Computing from partial solutions

Anna Gál[*] Shai Halevi[†] Richard J. Lipton[‡] Erez Petrank[§]

November 14, 1997

Abstract

We consider the question: Is finding just a part of a solution easier than finding the full solution? For example, is finding only an ϵ fraction of the bits in a satisfying assignment to a 3-CNF formula easier than computing the whole assignment? For several important problems in $\mathcal{NP}$ we show that obtaining only a small fraction of the solution is as hard as finding the full solution.

This can be interpreted in two ways: On the positive side, it is enough to look for an efficient algorithm that only recovers a small part of the solution, in order to completely solve any of these problems. On the negative side, any partial solution to these problems may be hard to find. For example, unless $NP \subseteq BPP$ there is no efficient algorithm which always finds $\sqrt{n}$ bits of a satisfying assignment of a 3-CNF formula with n variables.

[*]University of Texas at Austin. Part of this work was done while at DIMACS and Princeton University. E-mail: panni@cs.utexas.edu

[†]E-mail: shaih@watson.ibm.com

[‡]Princeton University and Bellcore Research. Supported by NSF CCR-9633103 and AFOSR F49620-97-0190. E-mail:rjl@cs.princeton.edu

[§]DIMACS Center, P.O.Box 1179, Piscataway, NJ 08855. E-mail: erez@dimacs.rutgers.edu.

1 Introduction

Is it easier to find parts of the solution than getting it all? We ask this question for several famous $\mathcal{NP}$ problems, including 3-SAT, Graph Isomorphism, the Shortest Lattice Vector problem, Graph 3-Colorability, Vertex Cover and Clique. For all these problems we show that retrieving parts of the solution is as hard as retrieving it all. This is true even if the algorithm may choose which parts of the solution it is going to retrieve.

This question can be viewed in various ways. First, it has immediate application to the search for efficient algorithms for these problems: Suppose, for example, that one wishes to solve Graph Isomorphism. Our results imply that instead of looking for a method for constructing the whole isomorphism, it is sufficient to find a method that can construct just a small part of it. Namely, we show that if there is an efficient procedure which constructs a small part of the isomorphism, then we can use it to efficiently find the entire solution. We stress that for our construction, it is not enough that the efficient procedure returns an isomorphism between small sub-graphs. Instead it must return such an isomorphism *which is a part of a full isomorphism* between the two graphs.

A second view on this question is as an issue in fault tolerance. Suppose that a solution to a computational problem is sent to us on a channel that omits parts of the messages. Can we recover the full solution if we only receive parts of it? In this work we prove a positive answer for all the problems mentioned above, by specifying omission-correcting algorithms for these problems. We stress that the 'obvious' solution of wrapping the answer by a layer of erasure code does not fit in our framework, since we are interested in the resilience of the computational problem itself. Instead, we use *erasure-resilient reductions*, where in order to tolerate omissions we generate (one or more) different instances of the problem, and use the partial answers to these modified instances to recover the full solution to the original instance. In a sense, these erasure-resilient reductions can be seen as a way to encode an erasure code in a "natural" manner. However, the results which we obtain are surprisingly stronger than just adding an erasure code on top of the solution: We obtain resistance to erasure of nearly all the answer, whereas standard erasure codes (with sub-exponential expansion) can only deal with erasure of some constant fraction.

Another contribution of our work is providing robust proofs of membership. For example, in the 3-SAT case, we present an efficient algorithm which on input ϕ, outputs a new formula ϕ', such that retrieving even a small part of the satisfying assignment to ϕ' allows reconstructing a satisfying assignment to ϕ. This immediately suggests a very robust proof that ϕ has a satisfying assignment: The assignment to ϕ'! Any (large enough) fraction of the satisfying assignment to ϕ' allows efficient construction of a complete satisfying assignment to ϕ. Thus, from small parts of the robust proof, one may reconstruct a full proof for the membership of ϕ in 3-SAT. Furthermore, since 3-SAT is $\mathcal{NP}$-complete and since Cook's reduction [7] is witness preserving, this can be used as a procedure to obtain robust proofs for membership in any $\mathcal{NP}$ language. Note the difference between these robust proofs and PCP proofs such as the ones in [3, 4]. In our case an adversary may choose the bits that we get to see and still we must be able to reconstruct the full witness, whereas the verifier of a PCP proof chooses the bits of the proof that he reads (usually randomly) and only has to verify the validity of the proof.[1]

1.1 Summary of results

Our results can be summarized as follows:

1. **Satisfiability (SAT):** For any $\epsilon > 0$, given a CNF formula over n variables it is possible to construct in probabilistic polynomial time another formula $\phi^{(1)}$ variables (with $|\phi'| = |\phi| + n^{O(1)}$), such that with probability almost 1, given any $N^{\frac{1}{2}+\epsilon}$ bits from a satisfying assignment to ϕ', one can efficiently construct a satisfying assignment to ϕ. See Theorem 1 in Section 2.3.

   We would like to stress the surprising strength of this result: recall that standard erasure codes that have sub-exponential expansion can only deal with erasure of less than half of the bits, whereas in this computational model we can deal with erasure of nearly all the bits.

[1] Note that for the shortest lattice vector, the view of our theorem as a robust proof is not relevant, since we are dealing with a co-$\mathcal{NP}$ type question.

2. **Graph Isomorphism (GI):** There is a (deterministic) polynomial time procedure that on a given pair of graphs (G, H) constructs adaptively $\ell = poly(n)$ oracle queries $(G_1, H_1), \ldots, (G_\ell, H_\ell)$ such that if on each query the oracle answers with a map on only $O(\log n)$ of the vertices (which is a part of an isomorphism between the two graphs in this query), then the procedure finds an isomorphism between G and H. See Theorem 3 in Section 3.

3. **Shortest Lattice Vector (SLV):** Consider the following problem: Given a basis $B_{m\times n}$ for some real lattice $L(B) = \{Bx : x \in Z^n\} \subset R^m$, find the shortest vector (in the L_2-norm) in the lattice. We present an efficient probabilistic procedure to construct a basis $B'_{M\times n}$ for a lattice $L(B')$ such that, with probability almost 1, given any $\lfloor M^\epsilon \rfloor$ entries in the shortest non-zero vector in $L(B')$ we are able to find a shortest vector in the original lattice $L(B)$ in time polynomial in m. The constant ϵ is any constant $\epsilon > 0$ and $M = \max\{2m, \lceil m^{1/\epsilon} \rceil\}$. See Theorem 4 in Section 4.

4. **Other $\mathcal{NP}$-complete problems:** Using our results for SAT we can prove the following type of result for several other $\mathcal{NP}$-complete problems, including Graph 3-Colorability, Vertex Cover and Clique (and Independent Set).

   Suppose that we have access to an oracle for a given problem L. Given an instance X of size n of the problem L, we can efficiently construct a probabilistic query X' of size $N = n^{O(1)}$ to the oracle for L such that, with probability almost 1, from any (large enough) polynomial fraction of a witness for $X' \in L$ one can efficiently construct a witness for $X \in L$. The exact polynomial fraction that we need differ between the problems. For the Clique problem (and Independent Set) it is sufficient to get a $(1/N^\epsilon)$-fraction of the answer for any $0 < \epsilon < 1/2$. For Graph 3-Colorability and Vertex Cover, we need to get a $(1/N^\epsilon)$-fraction of the answer for $0 < \epsilon < 1/8$.

   The proof of these results is based on the structure of the reductions from 3-SAT to these problems. It seems that many $\mathcal{NP}$-complete problems have reductions with such structure, thus our results may apply to other problems. In this extended abstract we illustrate our methods by a formal proof of the above result for Graph 3-Colorability. See Theorem 5 in Section 5.

   The same approach can also be applied to the Hamiltonian-path and Tripartite-matching problems, but the parameters of the reduction are worse than for the other problems (namely, we can only tolerate erasure of some small constant fraction of the answer). In this extended abstract we do not discuss these problems.

   We note that while there are many problems for which we can prove that only some fraction of the solutions are needed, we yet do NOT have a general method for proving such results, and instead need to work on a problem-by-problem basis. In particular, there does not yet seem to be a single master theorem that would imply all these results.

1.2 Further motivation

Cryptographic applications. An important question in cryptography is the relation between obtaining partial information about a secret and computing the whole secret. Indeed, this is the question that underlies the notion of hard-core bits of cryptographic functions (see, e.g., [6, 5, 9]). Our results are in the same spirit, thus providing some more insight into this interesting question.

Structural complexity. An important theme in structural complexity is to study the power of different kinds of oracles. Essentially, one can view our results as shedding light on the power of oracles that only supply part of the answers.

1.3 Organization

In Section 2 we deal with the 3-SAT problem, in Section 3 we present our results on the Graph Isomorphism problem, in Section 4 we discuss the Shortest Lattice Vector problem and in Section 5 we illustrate how to get similar results for certain $\mathcal{NP}$-complete problems by a proof for Graph 3-Colorability.

2 Satisfying assignments to CNF formulae

Below we prove that any CNF formula ϕ can be encoded in another formula ϕ', in such a way that given even a small fraction of the bits in a satisfying assignment to ϕ', it is possible to recover a full satisfying assignment to ϕ. Formally,

Theorem 1 *: For any $\epsilon > 0$, there exist an efficient probabilistic algorithm $\textsc{Enc}_{sat}$, and an efficient deterministic algorithm $\textsc{Rec}_{sat}$ such that*

1. *If ϕ is a CNF formula over n variables, then $\phi' = \textsc{Enc}_{sat}(\phi)$ is a CNF formula over $N = n^{O(1)}$ variables, with $|\phi'| = |\phi| + n^{O(1)}$.*
2. *With probability $1 - 2^{-n}$ (over the coins of $\textsc{Enc}_{sat}$), the formula ϕ' has the following property: If s' is any assignment to $N^{\frac{1}{2}+\epsilon}$ of the variables in ϕ' which can be extended to a full satisfying assignment, then $\textsc{Rec}_{sat}(\phi, \phi', s')$ is a satisfying assignment for ϕ.*

For some applications, it might be useful to have a deterministic transformation $\textsc{Enc}_{sat}$. In this case we can show the following weaker result.

Theorem 2 *: For any $\epsilon > 0$, there exist efficient deterministic algorithms $\textsc{Enc}_{sat}, \textsc{Rec}_{sat}$ such that*

1. *If ϕ is a CNF formula over n variables, then $\phi' = \textsc{Enc}_{sat}(\phi)$ is a CNF formula over $N = n^{O(1)}$ variables, with $|\phi'| = |\phi| + n^{O(1)}$.*
2. *If s' is any assignment to $(\frac{1}{2} + \epsilon)N$ of the variables in ϕ' which can be extended to a full satisfying assignment, then $\textsc{Rec}_{sat}(\phi, \phi', s')$ is a satisfying assignment for ϕ.*

(We note that it is possible to extend Theorem 2 to $\epsilon = o(1)$. In particular, the proof given in the sequel goes through as long as $\epsilon = \Omega(N^{-1/12})$.)

2.1 Failure of a naive solution

Trying to prove the theorem above, the following solution comes to mind. Fix some efficiently computable erasure code E that can decode in the presence of up to $(\frac{1}{2} - \epsilon)$-fraction of erasures. Then, given a CNF formula $\phi(x_1 \dots x_n)$, construct a polynomial size circuit C_E, that on input $(x_1 \dots x_n, y_1 \dots y_\ell)$ checks that $\vec{y}$ is a proper encoding of $\vec{x}$ using the code E. Next, use the standard Cook reduction [7] to (efficiently) transform the circuit C_E into a CNF formula ϕ_E, and finally set $\phi' = \phi \wedge \phi_E$. Now, any satisfying assignment to ϕ' must also satisfy ϕ. Moreover, since E is an erasure code, then as long as we get more than half of the bits, we should be able to recover the full assignment.

Unfortunately, this solution does not work. The problem is that the standard transformation of Boolean circuits into CNF formulas introduces additional temporary variables $z_1, \dots, z_m$ on top of the original variables $x_1, \dots x_n, y_1, \dots, y_\ell$, and the number of these new variables is linear in the size of the circuit (and thus, polynomial in $n + \ell$). Therefore, a partial assignment to ϕ' may give values to many of the z_i's and to only a very small number of the y_i's or x_i's.

2.2 First attempt: Erasure of nearly half of the bits

Although the naive solution above does not work, it can be modified to prove Theorem 2 which deals with erasure of (a bit less than) half of the variables. This is done using some "padding" tricks. The same padding technique will later be used in the proof of Theorem 1.

We start by stating the existence of good deterministic constructions of erasure codes. Perhaps the simplest construction is due to Alon [1], in which a Reed-Solomon code is concatenated with the simplex binary code. The result (with the appropriate setting of the parameters) is summarized in the next proposition.

Proposition 2.1 ([1]) *There exists a binary linear code with codewords of size n and dimension $k > \sqrt{n}$, which can correct up to $n(\frac{1}{2} - \epsilon)$ erasures for any $\epsilon > 0$.[2] Encoding and decoding can be done in polynomial time (in n).*

[2] In fact, this construction can handle even $\epsilon = o(1)$. Specifically, it works for $\epsilon = \Omega(n^{-1/3})$.

Proof of Theorem 2. We augment the naive solution from above in a way that ensures a vast majority of "useful variables" in the new formula ϕ', and just a handful of "unuseful" (or temporary) variables. Thus any assignment to slightly more than half of these variables would enable recovery of the full assignment.

Intuitively, we start from the same formula as in the naive solution, and then "replicate" the useful variables until they become the vast majority. To do that, we use the observation that it is possible to check equality of two variables using a CNF formula without adding any temporary variables: To check that $a = b$, we just need to add the terms $(a - \overline{b}) \wedge (b - \overline{a})$ to the formula. A more detailed description follows:

Let $C_E(x_1 \dots x_n, y_1 \dots y_\ell)$ be a Boolean circuit that checks that $\vec{y} = E(\vec{x})$, where E is the erasure-code from Proposition 2.1. By the properties of this code, it is sufficient to set $\ell = n^2$. Moreover, since this is a linear code, it can be represented by an $\ell \times n$ Boolean matrix, and thus the size of the circuit C_E can be as small as ℓn. Thus, we can transform C_E into a CNF formula using only $\ell n = n^3$ temporary variables. Denote the resulting CNF formula by $\phi_E(x_1 \dots x_n, y_1 \dots y_{n^2}, z_1 \dots z_{n^3})$.

We now use padding to replicate the y_i's. For each variable y_i we introduce $n^2 - 1$ new variables $y_i^1, \dots, y_i^{n^2-1}$, and add clauses to make sure that $y_i = y_i^j$. Namely, for each $i = 1, \dots, n^2$, $j = 1, \dots, n^2 - 1$, we have the two clauses $(\overline{y_i} - y_i^j) \wedge (y_i - \overline{y_i^j})$. Denote the formula which consists of these $2n^2(n^2-1)$ clauses by ϕ_{eq}. Then, we set

$$\phi' = \phi \wedge \phi_E \wedge \phi_{eq}$$

The CNF formula ϕ' has $N = n + n^4 + n^3$ variables (n x-variables, n^4 y-variables, and n^3 z-variables). We now show that from any $(\frac{1}{2} + \epsilon)N$ of the bits of a satisfying assignment to ϕ', we can retrieve $\vec{x}$. By the setting of the parameters above, we get

$$\begin{aligned}\left(\frac{1}{2} + \epsilon\right) N &= \left(\frac{1}{2} + \epsilon\right)(n + n^4 + n^3) \\ &> n + n^3 + \left(\frac{1}{2} + \epsilon - \frac{1}{n} - \frac{1}{n^3}\right) n^4 \\ &> n + n^3 + \left(\frac{1}{2} + \frac{\epsilon}{2}\right) n^4 \qquad \text{(for large enough } n\text{)}\end{aligned}$$

Since there are at most n^3 z-variables and at most n x-variables, then we must have an assignment to at least $\left(\frac{1}{2} + \frac{\epsilon}{2}\right) n^4$ of the y-variables. Since each y_i has exactly n^2 copies, we must have an assignment to at least $\left(\frac{1}{2} + \frac{\epsilon}{2}\right) n^2$ different y_i's and this is enough to use the decoding algorithm and decode the satisfying assignment $\vec{x}$ of the formula ϕ. ■

Remark. Note that although the padding can be enhanced to deal with more erasures, there are no codes with polynomial expansion which are able to correct $n/2$ erasures or more (see [11, Eq. (4) on Page 43]).

2.3 Erasures of nearly all the bits

We now show how to overcome erasures of all but $n^{\frac{1}{2}+\epsilon}$ of the bits in a satisfying assignment. This is surprisingly better than the correction ability of erasure codes. However, we note that we are in fact in a much better position than in the case of a generic erasure code, since we can verify the answer, by checking if it satisfies the given CNF formula ϕ. Therefore, if we manage to reduce the number of "candidate solutions" to a polynomial in n, we can then go over all of them until we find the right one.

Proof of Theorem 1. The outline of the proof is as follows. We first show that if we have polynomial number of random linear equations in n variables, then any "sufficiently large" subset of these equations is of dimension at least $n - O(\log n)$, and thus leaves only a polynomial number of candidate solutions to the entire equation system. This, together with the ability to verify solutions, yields an "erasure-code" with the ability to correct up to $n - \sqrt{n}$ erasures. We then use the construction from the previous subsection with this "improved code", thus yielding the needed result.

The probabilistic transformation. Let ϵ be any positive constant, and denote $c = 2/\epsilon$. Given a CNF formula over n variables $\phi(x_1 \dots x_n)$, we start by choosing uniformly at random an $n^c \times n$ Boolean matrix B. (The matrix B plays a role similar to that of the generating matrix for the erasure code from the proof of

Theorem 2.) Let $C_B(x_1 \ldots x_n, y_1 \ldots y_{n^c})$ be a circuit which verifies that $\vec{y} = B\vec{x}$. Using the nice structure of linear equations, we can actually write C_B as a CNF formula using only n^{c+1} temporary variables. Denote the resulting formula by $\phi_B(x_1 \ldots x_n, y_1 \ldots y_{n^c}, z_1 \ldots z_{n^{c+1}})$.

We now use the same padding trick as in the proof of Theorem 2. Namely, for each variable y_i we introduce additional $n^c - 1$ new variables y_i^j $(j = 1 \ldots n^c - 1)$, and we encode the equalities $y_i = y_i^1 = \ldots$ in a CNF formula ϕ_{eq}. (Recall that for all i, j, ϕ_{eq} contains the two clauses $(\overline{y_i} - y_i^j) \wedge (y_i - \overline{y_i^j})$.) Finally we set $\phi' = \phi \wedge \phi_B \wedge \phi_{eq}$, thus obtaining a formula over $N = n + n^{2c} + n^{c+1}$ variables (n x-variables, n^{2c} y-variables and n^{c+1} z-variables).

Dealing with big erasures in the resulting formula. Next we show that (with high probability over the choice of B), it is possible to recover a satisfying assignment for ϕ from any $N^{\frac{1}{2}+\epsilon}$ bits in a satisfying assignment for ϕ'. By the setting of the parameters above we have

$$\begin{aligned} N^{\frac{1}{2}+\epsilon} &= (n + n^{2c} + n^{c+1})^{\frac{1}{2}+\frac{2}{c}} \\ &> n^{2c(\frac{1}{2}+\frac{2}{c})} \\ &= n^{c+4} \\ &\geq n^{c+3} + n^{c+1} + n \qquad \text{(for large enough } n) \end{aligned}$$

Since we have only n x's and n^{c+1} z's, this means that we are left with at least n^{c+3} y-variables. Since each of these variables has exactly n^c copies, then we must have the value of at least n^3 different y_i's.

Denote the partial vector of the known y_i's by $\vec{y}\,'$, and denote by B' the sub-matrix of B which is induces by taking only the rows which correspond to the known y_i's. This gives us the linear system $B'\vec{x} = \vec{y}\,'$. We next show that with probability of at least $1 - 2^{-n}$, the rank of B is at least $n - 2c\log n$. Hence, the number of candidate solutions is at most n^{2c}, and we can use exhaustive search to check which of them satisfies the original formula ϕ.

So it remains to show that when we choose at random an $n^c \times n$ binary matrix B, then with probability at least $1 - 2^{-n}$, every $n^3 \times n$ sub-matrix of B has rank of at least $r = n - 2c\log n$. We prove this by a simple counting argument. Notice that

- The number of $n^3 \times n$ sub-matrices of B is $\binom{n^c}{n^3} \leq n^{cn^3} = 2^{(c\log n)n^3}$.
- The number of r-dimensional subspaces of Z_2^n is at most 2^{n^2} (since every subspace, and in particular every r-dimensional subspace, can be defined by an $n \times n$ matrix).
- For any fixed subspace of dimension r and any fixed row of B, the probability that this row belongs to that subspace is exactly $2^{r-n} = 2^{-2c\log n}$. Thus, the probability that the row space of a fixed $n^3 \times n$ sub-matrix of B is contained in this fixed r-dimensional subspace is exactly $2^{-2(c\log n)n^3}$.

Using the union bound, we get

$$\begin{aligned} &\Pr[\exists \text{ sub-matrix } B'_{n^3\times n} \text{ of rank } \leq r] \\ &\leq \sum_{B'_{n^3\times n}} \sum_{r\text{-dimensional } V} \Pr[\text{row space of } B' \text{ is contained in } V] \\ &\leq 2^{(c\log n)n^3} \cdot 2^{n^2} \cdot 2^{-2(c\log n)n^3} = 2^{n^2-(c\log n)n^3} < 2^{-n} \end{aligned}$$

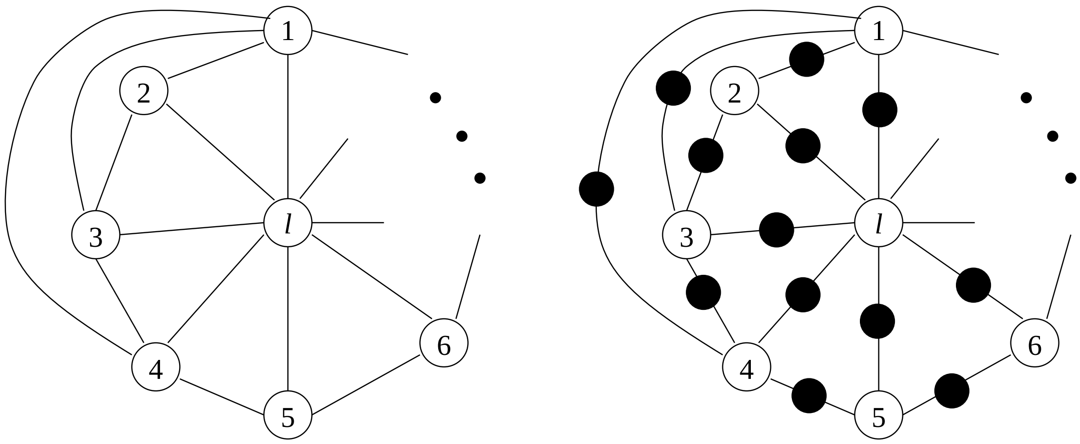
■

3 Graph Isomorphism

We consider the graph isomorphism problem: given two isomorphic graphs G and H on n vertices find an isomorphism between them. We study the complexity of the graph isomorphism problem assuming that we have access to an oracle that provides us with a partial isomorphism on a subset of certain size of the vertices for arbitrary isomorphic pairs of graphs. We stress that the partial isomorphism which is provided by the oracle must be part of a complete isomorphism between the graphs G and H.

We use the notation $G \sim H$ for G isomorphic to H. We denote the vertex set of a graph G by $V(G)$ and the edge set of G by $E(G)$.

We prove the following.



(a) The graph A_l (b) The graph B_l

Figure 1: the graphs A_l and B_l. Note that A_l contains one vertex of degree $l-1$, one of degree 5, two of degree 4, and all the vertices have degree 3.

Theorem 3 *Suppose we have access to an oracle that given any two isomorphic graphs on m vertices, reveals a partial map on at least $(3+\epsilon)\log m$ vertices (for some constant $\epsilon > 0$) which is part of an isomorphism between the two graphs. Then we can solve the graph isomorphism problem for n vertex graphs by a deterministic algorithm in $n^{O(1)}$ time.*

Proof: The outline of the proof is as follows: Given two isomorphic graphs G and H, we first query the oracle on G and H, thus obtaining a partial map on $t \geq (3+\epsilon)\log n$ nodes. What we would like to do next is to remove these t nodes from both G and H, and then apply the algorithm recursively to the remaining, smaller, graphs. Unfortunately, this does not work, since we have no guarantee that the isomorphism that we get on the remaining graphs will be "compatible" with the isomorphism that we got for the first t nodes. To overcome this, we do not just remove the t nodes from the graphs, but instead we replace them with some gadget that will enforce "compatibility" between the isomorphisms. Since this gadget contains less than t nodes, then the resulting graphs will be smaller, and so the procedure is guaranteed to terminate within at most n oracle calls. More details follow.

The gadget. Let l be an integer, and let A_l be a graph on l vertices with the following properties.

1. A_l contains $2l$ edges,
2. Each vertex of A_l has degree at least 3, and
3. A_l has no non-trivial self automorphisms.

A construction of such a graph (for $l \geq 7$) is depicted in Figure 1(a). We now replace each edge of A_l by a path of length 2, thus obtaining a graph B_l on $3l$ vertices, as depicted in Figure 1(b). In the sequel we refer to the vertices of B_l which came from A_l as A-vertices, and to the other vertices of B_l we refer as middle-vertices.

The algorithm. Let G and H be graphs on n vertices. We begin by querying the oracle on (G, H). The answer of the oracle will provide us with a partition of the vertices of both G ${}_1) \cup V(G_2)$ and $V(H) = V(H_1) \cup V(H_2)$, such that $t = |V(G_2)| = |V(H_2)| \geq (3+\epsilon)\log n$, as well as a partial map between $V(G_2)$ and $V(H_2)$ which is part of an isomorphism between G and H. Then we construct a new oracle query (G', H') as follows.

Below we denote by G_1 the induced subgraph of G on $V(G_1)$ and by H_1 the induced subgraph of H on $V(H_1)$. Also, denote $\epsilon' = \epsilon/4$ and let $l = \lceil (1+\epsilon')\log n \rceil$. To construct G', H', we replace the vertices of G_2 and H_2 in G and H, respectively, by copies of the graph B_l. Thus, the vertex sets of the resulting graphs are

$$V(G') = V(G_1) \cup V(B_l) \text{ and } V(H') = V(H_1) \cup V(B_l)$$

Since B_l contains exactly $3l$ vertices, then the number of vertices in G', H' is

$$n - t + 3l \leq n - (3+\epsilon)\log n + 3\left\lceil 1 + \frac{\epsilon}{4}\log n \right\rceil < n - \frac{\epsilon}{4}\log n + 3 < n$$

The edges of G' (resp. H') consist of the edges of G_1 and B_l (resp. H_1 and B_l) as well as some more edges which connect these components. We refer to these additional edges as "cross-edges". We want the cross-edges to have two important properties.

1. Two vertices in $V(G_1)$ have the same set of neighbors from $V(B_l)$ if and only if they had the same set of neighbors in $V(G_2)$ in the original graph G (and the same holds for H).
2. The cross-edges should "encode" the map that we know between G_2 and H_2.

We achieve this as follows. Fix an arbitrary labeling of the A-vertices of B_l. For any vertex $x \in V(G_1)$, denote by S_x the set of its neighbors from $V(G_2)$ in the original graph G, and consider the collection $\mathcal{C}_G = \{S_x : x \in V(G_1)\}$. Clearly, $\mathcal{C}_G$ contains at most $|V(G_1)|$ different subsets of $V(G_2)$. We then map each subset $S_x \in \mathcal{C}_G$ to a unique subset of the A-vertices of B_l of size at least 3. This is possible since B_l contains l A-vertices, and therefore the number of subsets of size at least 3 is

$$\begin{array}{l} 2^l - 1 - l - \binom{l}{2} \quad > \quad 2^l - l^2 \\ \geq \quad 2^{(1+\epsilon')\log n} - ((1+\epsilon')\log n)^2 \\ = \quad n^{1+\epsilon'} - (1+\epsilon')^2 \log^2 n \quad > \quad n \quad > \quad |V(G_1)| \end{array}$$

Similarly, for each vertex $y \in V(H_1)$ we denote by S_y the set of its neighbors from $V(H_2)$ in the original graph H, we consider the collection $\mathcal{C}_H = \{S_y : y \in V(H_1)\}$, and map each subset $S_y \in \mathcal{C}_H$ to a unique subset of size at least 3 of the A-vertices in B_l. To reflect the map between G_2 and H_2 which was revealed by the oracle, we make sure that if a set $S_x \subseteq V(G_2)$ is mapped by the answer of the oracle to a set $S_y \subseteq V(H_2)$ then we map both S_x and S_y to the same subset of A-vertices in B_l.

Now we are ready to define the cross-edges in G' and H'. A vertex $x \in V(G_1)$ will be connected to each point in the subset of $V(B_l)$ which has been assigned to S_x, and similarly a vertex $y \in V(H_1)$ will be connected to each point of the subset of $V(B_l)$ which has been assigned to S_y.

The next call to the oracle will be on G' and H'. We recursively apply the above procedure, until we are left with a trivial problem. Once the recursive call returns an isomorphism between G' and H', we take the union of the map between G_1 and H_1 (which are part of G', H', respectively) and the map between G_2 and H_2 which was given to us by the oracle. Below we prove that this union is necessarily an isomorphism between the original graphs G and H.

Analysis. The crucial observation in this proof is the following

Claim 3.1 *Any isomorphism between G' and H' must map $V(B_l) \subseteq G'$ to $V(B_l) \subseteq H'$ such that each A-vertex of B_l is mapped to itself (i.e., to a vertex with the same label according to the fixed labeling of the A-vertices).*

Proof: Notice that by the construction above, every vertex in G_1 is connected to at least 3 vertices in B_l, even if it has no neighbors in G (and the same holds for every vertex in H_1). Since B_l is constructed so that all the A-vertices have degree at least 3, then the only vertices in G', H' which have degree 2 or less are the middle-vertices of B_l. It follows that middle-vertices in G' must be mapped to middle-vertices in H'. Since the only neighbors of the middle-vertices are the A-vertices, it also follows that A-vertices in G' must be mapped to A-vertices in H'. Finally, since the graph A_l (and therefore also B_l) does not have any non-trivial self automorphism, we conclude that a copy of each A-vertex in G' must be mapped to a copy of the same A-vertex in H'. ■

To complete the proof, note that if a node $x \in V(G_1)$ is mapped to a node $y \in V(H_1)$, then they both have the same set of neighbors from B_l. By construction, this means that the map between H_2 and G_2 maps the vertices in S_x to the vertices in S_y. ■

Choosing between many maps. Originally, we came across this problem when we tried to construct a procedure that gives a partial solution to graph isomorphism. That procedure was supposed to offer a polynomial number of candidates, such that one of them was a correct partial solution. Let us show that this would be enough to achieve an algorithm for GI which is faster than the algorithms known today.

Corollary 3.2 *Suppose we have access to an oracle that given two isomorphic graphs on m vertices outputs a polynomial number of maps $\alpha_1, \ldots, \alpha_T$ on at least ϵm vertices for some fixed $\epsilon > 0$, such that at least one of the maps is part of an isomorphism between the graphs. Then we can solve the Graph Isomorphism problem for n vertex graphs by a deterministic algorithms in $n^{O(\log n)}$ time.*

Proof: We use the same strategy as in the proof of Theorem 3. At each recursive level, after the call to the oracle on G and H we try to proceed with one of the maps $\alpha_1, \ldots, \alpha_T$ returned by the oracle. We choose one of the maps α_i and we construct G' and H' as above. If we succeed to construct an isomorphism between G' and H' then using α_i to map $V(G_2)$ to $V(H_2)$ we obtain an isomorphism between G and H as in the proof of Theorem 3. If G' is not isomorphic to H' we try again with another α_j.

Since we required that the oracle answers with maps on at least ϵ fraction of the vertices at each call for some constant $\epsilon > 0$, we only need $O(\log n)$ recursive levels for n vertex graphs. At each level we may have to make $T = n^{O(1)}$ recursive calls to our algorithm. Thus the running time of the algorithm will be $n^{O(\log n)}$.
■

4 Shortest vectors in real lattices

In this section we consider the problem of finding the shortest non-zero vector in an n-dimensional lattice. Throughout this section, an n-dimensional lattice L in R^m (where $m \geq n$) is represented by a basis matrix $B_{m\times n}$, such that $L = L(B) = \{Bx : x \in Z^n\}$. We also assume that all the "real numbers" which we deal with are represented with some finite precision, where the number of precision bits is polynomial in m. We then prove the following.

Theorem 4 *For any $\epsilon > 0$, there exist an efficient probabilistic algorithm* $\textsc{Enc}_{sv}$*, and an efficient deterministic algorithm* $\textsc{Rec}_{sv}$ *such that*

1. *If B is an $m \times n$ real matrix, then $B' = \textsc{Enc}_{sv}(B)$ is an $M \times m$ real matrix with $M = \max\{2m, \lceil m^{1/\epsilon}\rceil\}$ (so B' is a generator matrix for an n-dimensional lattice in R^M).*
2. *With probability $1 - 2^{-m}$ (over the coins of $\textsc{Enc}_{sv}$), the matrix B' has the following property: For any partial vector s' which contains $\lfloor M^\epsilon \rfloor$ entries in the shortest non-zero vector in $L(B')$, the output of $\textsc{Rec}_{sv}(B, B', s')$ is the shortest non-zero vector in $L(B)$.*

Proof: The strategy for generating B' from B is as follows: We set $B' = TB$ where T is an $M \times m$ transformation matrix with the following properties

1. $\forall\, x \in R^m$, $\|Tx\| = \|x\|$ (where $\|\cdot\|$ denotes the Euclidean norm).
2. Every $m \times m$ sub-matrix of T is non-singular.

Thus, B' is a basis for the lattice $L(B') = \{TBx : x \in Z^n\} = \{Tp : p \in L(B)\}$, and (by Property 1) the shortest non-zero vector in this lattice is $s' = Ts$, where s is the shortest non-zero vector in $L(B)$. Any m entries in s' can be viewed as a solution to the linear system of equations $s'' = T's$, where s'' is the m sub-vector of s' whose entries are known, and T' is the corresponding $m \times m$ sub-matrix of T. By Property 2, T' is non-singular, and thus we can reconstruct s from s''.

It is therefore sufficient to show how to pick a transformation matrix T as above. For this, we use the following fact

Fact 4.1 *An $M \times m$ matrix T satisfies Property 1 if and only if all the columns of T are orthogonal to each other and have norm one.*

Proof omitted.

To pick the matrix T, we pick the columns of T one at a time with r-bit precision, where r is a parameter to be determined later. Throughout this process we maintain the invariant that after choosing k vectors, every $k \times k$ sub-matrix is non-singular (with high probability).

For simplicity, in the description below we ignore the effect of rounding errors which result from working with finite precision. We briefly discuss these matters at the end of Appendix A. Also, below we assume that we have a polynomial time subroutine which on input $(1^l, 1^r)$ returns a uniformly distributed point inside the l-dimensional unit sphere with r-bit precision. In this extended abstract we do not describe the implementation of such a subroutine. However, we use the following important fact

Fact 4.2 *Assume that the vector $a = \langle \alpha_1 \dots \alpha_l \rangle$ is chosen uniformly at random in the l-dimensional unit sphere with r-bit precision. Then for every i we have $\Pr[\alpha_i = 0] \le 2^{-r}\sqrt{l}$.*

Proof omitted.

Picking the matrix T. Denote the columns of T by $t_1, t_2, \dots t_m$. The first column t_1 is chosen at random inside the M-dimensional unit sphere, and then scaled so that its Euclidean norm is one. After this choice, the invariant holds if none of the entries in this vector is zero. Using Fact 4.2, this happens with probability of at least $1 - 2^{-r}M^{3/2}$.

Assume now that we already picked columns t_1 through t_{k-1}, which satisfy the invariant. To pick column t_k, we compute an orthonormal basis for the $(M-k+1)$-dimensional subspace of R^M which is orthogonal to $t_1 \dots t_{k-1}$. Denote the vectors in this basis by $b_1, \dots b_{M-k+1}$. We then uniformly pick a vector $a = \langle \alpha_1, \dots, \alpha_{M-k+1} \rangle$ inside the $(M-k+1)$-dimensional unit sphere and scale it so that its Euclidean norm is one. Finally, we set $t_k = \sum \alpha_i b_i$. The correctness of this construction follows from the proposition below

Proposition 4.3 *Assume that columns $t_1 \dots t_{k-1}$ satisfy the invariant that all the $(k-1) \times (k-1)$ sub-matrices of them are non-singular, and that column t_k is chosen by the procedure above. Then, with probability of at least $1 - 2^{-r}\sqrt{M}\binom{M}{k}$, all the $k \times k$ sub-matrices of $t_1 \dots t_k$ are non-singular.*

Proof: See Appendix A.

Using Proposition 4.3, we can now compute the precision we need to get a transformation matrix T with the desired properties. Notice that by construction, T must satisfy Property 1. It remains to compute the probability that it also satisfies Property 2. This probability can be lower-bounded as follows

$$\begin{aligned} \Pr[\text{Property 2 does not hold}] &\le \Pr[\text{Invariant is violated}] \\ &\le \sum_{k=1}^{m} 2^{-r}\sqrt{M}\binom{M}{k} \;<\; m \cdot 2^{-r}\sqrt{M}\binom{M}{m} \;<\; 2^{\log m - r + (m+\frac{1}{2})\log M} \end{aligned}$$

Recall now that $M = m^{1/\epsilon}$, and so $\log m - r + (m + \dfrac{1}{2})\log M = -r + \log m(1 + \dfrac{m}{\epsilon} + \dfrac{1}{2\epsilon})$. Therefore, if we set $r = m + \lceil \log m \left(1 + \frac{m}{\epsilon} + \frac{1}{2\epsilon}\right) \rceil$ then we have $\Pr[\text{Property 2 does not hold}] < 2^{-m}$, as needed. ∎

Discussion. There are a few points worth mentioning about the construction above.

Representation of partial answers. We stress that in the construction above we can get an arbitrarily small polynomial fraction of the entries in the shortest vector in $L(B')$. However, we must get *all the bits* in each of these entries. The above construction fails if instead we only get, say, some fraction of the bits in each of the entries. We do not know of any construction that solves this case.

Complex lattices. If we consider complex lattices, then we have a deterministic construction for the transformation matrix T. One can simply set M to be the first power of two which is larger than $m^{1/\epsilon}$, and then take as T the first m columns of the $M \times M$ FFT matrix. It is easy to verify that this matrix indeed satisfies Properties 1 and 2 over the complex numbers. This yields a similar theorem to Theorem 4, but with a deterministic construction.

5 Other $\mathcal{NP}$-complete problems

Our results for SAT (Theorem 1) allow to obtain similar results for other $\mathcal{NP}$-complete problems that SAT is reducible to via a reduction with certain properties. In many cases, a reduction from 3-SAT to a given language L constructs an instance of the given membership problem such that there is a simple projection from witnesses for membership in L to satisfying assignments of the corresponding formula. Our idea is to try to pad such reductions in a way that ensures that from any small fraction of a witness we can recover a sufficient part of a satisfying assignment of the corresponding formula.

We illustrate our arguments by a formal proof for Graph 3-Colorability.

Theorem 5 : *For any $\epsilon > 0$, there exist an efficient probabilistic algorithm* ENC_{3col}, *and an efficient deterministic algorithm* REC_{3col} *such that*

1. *If G is a graph over n vertices, then $G' = \mathrm{ENC}_{3col}(G)$ is a graph over $N = n^{O(1)}$ vertices.*
2. *With probability $1 - 2^{-n}$ (over the coins of ENC_{3col}), the graph G' has the following property: If π' is any 3-coloring of $N^{\frac{7}{8}+\epsilon}$ of the vertices in G' which can be extended to a full 3-coloring of G', then $\mathrm{REC}_{3col}(G, G', \pi')$ is a 3-coloring of G.*

Proof: First we construct a formula ϕ using Cook's reduction [7] from 3-Colorability to 3-SAT, and we apply our Theorem 1 to construct a formula ϕ' in $m = n^{O(1)}$ variables with the property that from any $m^{1/2+\epsilon}$ bits of a satisfying assignment to ϕ' we can reconstruct a satisfying assignment to ϕ. Next we use a padded version of the standard reduction from 3-SAT back to 3-Colorability [12, 8].

Recall that the original reduction constructs a graph on $M = 6h + 2m + 2$ vertices, where h is the number of clauses and m is the number of variables in the formula. There are 6 vertices labeled by $C_1, \ldots, C_6$ associated with the clause C, and 2 vertices labeled by the complementary literals x_i and $\overline{x_i}$ associated with the variable x_i. There are two more vertices labeled TOP and GROUND. For each clause C, the 6 vertices are arranged in the graph to form 2 triangles C_1, C_2, C_4 and C_3, C_5, C_6 connected to each other by an edge between C_4 and C_5. The vertices C_1, C_2 and C_3 are each connected to one of the literals of the clause C. The vertex C_6 is always connected to the TOP. All the pairs of complementary literals are connected by an edge, and each literal is connected to GROUND. This graph is 3-colorable if and only if the formula has a satisfying assignment. Moreover, it has the property, that only two colors can be used to color the vertices that are labeled by literals, since the third color is assigned to the Ground. From the colors assigned to the vertices labeled by the literals one directly gets a satisfying assignment to the formula.

We pad the reduction as follows. We add $\frac{1}{2}m^3 - 1$ extra copies of each variable and its negation, and add edges that connect each pair of complementary literals. As before, each vertex that is labeled by a literal is connected to GROUND. We denote the graph we obtain by G'. This graph has $N = m^4 + 6h + 2$ vertices. Assume now that we are given a 3-coloring of $N^{\frac{7}{8}+\epsilon}$ of the nodes. Since $h \leq \binom{m}{3} < m^3/6$, then

$$
\begin{aligned}
N^{\frac{7}{8}+\epsilon} &= \left(m^4 + 6h + 2\right)^{\frac{7}{8}+\epsilon} \\
&> m^{4(\frac{7}{8}+\epsilon)} = m^{3.5+4\epsilon} \\
&> \frac{1}{2}m^{3.5+\epsilon} + m^3 + 2 \quad > \quad \frac{1}{2}m^{3.5+\epsilon} + 6h + 2
\end{aligned}
$$

Thus, among these $N^{\frac{7}{8}+\epsilon}$ vertices of G' there must be at least $m^{3.5+\epsilon}$ vertices which are labeled by literals. Since for each variable there are m^3 vertices which are labeled by this variable (and its negation), then we learn the value of at least $m^{3.5+\epsilon}/m^3 = m^{\frac{1}{2}+\epsilon}$ of the variables. This gives us at least $m^{\frac{1}{2}+\epsilon}$ bits of a satisfying assignment to ϕ', from which (by Theorem 1) we can construct a satisfying assignment to ϕ. Since Cook's reduction is witness preserving this gives us a 3-coloring of the original graph G. ■

We are able to prove results of the same type also for Vertex Cover and the Clique problem. We note that in the case of the Clique problem (and Independent Set), the proof is even more straightforward, we can use the standard reduction without padding.

Acknowledgments

We would like to thank Dan Boneh for useful discussions, Ran Raz for his help in the proof of Subsection 2.3, and Ronny Roth for valuable lessons on error correcting codes.

While working on the SAT problem, we asked Sanjeev Arora about it. He suggested adding to standard probabilistically checkable proofs another layer of encoding, thus, obtaining resistance to erasure of an ϵ fraction of the bits, for some small constant fraction ϵ. It turned out eventually that the methods presented in this paper imply better results (see Theorems 1 and 2 above).

References

[1] N. Alon, Packing with Large Minimum Kissing Numbers, To appear in *Siam Journal on Discrete Math.*

[2] S. Arora, Personal communication.

[3] S. Arora and S. Safra, Probabilistic checking of proofs, In *Proc. of "33-rd IEEE Symposium on the Foundations of Computer Science"*, 1992, pp. 2-13.

[4] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, Proof verification and hardness of approximation problems, In *Proc. of the 33-rd IEEE FOCS*, 1992, pp. 14-23.

[5] W. B. Alexi and B. Chor and O. Goldreich and C. P. Schnorr, RSA/Rabin functions: certain parts are as hard as the whole, in *SIAM J. Computing*, **17**:2, pages 194–209, 1988.

[6] M. Blum, S. Micali, How to generate cryptographically strong sequences of pseudo-random bits, In *SIAM J. Computing*, 13(4): 850-863, 1984.

[7] S. A. Cook. The complexity of theorem-proving procedures. In *Proc. 3rd ACM Symp. on Theory of Computing*, pages 151–158, 1971.

[8] M. Garey, D. Johnson, L. Stockmeyer, Some simplified NP-complete graph problems, *Theoretical Computer Science*, 1 (1976), pp. 237-267.

[9] O. Goldreich, L. Levin, A hard-core predicate for all one-way functions, In *Proc. 21st ACM Symp. on Theory of Computing*, pages 25-32, 1989.

[10] J. Justesen, A Class of Constructive Asymptotically Good Algebraic Codes, *IEEE Transactions on Information Theory*, **18** pp. 652-656, 1972.

[11] F.J. MacWilliams, N.J.A. Sloane, *The Theory of Error-Correcting Codes,* North Holland, Amsterdam, 1977.

[12] L. Stockmeyer, Planar 3-colorability is NP-complete, SIGACT News 5 (3), 1973, pp. 19-25.

A Proof of Proposition 4.3

Below we prove that if we pick t_k according to the procedure above and consider the $k \times k$ sub-matrix which is induced by taking only the first k entries in each vector, then the probability that this matrix is singular is at most $2^{-r}\sqrt{M}$. The same analysis holds for any other fixed sub-matrix, and so we obtain Proposition 4.3 using the union bound.

A technical lemma. We start by setting a few notations.

- If V is a linear space, then by $\dim(V)$ we denote the dimension of V. If B is a matrix, then the rank of B is denoted $\mathrm{rank}(B)$.
- For any vector $x \in R^M$, denote by $x[\cdots k]$ (resp. $x[k+1\cdots]$), the vector which is obtained by taking only the first k (resp. the last $M-k$) entries of x. Similarly, if V is a set of vectors then we denote

$$V[\cdots k] \stackrel{\text{def}}{=} \{x[\cdots k] \;:\; x \in V\} \quad\text{and}\quad V[k+1\cdots] \stackrel{\text{def}}{=} \{x[k+1\cdots] \;:\; x \in V\}$$

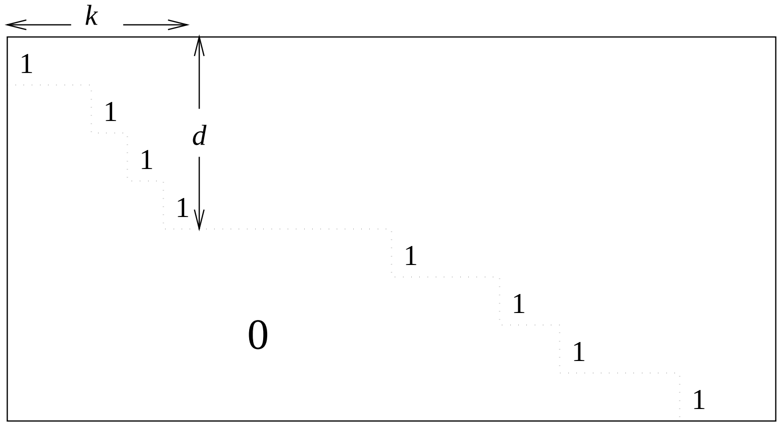


Figure 2: A basis of $V^\perp$ in row-Echelon form. The dimension of $V^\perp[\cdots k]$ equals the number of rows with non-zero entries in positions $1 \ldots k$.

The main technical lemma which we need in the proof is the following

Lemma A.1 *Let T be a $M \times (k-1)$ real matrix, such that $M \geq 2k-1$. Denote the columns of T by $t_1 \ldots t_{k-1}$, and denote by $V^\perp$ the $(M-k+1)$-dimensional subspace which is orthogonal to $t_1 \ldots t_{k-1}$. If T satisfies the invariant that no $(k-1) \times (k-1)$ sub-matrix of T is singular, then* $\dim(V^\perp[\cdots k]) = k$.

(Recall that $V^\perp$ itself is a subspace of R^M of dimension $M-k+1$. The lemma asserts that if we only consider the first k entries in each vector in $V^\perp$, then the induced subspace of R^k has dimension k.)

Proof: Since $V^\perp[\cdots k] \subseteq R^k$ then clearly $\dim(V^\perp[\cdots k]) \leq k$. We now show that also $\dim(V^\perp[\cdots k]) \geq k$. Let $B = \{b_1 \ldots b_{M-k+1}\}$ be a basis for $V^\perp$, and assume that B is in row-Echelon form (i.e., it was obtained by taking any basis of $V^\perp$, putting it in an $(M-k+1) \times M$ matrix and applying Gaussian elimination to the rows of this matrix). We depict the general form of B in Figure 2.

Denote by d the index ($d \in \{1 \ldots M-k+1\}$) such that the vectors $b_1, \ldots, b_d$ contain at least one non-zero entry in positions $1 \ldots k$ and the vectors $b_{d+1}, \ldots, b_{M-k+1}$ contains only zeros in these positions. Since B is a basis of $V^\perp$, then $B[\cdots k]$ spans the space $V^\perp[\cdots k]$. However, since only the vectors $b_1, \ldots, b_d$ contains non-zero entries in locations $1 \ldots k$, then the vectors $b_1[\cdots k], \ldots, b_d[\cdots k]$ alone span the space $V^\perp[\cdots k]$. Finally, since B is in row-Echelon form, then these vectors are linearly independent. We therefore conclude that

$$d = \dim(V^\perp[\cdots k]) \tag{1}$$

Consider now the vectors $b_{d+1}, \ldots, b_{M-k+1}$. By construction, they are linearly independent, they contain only zeros in locations $1 \ldots k$, and they are all orthogonal to $t_1, \ldots, t_{k-1}$. It follows that if we only consider entries $k+1 \ldots M$ in these vectors, we get

$$b_{d+1}[k+1\cdots],\ \ldots, b_{M-k+1}[k+1\cdots] \text{ are linearly independent} \tag{2}$$

and

$$b_{d+1}[k+1\cdots],\ \ldots, b_{M-k+1}[k+1\cdots] \text{ are orthogonal to } t_1[k+1\cdots],\ \ldots, t_{k-1}[k+1\cdots] \tag{3}$$

Below we denote by U the subspace of R^{M-k} which is spanned by $b_{d+1}[k+1\cdots],\ \ldots, b_{M-k+1}[k+1\cdots]$, and let $U^\perp$ denote the subspace orthogonal to U. Equation (2) implies that $\dim(U) = (M-k+1)-d$, and therefore

$$\dim(U^\perp) = (M-k) - (M-k+1-d) = d-1 \tag{4}$$

Finally, consider the matrix $T[k+1\cdots]$. This is the $(M-k) \times (k-1)$ sub-matrix of T, containing the vectors $t_1[k+1\cdots],\ \ldots, t_{k-1}[k+1\cdots]$. Since $M-k \geq k-1$, then this matrix contains some $(k-1) \times (k-1)$ sub-matrices (which are also sub-matrices of T). By the premise of Lemma A.1, these sub-matrices must be non-singular, and therefore the rank of $T[k+1\cdots]$ must be exactly $k-1$. On the other hand, Equation (3) implies that all the columns of $T[k+1\cdots]$ belong to $U^\perp$, and so the rank of $T[k+1\cdots]$ cannot be larger than the dimension of $U^\perp$. We conclude that

$$k-1 = \mathrm{rank}(T[k+1\cdots]) \le \dim(U^\perp) = d-1$$

which implies that $\dim(V^\perp[\cdots k]) = d \ge k$, as needed. ■

Completing the proof. Armed with Lemma A.1, we are now ready complete the proof of Proposition 4.3. Recall now that to pick t_k, we compute some orthonormal basis of $V^\perp$ (which we denote by $B = \{b_1, \ldots, b_{M-k+1}\}$), pick a random vector inside the $(M-k+1)$-dimensional unit sphere (which we denote by $a = \langle \alpha_1 \ldots \alpha_{M-k+1} \rangle$), scale it to norm one and set $t_k = \text{scale-factor} \cdot \sum_i \alpha_i b_i$. (Clearly, the multiplication by the scale-factor has no effect on whether or not any sub-matrix is singular, so we ignore it below.)

Lemma A.1 asserts that the dimension of $V^\perp[\cdots k]$ is k. On the other hand, the subspace which is spanned by $t_1[\cdots k], \ldots, t_{k-1}[\cdots k]$ has dimension at most $k-1$. It follows that at least one of the vectors $b_i[\cdots k]$ $(i = 1 \ldots M-k+1)$ *does not belong* to this $(k-1)$-dimensional subspace. Assume w.l.o.g. that this vector is $b_{M-k+1}[\cdots k]$. We can now view the process of picking a inside the unit sphere in R^{M-k+1} as choosing the entries α_i one at a time, according to the appropriate distributions. Now notice that

1. The induced distribution on α_{M-k+1}, subject to any particular choice of α_1, is uniform in some interval which is symmetric around the origin.
2. For any choice of $\alpha_1 \ldots \alpha_{M-k}$, there is at most one choice of α_{M-k} which will cause $t_k[\cdots k]$ to be in the $(k-1)$-dimensional subspace which is spanned by $t_1[\cdots k], \ldots, t_{k-1}[\cdots k]$.

   For any choice of $\alpha_1 \ldots \alpha_{M-k}$, denote by $L(\alpha_1, \ldots, \alpha_{M-k})$ this unique value of α_{M-k+1} (if there is no such value, then denote $L(\alpha_1, \ldots, \alpha_{M-k}) = 0$).

Now we have

$$\begin{aligned}
\Pr_a[\text{resulting } T[\cdots k] \text{ is singular}] &\le \Pr_a[\alpha_{M-k+1} = L(\alpha_1, \ldots, \alpha_{M-k})] \\
&\overset{(a)}{=} E_{\alpha_1 \ldots \alpha_{M-k}}\left[\Pr_{\alpha_{M-k+1}}[\alpha_{M-k+1} = L(\alpha_1, \ldots, \alpha_{M-k}) \mid \alpha_1, \ldots, \alpha_{M-k}]\right] \\
&\overset{(b)}{\le} E_{\alpha_1 \ldots \alpha_{M-k}}\left[\Pr_{\alpha_{M-k+1}}[\alpha_{M-k+1} = 0 \mid \alpha_1, \ldots, \alpha_{M-k}]\right] \\
&\overset{(c)}{=} \Pr_a[\alpha_{M-k+1} = 0] \\
&\overset{(d)}{\le} 2^{-r}\sqrt{M-k+1} \quad \le 2^{-r}\sqrt{M}
\end{aligned}$$

Where Equalities (a) and (c) are the law of iterative expectations, Inequality (b) holds since α_{M-k+1} is chosen uniformly from some interval which contains the origin, and Inequality (d) follows from Fact 4.2 since we pick a with r-bit precision. This completes the proof of Proposition 4.3. ■

Working with finite precision. The above description does not take into account the effect of rounding errors, which may be introduced since we must work with finite precision. The effects of these errors may cause the transformation matrix T to fail to satisfy either of the two required Properties.

Failing to satisfy Property 1 means that the columns of T are not quite orthogonal to each other, or that their norm is not exactly one, so we may end up with a matrix T satisfying $\|Tx\| = (1 \pm \epsilon)\|x\|$, where ϵ is some expression which depends on the precision r. However, it is still possible to efficiently compute a precision r large enough so that we are guaranteed that the shortest vector in $L(B')$ is Ts (where s is the shortest vector in $L(B)$), and this is sufficient for our needs.

Failing to satisfy Property 2 may be a more serious problem, since it may then be impossible to recover s from the given entries of s'. One way to solve this problem it to choose T with $2r$ bits of precision, and then to add to each column a random vector of magnitude 2^{-r}. This will add some small factor to the deviation from Property 1, but with very high probability it would cause every sub-matrix of T to be non-singular.