0. The vertex set V of G is defined as $V := V_H \cup V'_H \cup \{s_0\}$. The vertices in $V_H \cup V'_H$ are red, and the vertex s_0 is white.
- The edge set E of G consists of all edges of E_H and the following edges: for every vertex $v \in V_H$, 2 copies of the edge (v, v') and $(2 - \text{indegree}(v))$ copies of the edge (s_0, v).

The following lemma implies that there is an approximation-preserving reduction.

Lemma 3.3. *A feasible solution to the DVD problem for the instance (L, H) can be transformed into a feasible bootstrap solution for the instance (L, G) with at most the same cardinality, and vice versa.*

Proof. Let S be a feasible solution to the DVD problem, i.e., every path of L vertices in H contains a vertex in S. We show that S is a feasible bootstrap solution for the instance (L, G). Using Fact 2.1, we only need to show that every interesting path in G contains a non-final vertex that is in S. Let $p = v_1 \dots v_k$ be an interesting path in G. We observe that $v_1, \dots, v_k$ are all red vertices: v_1 is red by the definition of an interesting path, and v_j (for any $2 \leqslant j \leqslant k$) is red since it has positive indegree (and thus cannot be s_0). By the definition of an interesting path, we know that p contains $L+1$ red vertices, so $k = L + 1$. We further observe that every v_i (for any $1 \leqslant i \leqslant k - 1$) is in V_H since v_i has positive outdegree (and thus cannot be in V'_H). Thus $v_1, \dots, v_{k-1}$ form a path of L vertices in H. Since S is a feasible solution to the DVD problem, at least one vertex among $v_1, \dots, v_{k-1}$ is in S. Thus p contains a non-final vertex that is in S.

Conversely, let S be a feasible bootstrap solution. We show that $S \cap V_H$ is a feasible solution to the DVD problem. Let $p = v_1, \dots, v_L$ be a path of L vertices in H. We only need to show that p contains a vertex in $S \cap V_H$. We construct a path p' in G that is the concatenation of p and the vertex v'_L. We remark that p' starts and ends on red vertices and contains exactly $L + 1$ red vertices. Therefore, it is an interesting path. From Fact 2.1, S contains a vertex u that is a non-final vertex of p', i.e., $u \in \{v_1, \dots, v_L\}$, hence u is on p. Since $\{v_1, \dots, v_L\} \subseteq V_H$, $u \in S \cap V_H$. Thus p contains a vertex in $S \cap V_H$. ∎

Let us now deal with the general case where the vertices of H have arbitrary indegrees. First, we initialize the DAG G using the same transformation as before, except that G now contains $(2-\text{indegree}(v))$ copies of an edge (s_0, v) only if the vertex v has indegree at most 2 (rather than for any vertex v). After this transformation, every red vertex in G has indegree at least 2. To transform G into a DAG for the bootstrap problem, we just need to deal with the red vertices with indegree at least 3. Let v be such a vertex. We observe that $v \in V_H$. Let $v_1, \ldots, v_d$ be the direct successors of v in H. We remove from G the edges (v_i, v) (for each i) and add to G new blue vertices $w_1^{(v)}, \ldots, w_d^{(v)}$ and the following edges:

1. two copies of the edge $(v_1, w_1^{(v)})$,
2. for $i = 2, \ldots, d$, an edge $(w_{i-1}^{(v)}, w_i^{(v)})$ and an edge $(v_i, w_i^{(v)})$,
3. two copies of the edge $(w_d^{(v)}, v)$.

The transformation is depicted in Fig. 4. Let $G = (V, E)$ be the final graph. We can verify that (L, G) is an instance of the bootstrap problem.

We show that Lemma 3.3 holds in this general setting. The transformation from a feasible DVD solution to a feasible bootstrap solution is a trivial extension from the previous proof. Let us now focus on the transformation from a feasible bootstrap solution to a feasible DVD solution. The following proposition is the key to the proof.

Proposition 3.1. *Let S be a feasible bootstrap solution for (L, G) which contains a blue vertex $w_i^{(v)}$ for some $v \in V_H$ and some integer i. Then $(S \setminus \{w_i^{(v)}\}) \cup \{v\}$ is also a feasible bootstrap solution for (L, G).*

Proof. Let $S' = (S \setminus \{w_i^{(v)}\}) \cup \{v\}$. From Fact 2.1, we only need to prove that every interesting path contains a non-final vertex that is in S'. Since S is a feasible bootstrap solution, the only non-trivial part is to prove that every interesting path ending in v contains a non-final vertex that is in S'. Let $p = v_1 \ldots v_k$ be such a path, and let $j < k$ be the index of the last non-final red vertex of p (i.e., v_j is a red vertex and there is no red vertex among $v_{j+1}, \ldots, v_{k-1}$). Let p' be the path $v_1 \ldots v_j v'_j$. Since there are exactly L red vertices among $v_1, \ldots, v_L$ and v'_j is red, p' is an interesting path. Thus some non-final vertex u of p' (i.e., $u \in \{v_1, \ldots, v_j\}$) is in S. We observe that p' cannot contain $w_i^{(v)}$ since $w_i^{(v)} \in \{v_{j+1}, \ldots, v_{k-1}\}$. Thus $u \in S'$ and is a non-final vertex of p. ☐

Let S be a feasible bootstrap solution. We construct another feasible bootstrap solution S' with $|S'| \leqslant |S|$ such that S' only contains red and white vertices. As soon as S contains a blue vertex, let it be $w_i^{(v)}$ for some $v \in V_H$ and some integer i, we replace the blue vertex $w_i^{(v)}$ in S by the red vertex v. Let S' be the final S. Then S' contains only red and white vertices and has cardinality at most $|S|$. From Proposition 3.1, S' is a feasible bootstrap solution.

We now show that $S' \cap V_H$ is a solution to the DVD problem using similar arguments as before. Let $p = v_1 \ldots v_L$ be a path of L vertices in H. We only need to show that p contains a vertex in $S' \cap V_H$. We construct a (unique) path p' in G, which starts at v_1, goes through $v_2, \ldots, v_L$, and ends at the red vertex v'_L. We remark that p' starts and ends on red vertices and contains exactly $L+1$ red vertices, namely $\{v_1, \ldots, v_L, v'_L\}$. Therefore, it is an interesting path. From Fact 2.1, S' contains a vertex u that is a non-final vertex of p'. Since S' contains only red and white vertices, we have $u \in \{v_1, \ldots, v_L\}$, hence u is on p. Since $\{v_1, \ldots, v_L\} \subseteq V_H$, $u \in S' \cap V_H$. Thus p contains a vertex in $S' \cap V_H$.

This concludes the proof of Lemma 3.2.

Acknowledgements. We would like to thank Florian Bourse and Pierrick Méaux for their discussion on other FHE models, and the anonymous reviewers for their comments and suggestions.

References

[1] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 430–454. Springer, Heidelberg, April 2015.

[2] Jacob Alperin-Sheriff and Chris Peikert. Practical bootstrapping in quasilinear time. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 1–20. Springer, Heidelberg, August 2013.

[3] Joppe W. Bos, Kristin E. Lauter, and Michael Naehrig. Private predictive analysis on encrypted medical data. *Journal of Biomedical Informatics*, 50:234–243, 2014.

[4] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. *TOCT*, 6(3):13, 2014.

[5] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. *SIAM J. Comput.*, 43(2):831–871, 2014.

[6] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In Moni Naor, editor, *ITCS 2014*, pages 1–12. ACM, January 2014.

[7] Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Pail-

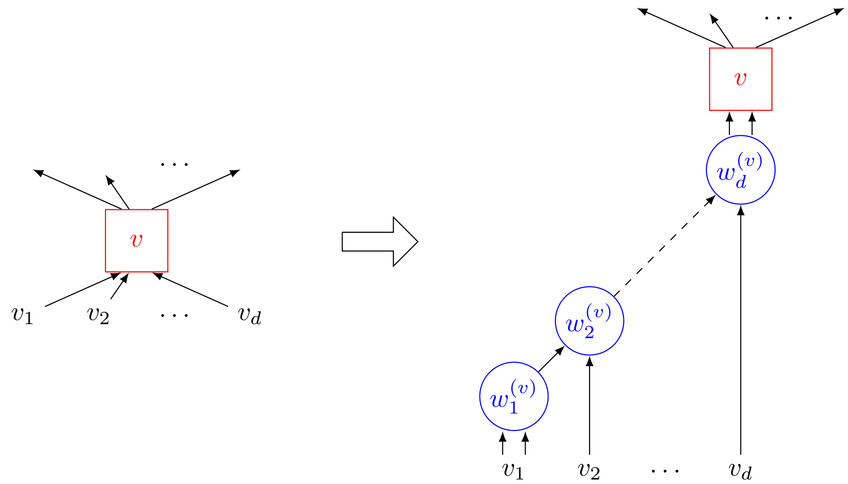


Figure 4: Reducing of the Indegree of a Vertex v (in the Proof of Lemma 3.2). Circle vertices are blue, square vertices are red, $v_1, \ldots, v_d$ are the direct predecessors of v, $w_1^{(v)}, \ldots, w_d^{(v)}$ are new blue vertices.

lier, and Renaud Sirdey. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. In Thomas Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*, pages 313–333. Springer, Heidelberg, March 2016.

[8] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 315–335. Springer, Heidelberg, May 2013.

[9] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *ASIACRYPT*, Lecture Notes in Computer Science. Springer, 2016.

[10] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 487–504. Springer, Heidelberg, August 2011.

[11] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 446–464. Springer, Heidelberg, April 2012.

[12] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 617–640. Springer, Heidelberg, April 2015.

[13] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

[14] Craig Gentry. Toward basing fully homomorphic encryption on worst-case hardness. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 116–137. Springer, Heidelberg, August 2010.

[15] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 107–109. IEEE Computer Society Press, October 2011.

[16] Craig Gentry and Shai Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 129–148. Springer, Heidelberg, May 2011.

[17] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 850–867. Springer, Heidelberg, August 2012.

[18] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.

[19] Judy Goldsmith, Matthias Hagen, and Martin Mundhenk. Complexity of DNF minimization and isomorphism testing for monotone formulas. *Inf. Comput.*, 206(6):760–775, 2008.

[20] Thore Graepel, Kristin Lauter, and Michael Naehrig. ML confidential: Machine learning on encrypted data. In Taekyoung Kwon, Mun-Kyu Lee, and Daesung Kwon, editors, *ICISC 12*, volume 7839 of *LNCS*, pages 1–21. Springer, Heidelberg, November 2013.

[21] Shai Halevi and Victor Shoup. Algorithms in HElib. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2014.

[22] Shai Halevi and Victor Shoup. Bootstrapping for HElib. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 641–670. Springer, Heidelberg, April 2015.

[23] Kamal Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[24] Kristin E. Lauter, Adriana López-Alt, and Michael Naehrig. Private computation on encrypted genomic data. In Diego F. Aranha and Alfred Menezes, editors, *LATINCRYPT 2014*, volume 8895 of *LNCS*, pages 3–27. Springer, Heidelberg, September 2015.

[25] Tancrède Lepoint and Pascal Paillier. On the minimal number of bootstrappings in homomorphic circuits. In Andrew A. Adams, Michael Brenner, and Matthew Smith, editors, *FC 2013 Workshops*, LNCS, pages 189–200. Springer, Heidelberg, April 2013.

[26] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 1219–1234. ACM Press, May 2012.

[27] Michael Naehrig, Kristin E. Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *CCSW*, pages 113–124. ACM, 2011.

[28] Koji Nuida and Kaoru Kurosawa. (Batch) fully homomorphic encryption over integers for non-binary message spaces. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 537–555. Springer, Heidelberg, April 2015.

[29] Doowon Paik, Sudhakar M. Reddy, and Sartaj Sahni. Deleting vertices to bound path length. *IEEE Trans. Computers*, 43(9):1091–1096, 1994.

[30] Marie Paindavoine and Bastien Vialla. Minimizing the number of bootstrappings in fully homomorphic encryption. In *SAC*, volume 9566 of *Lecture Notes in Computer Science*, pages 25–43. Springer, 2015.

[31] Kurt Rohloff and David Bruce Cousins. A scalable implementation of fully homomorphic encryption built on NTRU. In Rainer Böhme, Michael Brenner, Tyler Moore, and Matthew Smith, editors, *FC 2014 Workshops*, volume 8438 of *LNCS*, pages 221–234. Springer, Heidelberg, March 2014.

[32] Ola Svensson. Hardness of vertex deletion and project scheduling. *Theory of Computing*, 9(24):759–781, 2013.

[33] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 24–43. Springer, Heidelberg, May 2010.

[34] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, New York, NY, USA, 1st edition, 2011.