CDMTCS
Research Report Series

The Parameterized Complexity of Some Fundabmental Problems in Coding Theory

Rodney G. Downey
Department of Mathematics
Victoria University
Wellington, New Zealand

Michael R. Fellows
Department of Computer Science
University of Victoria
Victoria, B.C. Canada

Alexander Vardy
Coordinated Science Laboratory
University of Illinois
Urbana, IL U.S.A

Geoff Whittle
Department of Mathematics
Victoria University
Wellington, New Zealand

CDMTCS-052
August 1997

Centre for Discrete Mathematics and
Theoretical Computer Science

The Parameterized Complexity of Some Fundamental Problems in Coding Theory

Rod G. Downey
Department of Mathematics
Victoria University, Private Bag 600
Wellington, New Zealand
`downey@math.vuw.ac.nz`

Michael R. Fellows
Department of Computer Science
University of Victoria
Victoria, B.C. V8W 3P6, Canada
`mfellows@csr.uvic.ca`

Alexander Vardy[*]
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1308 W. Main Street, Urbana, IL 61801, USA
`vardy@golay.csl.uiuc.edu`

Geoff Whittle
Department of Mathematics
Victoria University, Private Bag 600
Wellington, New Zealand
`geoff.whittle@math.vuw.ac.nz`

June 10, 1997

Abstract

The parameterized complexity of a number of fundamental problems in the theory of linear codes and integer lattices is explored. Concerning codes, the main results are that MAXIMUM-LIKELIHOOD DECODING and WEIGHT DISTRIBUTION are hard for the parametrized complexity class $W[1]$. The NP-completeness of these two problems was established by Berlekamp, McEliece, and van Tilborg in 1978, using a reduction from 3-DIMENSIONAL MATCHING. On the other hand, our proof of hardness for $W[1]$ is based on a parametric polynomial-time transformation from PERFECT CODE in graphs. An immediate consequence of our results is that bounded-distance decoding is likely to be hard for linear codes. Concerning lattices, we address the THETA SERIES problem of determining, for an integer lattice Λ and a positive integer k, whether there is a vector $x \in \Lambda$ of Euclidean norm k. We prove here for the first time that THETA SERIES is NP-complete, and show that it is also hard for $W[1]$. We furthermore prove that the NEAREST VECTOR problem for integer lattices is hard for $W[1]$. These problems are the counterparts of WEIGHT DISTRIBUTION and MAXIMUM-LIKELIHOOD DECODING for lattices, and are closely related to the well-known SHORTEST VECTOR problem. Relations between all these problems and combinatorial problems in graphs are discussed.

Keywords: parametrized complexity, linear codes, decoding, codes in graphs, integer lattices.

[*]Research supported by the Packard Foundation, the National Science Foundation, and by a grant from JSEP.

1. Introduction

Our main objective in this paper is to explore the parameterized complexity of certain fundamental computational problems in the theories of linear codes and integer lattices. There is a natural close relationship between computational problems in these areas. We prove one main combinatorial transformation, which we then use to show hardness for problems in both domains.

There has been a substantial amount of previous work on the complexity of the problems considered here. Although many of these problems are naturally parameterized, all the prior work was in the framework of NP-completeness. The following three problems, considered by Berlekamp, McEliece, and van Tilborg [BMvT78] in 1978, are of importance in the theory of linear codes:

Problem: Maximum-Likelihood Decoding
Instance: A binary $m \times n$ matrix H, a target vector $s \in \mathbb{F}_2^m$, and an integer $k > 0$.
Question: Is there a set of at most k columns of H that sum to s ?
Parameter: k

Problem: Weight Distribution
Instance: A binary $m \times n$ matrix H, and an integer $k > 0$.
Question: Is there a set of k columns of H that sum to the all-zero vector?
Parameter: k

Problem: Minimum Distance
Instance: A binary $m \times n$ matrix H, and an integer $k > 0$.
Question: Is there a set $(\neq \varnothing)$ of at most k columns of H that sum to the all-zero vector?
Parameter: k

Notice that the difference between the definitions of the Minimum Distance and Weight Distribution problems is very slight. Weight Distribution requires exactly k columns in a solution, while Minimum Distance requires at most k columns in a solution.

Berlekamp, McEliece and van Tilborg [BMvT78] proved that Maximum-Likelihood Decoding and Weight Distribution are NP-complete, by means of a reduction from 3-Dimensional Matching. They conjectured that Minimum Distance is also NP-complete, and Vardy [Var97b] recently proved this conjecture using a non-parametric reduction from Maximum-Likelihood Decoding. Since 3-Dimensional Matching is fixed-parameter tractable, these earlier results do not allow us to conclude anything about the parameterized complexity of the three problems.

Over the past few years, it was shown that many NP-complete problems are fixed-parameter tractable. For example Vertex Cover, a well-known NP-complete problem [GJ79, p. 53] which asks whether a graph G on n vertices has a vertex cover of size at most k, can be solved [BFR96] in time $O(kn + ({}^4\!/_3)^k k^2)$. Loosely speaking, the parametrized complexity hierarchy

$$\mathrm{FPT} = W[0] \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[\mathrm{P}] \subseteq \cdots \subseteq XP$$

introduced by Downey and Fellows [DF95a, DF95b] distinguishes between those problems that are fixed-parameter tractable and those that are not. For more details on the W hierarchy of parameterized complexity, see § 4, the Appendix, and references therein, in particular [DF97b].

One of our main results in this paper is a proof that MAXIMUM-LIKELIHOOD DECODING and WEIGHT DISTRIBUTION are hard for the parametrized complexity class $W[1]$. We also show that both problems belong to the class $W[2]$. The proof of $W[1]$-hardness is based on a parametric polynomial-time reduction from the PERFECT CODE problem for graphs. Such a proof establishes both $W[1]$-hardness and NP-completeness at the same time. Furthermore, an immediate consequence of this result is that bounded-distance decoding is likely to be hard for binary linear codes, unless the parametrized complexity hierarchy collapses with $W[1] = \mathrm{FPT}$.

Three closely related problems in the theory of integer lattices are natural counterparts of the three problems concerning linear codes, discussed above. These problems are defined as follows:

Problem: NEAREST VECTOR
Instance: A basis $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{Z}^n$ for a lattice Λ, a target vector $s \in \mathbb{Z}^n$, and an integer $k > 0$.
Question: Is there a vector $x \in \Lambda$, such that $\|x - s\|^2 \le k$?
Parameter: k

Problem: THETA SERIES
Instance: A basis $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{Z}^n$ for a lattice Λ, and an integer $k > 0$.
Question: Is there a vector $x \in \Lambda$, such that $\|x\|^2 = k$?
Parameter: k

Problem: SHORTEST VECTOR
Instance: A basis $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{Z}^n$ for a lattice Λ, and an integer $k > 0$.
Question: Is there a non-zero vector $x \in \Lambda$, such that $\|x\|^2 \le k$?
Parameter: k

where a lattice Λ is defined as the set of all linear combinations with integer coefficients of the elements of its basis X, and $\|\cdot\|$ denotes the Euclidean norm. Again, notice that the difference between the definitions of SHORTEST VECTOR and THETA SERIES is very slight. THETA SERIES requires $\|x\|^2 = k$ in a solution, while SHORTEST VECTOR requires only the inequality $\|x\|^2 \le k$.

Peter van Emde Boas [vEB80] proved in 1980 that NEAREST VECTOR is NP-complete, and conjectured that the SHORTEST VECTOR problem is also NP-complete. There has been a considerable amount of work devoted to the proof of this conjecture — see [ABSS93] and [Var97a] for a discussion. Ajtai [Ajt97] has recently proved that the SHORTEST VECTOR problem is hard for NP under randomized reductions. This comes very close to proving the conjecture of [vEB80]. Akin to the situation with linear codes, nothing is currently known regarding the parametrized complexity of the three problems discussed above.

Herein, we prove for the first time that the THETA SERIES problem is NP-hard. This seems to add even further weight to the conjecture of [vEB80]. Moreover, since our reduction is parametric, it also shows that THETA SERIES is hard for the parametrized complexity class $W[1]$. Along similar lines, we prove that the NEAREST VECTOR problem is also hard for $W[1]$.

Our results are based on a powerful combinatorial transformation that uses many of the ideas employed in the proofs of the main theorems in [DF95a, DF95b]. We feel that one of the most interesting aspects of our work is a demonstration of the potential utility of parametric methods and perspectives in addressing issues in "classical" complexity theory. We will assume that the reader has some familiarity with the parameterized complexity framework, such as can be found in [DF95a, DF95b, DF95c, DF97b], for example. For the benefit of readers that do not have this background, some discussion and essential definitions are presented in § 4 and the Appendix.

In the interests of readability, we defer the proof of our main combinatorial transformation to §3. In the next section, we prove the main results, using this transformation. In §4 we present an outline of the proof of membership in $W[2]$ for several of the problems considered in this paper. We furthermore discuss the remaining open problems, and speculate on whether the techniques used herein might be adapted to provide a proof of $W[1]$-hardness for Minimum Distance, or of NP-hardness for the Shortest Vector.

2. The main results through red/blue graphs

We start with some notation and terminology. Let $G = (V, E)$ be a graph. We say that two distinct vertices $u, v \in V$ are *neighbors* if they are adjacent in G, namely if $(u, v) \in E$. A set of vertices $V' \subseteq V$ is said to be a *perfect code* in G if every vertex of V is either contained in V' or has a *unique* neighbor in V', but not both. The starting point for our transformations is the parameterized problem of determining the existence of a k-element perfect code in a graph.

Problem: Perfect Code
Instance: A graph $G = (V, E)$ and an integer $k > 0$.
Question: Is there a k-element perfect code in G?
Parameter: k

This problem was shown to be hard for $W[1]$ in [DF95b]. Kratochvíl [KK88, Kra94] was the first to prove that this problem is NP-complete, several years earlier.

Our general approach in what follows is based on constructing and manipulating linear codes, and other sets of vectors, using bipartite graphs. Let $G = (\mathcal{R}, \mathcal{B}, E)$ be a bipartite graph with the partition of the vertices into the red set $\mathcal{R}$ and the blue set $\mathcal{B}$. We make the following definitions concerning special sets of red vertices in G.

Definition. *Suppose that $G = (\mathcal{R}, \mathcal{B}, E)$ is a red/blue bipartite graph, and let $R \subseteq \mathcal{R}$ be a a nonempty set of red vertices. We say that R is*

- *a* ***dominating set*** *if every vertex in $\mathcal{B}$ has at least one neighbor in R*
- *a* ***perfect code*** *if every vertex in $\mathcal{B}$ has a unique neighbor in R*
- *an* ***odd set*** *if every vertex in $\mathcal{B}$ has an odd number of neighbors in R*
- *an* ***even set*** *if every vertex in $\mathcal{B}$ has an even number of neighbors in R.*

Notice that what we define to be a perfect code is not the same for red/blue bipartite graphs and for general (uncolored) graphs. Similarly, our definition of a dominating set in a red/blue graph does not coincide with the conventional definition of dominating sets in general graphs. However, it will be always clear from the context which definition applies in each case.

We can now state the main problems concerning red/blue graphs that we consider in this paper.

Problem: Even Set
Instance: A red/blue graph $G = (\mathcal{R}, \mathcal{B}, E)$ and an integer $k > 0$.
Question: Is there a non-empty set of at most k vertices $R \subseteq \mathcal{R}$ that is an even set in G ?
Parameter: k

Problem: Exact Even Set
Instance: A red/blue graph $G = (\mathcal{R}, \mathcal{B}, E)$ and an integer $k > 0$.
Question: Is there a set of k vertices $R \subseteq \mathcal{R}$ that is an even set in G ?
Parameter: k

Problem: Odd Set
Instance: A red/blue graph $G = (\mathcal{R}, \mathcal{B}, E)$ and an integer $k > 0$.
Question: Is there a set of at most k vertices $R \subseteq \mathcal{R}$ that is an odd set in G ?
Parameter: k

Problem: Exact Odd Set
Instance: A red/blue graph $G = (\mathcal{R}, \mathcal{B}, E)$ and an integer $k > 0$.
Question: Is there a set of k vertices $R \subseteq \mathcal{R}$ that is an odd set in G ?
Parameter: k

We shall see shortly that all these problems are NP-complete. We will furthermore prove that Exact Even Set, Odd Set, and Exact Odd Set are hard for $W[1]$. The following theorem will serve as the main combinatorial engine in our proof.

Theorem 1. *Let G be a graph on n vertices, and let k be a positive integer. In time polynomial in n and k we can produce a red/blue bipartite graph G' and a positive integer k', such that:*

P1. *Every dominating set in G' has size at least k'.*

P2. *Every dominating set in G' of size k' is a perfect code in G'.*

P3. *There is a perfect code of size k in G if and only if there is a perfect code of size k' in G'.*

Notice that the red/blue graph G' encodes the information about the existence of a k-element perfect code in G. However, while the graph G is completely arbitrary, the red/blue graph G' has a substantial amount of useful structure, expressed by the properties **P1** and **P2**.

The theorem itself is established in the next section by means of a somewhat complicated graph-theoretic transformation, that has a general architecture similar to the one employed in proving the main theorems of [DF95a] and [DF95b]. In what follows, we will use Theorem 1 to yield significant results that illustrate the applicability of our graph-theoretic approach to parametrized problems concerning linear codes and integer lattices.

Parametrized complexity of problems concerning linear codes. First, note that we have the following merely by observing that in a red/blue bipartite graph, by definition, a perfect code is an odd set, and an odd set is necessarily a dominating set.

Theorem 2. ODD SET *and* EXACT ODD SET *are hard for* $W[1]$ *and* NP*-complete.*

Proof. Theorem 1 immediately implies a polynomial-time parameterized transformation from PERFECT CODE, which is a $W[1]$-hard and NP-complete problem, to ODD SET and to EXACT ODD SET. Given an instance G and k of PERFECT CODE, we construct G' and k' as in Theorem 1. It follows from properties **P1**–**P3** that G' contains an odd set of size at most k' if and only if G has a k-element perfect code, and the size of this odd set is, in fact, exactly k'. □

We now observe that ODD SET is very similar to MAXIMUM-LIKELIHOOD DECODING. This immediately leads to the following.

Theorem 3. MAXIMUM-LIKELIHOOD DECODING *is hard for* $W[1]$.

Proof. Given an instance $G = (\mathcal{R}, \mathcal{B}, E)$ and k of ODD SET, we construct an instance of MAXIMUM-LIKELIHOOD DECODING as follows. The binary $m \times n$ matrix $H = [h_{ij}]$ is the red/blue adjacency matrix of G, whose columns are indicators of the neighborhoods of vertices in $\mathcal{R}$. That is, w.l.o.g. we let $\mathcal{R} = \{1, 2, \ldots, n\}$ and $\mathcal{B} = \{1, 2, \ldots, m\}$, and then set $h_{ij} = 1$ if and only if $i \in \mathcal{B}$ is adjacent to $j \in \mathcal{R}$ in G. Thus G is a Tanner graph for the binary linear code having H as its parity-check matrix (cf. [Tan81, SS96]). We set the target vector s equal to the all-one vector $\mathbf{1}$ of length $m = |\mathcal{B}|$. It is easy to see that a set of k columns of H sums to $s = \mathbf{1}$ if and only if the corresponding k vertices of $\mathcal{R}$ constitute an odd set in G. □

We now employ Theorem 1 in a slightly more elaborate way, to establish the following.

Theorem 4. EXACT EVEN SET *is hard for* $W[1]$ *and* NP*-complete.*

Proof. We again reduce from PERFECT CODE. Given an instance G and k of PERFECT CODE, we first construct $G' = (\mathcal{R}, \mathcal{B}, E)$ and k' as in Theorem 1. Next, we let $G_1 = (\mathcal{R}_1, \mathcal{B}_1, E_1)$ and $G_2 = (\mathcal{R}_2, \mathcal{B}_2, E_2)$ denote two identical replicas of G'. We can combine G_1 and G_2 into a single red/blue graph H, by creating a new red vertex z and connecting it to all the vertices in $\mathcal{B}_1$ and $\mathcal{B}_2$. Finally, we obtain a red/blue graph H^* by adjoining to H a set of $|\mathcal{R}|$ blue vertices $\mathcal{B}^*$, one for each vertex of $\mathcal{R}$, and connecting them as follows: every $\beta \in \mathcal{B}^*$ is adjacent to one vertex in $\mathcal{R}_1$ and one vertex in $\mathcal{R}_2$ which correspond to the same vertex of $\mathcal{R}$, and every such pair of vertices in $\mathcal{R}_1$ and $\mathcal{R}_2$ is connected through some vertex of $\mathcal{B}^*$. The construction of H^* from G' is illustrated in Figure 1. The instance of EXACT EVEN SET is given by H^* and $2k' + 1$.

Now suppose that G has a k-element perfect code. Then by property **P3** of Theorem 1, the red/blue graph G' has a k'-element perfect code $R \subseteq \mathcal{R}$. It is straightforward to verify that the $2k'$ vertices corresponding to R in $\mathcal{R}_1$ and $\mathcal{R}_2$, together with the vertex z, constitute an even set of size $2k' + 1$ in H^*. Indeed, by construction, every blue vertex of $\mathcal{B}_1 \cup \mathcal{B}_2$ is adjacent to z and to exactly one vertex in the replica of R, either in $\mathcal{R}_1$ or in $\mathcal{R}_2$. Every blue vertex of $\mathcal{B}^*$ is either adjacent to both replicas of some vertex in R, or to none at all.

In the other direction, suppose that S is an even set of size $2k'+1$ in H^*. If S contains a vertex $\rho \in \mathcal{R}_1$, then it must also contain the corresponding vertex of $\mathcal{R}_2$, since otherwise the vertex of $\mathcal{B}^*$ adjacent to ρ will have exactly one neighbor in S. It follows that $|S \cap \mathcal{R}_1| = |S \cap \mathcal{R}_2|$, and the

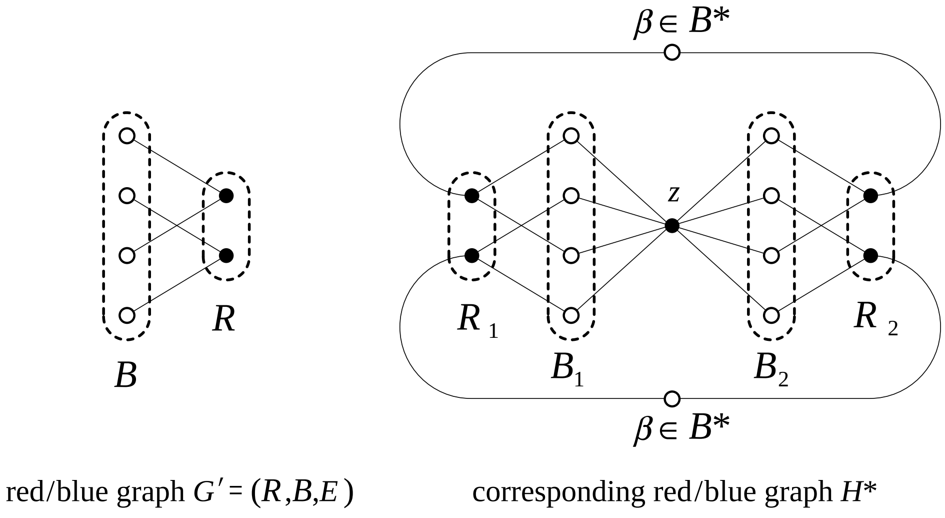
$\beta \in B^*$
B
R
z
R_1
B_1
B_2
R_2
$\beta \in B^*$
red/blue graph $G' = (R,B,E)$
corresponding red/blue graph H^*

Figure 1: *Construction of H^* from G'*

vertices of S are paired in this way. Since the size of S is odd, we conclude that S must contain the vertex z. This further implies that $S \cap \mathcal{R}_1$ is a dominating set in G_1, as otherwise there is a vertex $\beta \in \mathcal{B}_1$ which has the single neighbor z in S. Since $|S| = 2k'+1$ and $|S \cap \mathcal{R}_1| = |S \cap \mathcal{R}_2|$, the size of this dominating set is exactly k'. By property **P2** of Theorem 1, this dominating set is a perfect code in $G_1 = G'$, and property **P3** implies that G has a perfect code of size k. $\square$

It is easy to see that Exact Even Set and Weight Distribution are essentially different ways to phrase one and the same problem. Thus we have

Theorem 5. Weight Distribution *is hard for $W[1]$.*

Proof. This follows directly from Theorem 4, by taking the matrix H in Weight Distribution to be the red/blue adjacency matrix for the graph $G = (\mathcal{R}, \mathcal{B}, E)$ in Exact Even Set, in the same way as it was done in the proof of Theorem 3. $\square$

Complexity of bounded-distance decoding. The fact that Maximum-Likelihood Decoding is hard for $W[1]$, established in Theorem 3, implies hardness for bounded-distance decoding of binary linear codes, in a certain sense. We now explain this implication.

Maximum-likelihood decoding is a nearest neighbor search in the vector space $\mathbb{F}_2^n$ endowed with the Hamming distance $d(x,y)$ = number of positions where $x \in \mathbb{F}_2^n$ and $y \in \mathbb{F}_2^n$ differ. Let H be an $m \times n$ binary matrix, and let $\mathbb{C}$ be the binary linear code defined by H, that is $\mathbb{C} = \{x \in \mathbb{F}_2^n \ : \ Hx^t = \mathbf{0}\}$. Then for all $y \in \mathbb{F}_2^n$, a *maximum-likelihood decoder* for $\mathbb{C}$ always finds the closest codeword, that is $x \in \mathbb{C}$ such that $d(x,y)$ is the minimum possible. If $s = Hy^t$, this is equivalent to finding the smallest set of columns of H that sums to s; hence the corresponding decision problem is precisely Maximum-Likelihood Decoding.

While the complexity of maximum-likelihood decoding has been thoroughly studied [ABSS93, Bar94, BMvT78, BN90, Ste93], almost nothing is presently known regarding the complexity of bounded-distance decoding, even though most of the decoders used in practice are bounded-distance decoders. A decoder is said to be *bounded-distance* if there exists a constant $t > 0$ such that for all $y \in \mathbb{F}_2^n$, the decoder always finds the closest codeword $x \in \mathbb{C}$, provided $d(x,y) \le t$. Formally, for each positive integer t, we define the problem

Problem: BDD(t)
Instance: A binary $m \times n$ matrix H, a target vector $s \in \mathbb{F}_2^m$, and an integer $k \le t$.
Question: Is there a set of at most k columns of H that sum to s ?

Notice that BDD(t) is trivially in P for all t, since it can be solved in time $O(n^t)$ by simply computing Hx^t for every vector x in a Hamming sphere of radius t. Hence, the complexity of bounded-distance decoding has to be studied in a different framework. In particular, we would like to have an algorithm that solves BDD(t) in time $O(n^c)$, where c is a constant independent of t. That is, the multiplicative constant in $O(\cdot)$ may depend on t, but not the exponent.

The following corollary to $W[1]$-hardness of Maximum-Likelihood Decoding shows that such an algorithm does not exist, unless the W complexity hierarchy collapses with $W[1] = \text{FPT}$.

Theorem 6. *Suppose that $W[1] \ne \text{FPT}$. Then for any positive constant c, there exists an integer t_0, such that* BDD(t) *is not solvable in time $O(n^c)$ for all $t \ge t_0$, even if the multiplicative constant in $O(n^c)$ may depend on t.*

Proof. First observe that the claim of the theorem is equivalent to the following: for any positive constant c, there exists an integer t_0, such that BDD(t_0) is not solvable in time $O(n^c)$. This is so because if $t \ge t_0$, then the set of possible instances of BDD(t_0) is a subset of the set of possible instances of BDD(t). Hence, if BDD(t_0) cannot be solved in time $O(n^c)$, then neither can BDD(t) for all $t \ge t_0$.

Now, assume to the contrary that for some c, we can solve BDD(t) in time $O(n^c)$ for all t. Given an instance of Maximum-Likelihood Decoding, we set $t = k$ and query an oracle for BDD(t), which provides an answer to the question of Maximum-Likelihood Decoding in time $O(n^c)$. The constant in $O(n^c)$ may depend on $t = k$, let us denote this constant by a_k. Setting $f(k) = a_k$, we see that Maximum-Likelihood Decoding is solvable in time $f(k)n^c$. Hence, it is fixed-parameter tractable, which is possible only if $W[1] = \text{FPT}$ in view of Theorem 3. $\square$

Parametrized complexity of problems concerning integer lattices. The combinatorial transformation in Theorem 1 can be also used to show both NP-completeness and $W[1]$-hardness for Nearest Vector and Theta Series. We start with the Nearest Vector problem.

Theorem 7. Nearest Vector *is hard for $W[1]$.*

Proof. Given an instance G and k of Perfect Code, we again construct $G' = (\mathcal{R}, \mathcal{B}, E)$ and k' as in Theorem 1. W.l.o.g., let $\mathcal{R} = \{1, 2, \ldots, r\}$ and $\mathcal{B} = \{1, 2, \ldots, b\}$, and let H be the $r \times b$ blue/red adjacency matrix of G', which is the transpose of the matrix constructed in Theorem 3. That is, the *rows* of H are now indicators of the neighborhoods of the vertices in $\mathcal{R}$. If c is a real constant and $A = [a_{ij}]$ is an integer matrix, we let $cA = [ca_{ij}]$ denote the matrix obtained by multiplying each entry of A by c. We choose c to be a large integer, so that $c > k'$, and construct an instance of Nearest Vector as follows. First, we define a $(b+r) \times (b+r)$ integer matrix

$$M = \left[\begin{array}{c|c} cH & I_r \\ \hline 2cI_b & \mathbf{0} \end{array}\right] \tag{1}$$

where I_r is the $r \times r$ identity matrix, and $\mathbf{0}$ stands for the $b \times r$ all-zero matrix. It is easy to see that M has $n = r + b$ linearly independent rows, which constitute a basis for a sublattice Λ of $\mathbb{Z}^n$. We take the target vector as $s = (c, c, \ldots, c, 0, 0, \ldots, 0) \in \mathbb{Z}^n$ so that the first b entries of s are equal to c, while the last r entries are equal to zero. Then Λ, s, and k' is the instance of Nearest Vector, to which the instance G and k of Perfect Code is transformed.

If there is a perfect code of size k in G, then by property **P3** of Theorem 1 there is a perfect code $R \subseteq \mathcal{R}$ of size k' in G'. For convenience, we let M' denote the $r \times (b+r)$ matrix consisting of the first r rows of M. Thus each row of M' is naturally indexed by a unique vertex of $\mathcal{R}$. Let $x = (x_1, x_2, \ldots, x_n) \in \Lambda$ be the sum of those k' rows of M' that are indexed by the vertices of the perfect code R. Then it is easy to see that $x_i = c$ in the first b positions, and $\|x - s\|^2 = k'$.

In the other direction, suppose that there is a vector $x = (x_1, x_2, \ldots, x_n) \in \Lambda$ with $\|x - s\|^2 \le k'$, and denote $y = (y_1, y_2, \ldots, y_n) = x - s$. First observe that $y_i \equiv 0 \bmod c$ for $i = 1, 2, \ldots, b$, by the construction of M and s. Since $c > k'$ while $\|y\|^2 \le k'$, it follows that $y_i = 0$ in the first b positions. This further implies that $x_i = c$ for $i = 1, 2, \ldots, b$. Now let R be the subset of $\mathcal{R}$ consisting of all the vertices that index those rows of M' that are included in the linear combination comprising x (with a non-zero coefficient). We claim that R is a dominating set in G'. Indeed, if some vertex $i \in \mathcal{B}$ does not have a neighbor in R, then $x_i \equiv 0 \bmod 2c$ in the corresponding position, contrary to the fact that $x_i = c$ which we have already established. Hence, by property **P1** of Theorem 1, we have that $|R| \ge k'$. Together with $\|x - s\|^2 \le k'$ this implies that $\|x - s\|^2 = |R| = k'$, and R is a perfect code in G' by property **P2** of Theorem 1. Property **P3** of Theorem 1 now implies that G has a perfect code of size k. □

By modifying the argument of Theorem 7 only slightly, we can prove that Theta Series is also hard for $W[1]$. The idea is to incorporate the target vector s in the generator matrix M for Λ. The same argument furthermore shows, for the first time, that Theta Series is NP-complete.

Theorem 8. Theta Series *is hard for* $W[1]$ *and* NP-*complete.*

Proof. We proceed as in the proof of Theorem 7, except that we now choose c so that $c > 4k' + 1$, and replace the matrix in (1) by the slightly more elaborate $(b+r+1) \times (b+r+1)$ matrix

$$M = \left[\begin{array}{c|c|c} cH & \mathbf{0} & 2I_r \\ \hline c^2 I_b & \mathbf{0} & \mathbf{0} \\ \hline c\,c \cdots c & 1 & 0\,0 \cdots 0 \end{array}\right] \tag{2}$$

where $\mathbf{0}$ is used to denote both the all-zero column and the $b \times r$ all-zero matrix. We again think of the $n = r + b + 1$ rows of M as a basis for a sublattice Λ of $\mathbb{Z}^n$. Thus an instance G and k of Perfect Code is transformed by Theorem 1 into the red/blue graph $G' = (\mathcal{R}, \mathcal{B}, E)$ and k', which is further transformed into the instance Λ and $4k' + 1$ of Theta Series.

Suppose that G contains a k-element perfect code, and let $R \subseteq \mathcal{R}$ be the corresponding k'-element perfect code in G'. We again let M' denote the $r \times (b+r+1)$ submatrix of M consisting of the first r rows that are naturally indexed by the vertices of $\mathcal{R}$. Let $x = (x_1, x_2, \ldots, x_n) \in \Lambda$ be the following linear combination of $k'+1$ rows of M: k' rows of M' indexed by the k' vertices of the perfect code R, with coefficient $+1$, and the last row of M with coefficient -1. Then it is easy to see that $x_i = 0$ in the first b positions, and $\|x\|^2 = 4k' + 1$.

In the other direction, suppose that there exists $x = (x_1, x_2, \ldots, x_n) \in \Lambda$ with $\|x\|^2 = 4k' + 1$. Write $x = a_1 v_1 + a_2 v_2 + \cdots + a_n v_n$, where $a_1, a_2, \ldots, a_n$ are integers, not all zero, and $v_1, v_2, \ldots, v_n$ are the rows of M, listed in a top-to-bottom order. First, we again observe that $x_i \equiv 0 \bmod c$ for $i = 1, 2, \ldots, b$, by the construction of M. Since $c > 4k' + 1$, it follows that $x_i = 0$ in the first b positions. Further observe that $x_i \equiv 0 \bmod 2$ in the last r positions. Together with $\|x\|^2 \equiv 1 \bmod 4$, this implies that $x_{b+1} = a_n$ must be non-zero. Notice that

$$a_n^2 \;=\; x_{b+1}^2 \;\le\; \|x\|^2 \;=\; 4k' + 1 \;<\; c \tag{3}$$

As in the proof of Theorem 7, let R be the subset of $\mathcal{R}$ consisting of the vertices that index those rows of M' which have a non-zero coefficient in the linear combination comprising x. We again claim that R is a dominating set in G'. Otherwise, let $i \in \mathcal{B}$ be a vertex which does not have a neighbor in R, and let $y = x - a_n v_n$. We have already shown that $x_i = 0$. Hence $y_i = -a_n c \neq 0$ and $|y_i| < c^2$ in view of (3). This is a contradiction, since if i does not have neighbors in R, then $y_i \equiv 0 \bmod c^2$ by the construction of M. From this point on, the proof is exactly the same as in Theorem 7. Referring to Theorem 1, we conclude that $|R| \geq k'$, which together with $\|x\|^2 = 4k' + 1$ implies that $|R| = k'$ and R is necessarily a perfect code in G'. This, in turn, further implies the existence of a k-element perfect code in G. $\square$

As a final remark in this section, we observe that a theorem of Cai and Chen [CC93] states that if the optimization problem naturally associated to an integer-parameter problem has a fully polynomial-time approximation scheme (see [GJ79] for an exposition of this concept), then the corresponding parameterized problem is fixed-parameter tractable. We can draw from this the following corollary.

Corollary 9. *There is no fully polynomial-time approximation scheme for any of the problems discussed in this section, unless* $W[1] = \mathrm{FPT}$.

In the next section, we prove our principal combinatorial transformation (Theorem 1), which served us so well in this section.

3. The combinatorial engine

Our proof of Theorem 1 has many similarities with the proofs of the main theorems of [DF95a] and [DF95b], to which the reader may wish to refer. Notice that an implicit assumption in Theorem 1 is that the constant k' may depend on k, but not on n. This assumption is essential for all the parametrized transformations in the previous section (for more on this, see the Appendix). We now restate Theorem 1, while specifying a precise value for k' in terms of k.

Theorem 1. *Let $G = (V, E)$ be a graph on n vertices, and let k be a positive integer. In time polynomial in n and k we can produce a red/blue bipartite graph $G' = (\mathcal{R}, \mathcal{B}, E')$, and the positive integer*

$$k' \;=\; (2k+1) + 3\Big(2k(k+1) + (k+1)^2\Big)$$

such that:

P1. *Every dominating set in G' has size at least k'.*

P2. *Every dominating set in G' of size k' is a perfect code in G'.*

P3. *There is a perfect code of size k in G if and only if there is a perfect code of size k' in G'.*

We start the proof with some notation. In describing the construction of G' from G it is convenient to identify V with the set of integers $\{1, \ldots, n\}$. An *interval* of vertices is defined to be a subset of V consisting of consecutive integers, for example $\{3, 4, 5, 6\}$, and may be empty. Let $\mathcal{J}$ denote the set of all nonempty intervals having a size of at most $n - k$, that is:

$$\mathcal{J} \;=\; \{\; J \;:\; J \text{ is an interval in } V \text{ and } \; 1 \;\le\; |J| \;\le\; n-k \;\}$$

For an interval $J \in \mathcal{J}$, define the *initial boundary* $\partial(J)$ of J to be the largest non-negative integer strictly less than the smallest element of J. For example $\partial(\{2,3,4\}) = 1$, $\partial(\{6\}) = 5$, $\partial(\{1,2\}) = 0$. Define the *terminal boundary* $\partial'(J)$ to be the smallest positive integer strictly greater than the largest element of J. Thus $\partial'(\{2,3,4\}) = 5$, for example.

We will also need to refer to empty intervals, but we will still need to indicate where these empty intervals begin and end. For this purpose, we introduce the special symbols $\epsilon_0, \epsilon_1, \ldots, \epsilon_n$ and extend the definitions of ∂ and ∂' as follows, For $u = 0, 1, \ldots, n$, we define $\partial(\epsilon_u) = u$ and $\partial'(\epsilon_u) = u + 1$. Thus ϵ_u represents the "interval" of vertices in V that is empty, but "located" between u and $u + 1$. We let $\mathcal{J}^*$ denote the set of nonempty intervals $\mathcal{J}$ augmented with these empty intervals, that is $\mathcal{J}^* = \mathcal{J} \cup \{\, \epsilon_u \;:\; 0 \le u \le n \,\}$.

If $u, v \in V$ with $u < v$, we define the interval *between* u and v as $J(u,v) = \{u+1, \ldots, v-1\}$, provided $v - u \ge 2$. If $v = u + 1$ then $J(u,v) = \epsilon_u$. Similarly, for $u \in V$ we define the interval *preceding* u to be $J(0,u) = \{1, \ldots, u-1\}$ if $u \ge 2$, and if $u = 1$ then $J(0,u) = J(0,1) = \epsilon_0$. The interval *succeeding* $u \in V$ is defined as $J(u,\infty) = \{u+1, \ldots, n\}$ if $u < n$, and $J(n,\infty) = \epsilon_n$.

Let $[k]$ denote the set of integers $\{1, 2, \ldots, k\}$, and let $[k]^*$ denote the set of integers $\{0, 1, \ldots, k\}$. Part of our construction of $G' = (\mathcal{R}, \mathcal{B}, E')$ will be quantified over the set $[k]^* \times [k]^*$ of ordered pairs of elements of $[k]^*$, while other parts of the construction will be quantified over subsets of this set, namely $[k] \times [k]^*$ and $[k]^* \times [k]$. This distinction between $[k]$ and $[k]^*$ is basically a technicality, which is needed to account for the boundary cases in the construction.

The red foundation. Our description of G' starts with five sets of red vertices $R_1, R_2, \dots, R_5$. We will refer to the vertices of $R_1, R_2, \dots, R_5$ as *basic* red vertices. All the blue vertices, as well as some additional red vertices, will be added to G' as the construction progresses. The five sets $R_1, R_2, \dots, R_5$ are employed in our construction to represent the structure of each possible choice of a k-element subset of the set of vertices of G. Each one of the five sets is, in a sense, a gadget designed to capture a different aspect of this structure. We now describe these sets, along with their roles in the construction of G' and the notation used to refer to their vertices:

- Gadgets that indicate the k chosen vertices:
$$R_1 \;=\; \{\, a(i,u) \;:\; i \in [k],\; u \in V \,\}$$
- Gadgets that indicate the intervals between chosen vertices:
$$R_2 \;=\; \{\, b(i,J) \;:\; i \in [k]^*,\; J \in \mathcal{J}^* \,\}$$
- Gadgets that indicate pairs of intervals:
$$R_3 \;=\; \{\, c(i,i',J,J') \;:\; i,i' \in [k]^*,\; J,J' \in \mathcal{J}^* \,\}$$
- Gadgets that indicate choice/interval pairs:
$$R_4 \;=\; \{\, c'(i,i',u,J) \;:\; i \in [k],\; i' \in [k]^*,\; u \in V,\; J \in \mathcal{J}^* \,\}$$
- Gadgets that indicate interval/choice pairs:
$$R_5 \;=\; \{\, c''(i,i',J,u) \;:\; i \in [k]^*,\; i' \in [k],\; J \in \mathcal{J}^*,\; u \in V \,\}$$

It will be convenient to organize the basic red vertices into *blocks*. These blocks will constitute a partition of the basic red vertices, and there will be altogether
$$k'' \;=\; 1 + 2k + 2k(k+1) + (k+1)^2 \tag{4}$$
blocks. Specifically, we partition R_1 into k blocks, R_2 into $k+1$ blocks, R_3 into $(k+1)^2$ blocks, and R_4, R_5 into $k(k+1)$ blocks each. These blocks are defined as follows.

$$\begin{aligned}
\mathcal{A}(i) &= \{a(i,u) : i \in [k],\; u \in V\} && \text{for } i = 1,2,\dots,k \\
\mathcal{B}(i) &= \{b(i,J) : i \in [k]^*,\; J \in \mathcal{J}^*\} && \text{for } i = 0,1\dots,k \\
\mathcal{C}(i,i') &= \{c(i,i',J,J') : J,J' \in \mathcal{J}^*\} && \text{for } i \in [k]^* \text{ and } i' \in [k]^* \\
\mathcal{C}'(i,i') &= \{c'(i,i',u,J) : u \in V,\; J \in \mathcal{J}^*\} && \text{for } i \in [k] \text{ and } i' \in [k]^* \\
\mathcal{C}''(i,i') &= \{c''(i,i',J,u) : J \in \mathcal{J}^*,\; u \in V\} && \text{for } i \in [k]^* \text{ and } i' \in [k]
\end{aligned}$$

We let $\mathfrak{R}_0$ denote the set of k'' red blocks defined above. These will be referred to as the *basic blocks*. Additional blocks of blue and red vertices will be added to G' later in the construction.

The semantics of the reduction. Our semantic intentions in the design of $G' = (\mathcal{R}, \mathcal{B}, E')$ can be summarized as follows. The k blocks of R_1 represent the choice of k vertices of V that may form a k-element perfect code in G. The $k+1$ blocks of R_2 represent the intervals between the consecutive choices of vertices of V represented in R_1. The blocks of R_3, R_4, R_5 do not carry any meaning with respect to G; these blocks are needed to impose an internal structure on G'.

Although we have only started to describe G', enough is already visible to enable us describe, at least in part, the solution translations of the reduction. This description may serve to motivate and clarify some of the forthcoming details of the proof. There are two different solution translations, namely translations in the

forward direction: the manner in which a k-element perfect code in G translates into a k'-element perfect code in G';

backward direction: the manner in which a k'-element perfect code (or dominating set) in G' translates into a k-element perfect code in G.

The forward solution translation. In the forward direction, suppose that the elements of a perfect code of size k in $G = (V, E)$ are given by $u_1, u_2, \ldots, u_k$, and w.l.o.g. assume that

$$u_1 \;<\; u_2 \;<\; \cdots \;<\; u_k$$

Let $J_0 = J(0, u_1)$. In other words, J_0 is the interval (set) of elements of V that precede u_1. For $i = 1, 2, \ldots, k-1$, let $J_i = J(u_i, u_{i+1})$. Thus J_i is the interval of elements of V that are properly between u_i and u_{i+1}. Finally let $J_k = J(u_k, \infty)$ be the interval of elements of V following u_k. Now consider the sets:

$$S_1 \;=\; \{\, a(i, u_i) \;:\; i \in [k] \,\} \tag{5}$$

$$S_2 \;=\; \{\, b(i, J_i) \;:\; i \in [k]^* \} \tag{6}$$

$$S_3 \;=\; \{\, c(i, i', J_i, J_{i'}) \;:\; i \in [k]^*, i' \in [k]^* \} \tag{7}$$

$$S_4 \;=\; \{\, c'(i, i', u_i, J_{i'}) \;:\; i \in [k],\; i' \in [k]^* \} \tag{8}$$

$$S_5 \;=\; \{\, c''(i, i', J_i, u_{i'}) \;:\; i \in [k]^*,\; i' \in [k] \,\} \tag{9}$$

and let $S = S_1 \cup S_2 \cup \cdots \cup S_5$. Notice that S contains precisely one vertex from each of the k'' basic red blocks, where k'' is given by (4). Our construction of $G' = (\mathcal{R}, \mathcal{B}, E')$ will ensure that the set S may be extended to a perfect code in G'. This extension is accomplished by adding two more vertices for each vertex in S_3, S_4, and S_5, as specified later in our construction.

The backward solution translation. Our construction of G' will also ensure that any k'-element dominating set S in G' must be distributed so that there is exactly one element of S in each of the k'' basic blocks of $\mathfrak{R}_0$. Furthermore, S will be forced to have the restricted form of a perfect code in G'. The construction will ensure that for such a set S, the u-indices of the k elements of $S \cap R_1$ are distinct, and that these indices correspond to a k-element perfect code in G.

The construction. We will describe $G' = (\mathcal{R}, \mathcal{B}, E')$ by starting with the k'' basic red blocks of $\mathfrak{R}_0$, and applying various *operators* to these blocks. Each application of an operator results in further blocks of red and blue vertices being created, along with various edges. A high-level blueprint for $G' = (\mathcal{R}, \mathcal{B}, E')$ in terms of these operators is shown in Figure 2.

In constructing $G' = (\mathcal{R}, \mathcal{B}, E')$, operators are only applied to argument sets of red blocks. Thus each blue vertex $\beta \in \mathcal{B}$ is created by a single specific application of an operator, and the neighborhood of β is completely established by this application. This allows us to argue a series of claims concerning properties of G', as we continue to describe the steps of the construction.

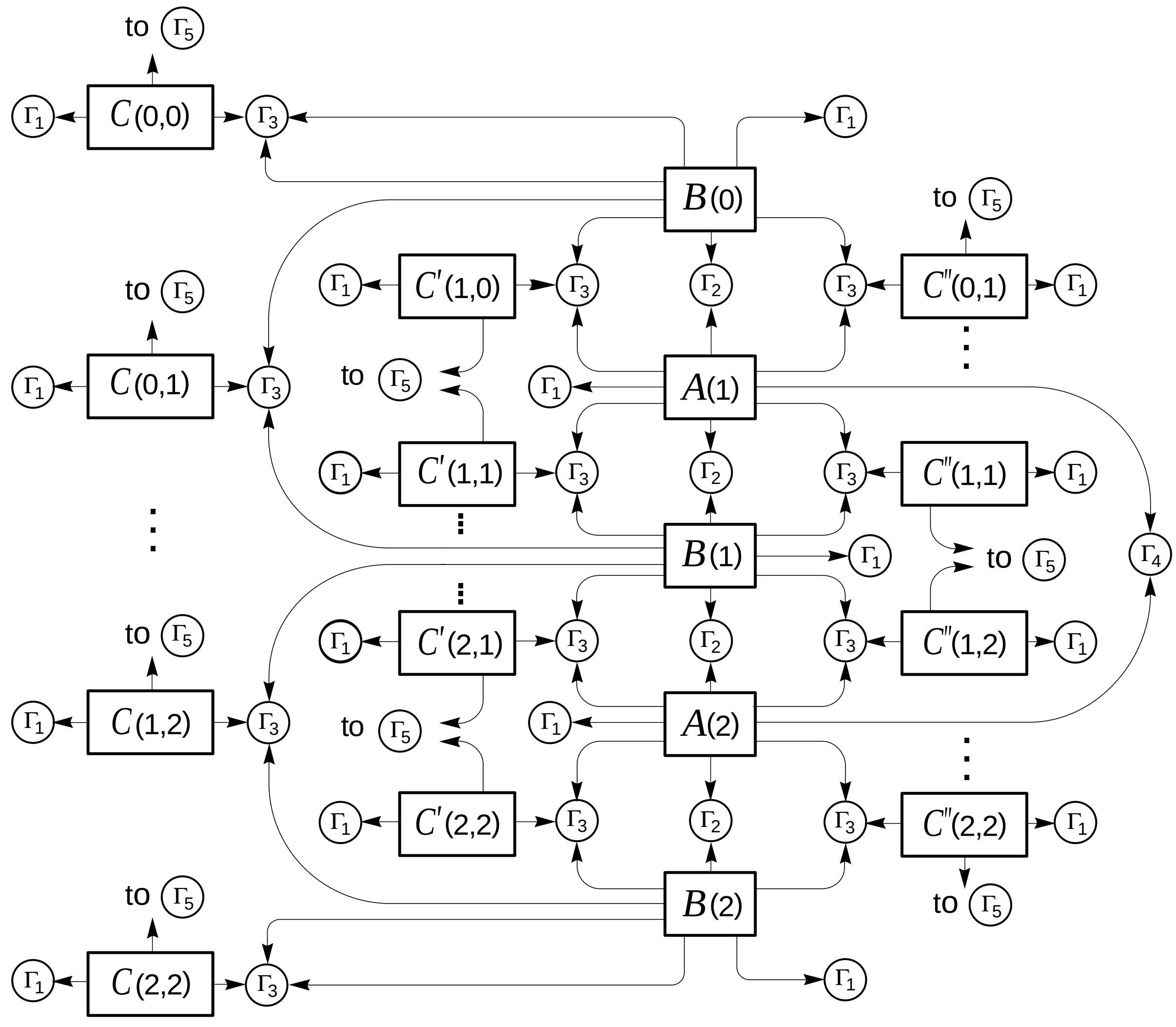

Figure 2: *The blueprint for G' in terms of basic blocks and operators (example for $k = 2$)*

The block guard operator Γ_1. The operator Γ_1 takes as an argument a single red block $\mathcal{X}$ and adds one blue vertex connected to every red vertex in $\mathcal{X}$.

The last step of the construction of $G' = (\mathcal{R}, \mathcal{B}, E')$ will be to apply this operator to each of the red blocks of the construction. When we get to the last step, there will be k' red blocks, the collection of which will be denoted $\mathfrak{R}$. Thus, we have

The last step: *Apply the block guard operator Γ_1 to every red block of the construction.*

We cannot make this the first step of the construction, as some of the k' red blocks to which Γ_1 is to be applied have yet to be created by applications of operators in earlier steps of the construction. However, Γ_1 does provide a simple initial example of an operator (the action of the block guard operator is illustrated in Figure 3), and we can easily prove certain important properties of $G' = (\mathcal{R}, \mathcal{B}, E')$ having only this much information about the construction.

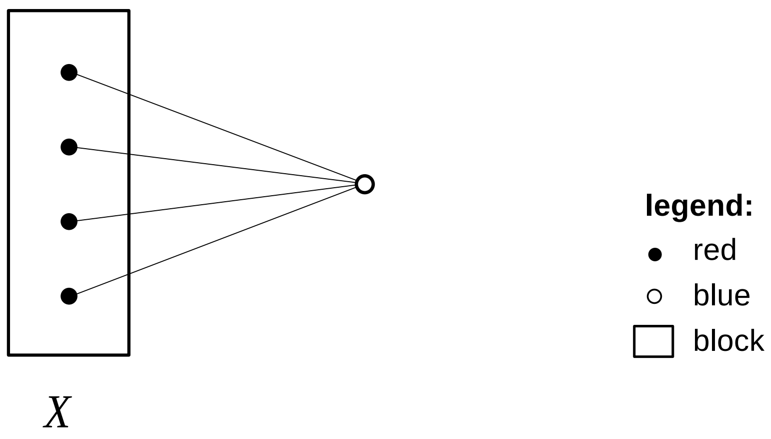

Figure 3: *The block guard operator* Γ_1

For example, the following lemma follows directly from the definition of Γ_1. This simple lemma already establishes property **P1** of Theorem 1.

Lemma 1. *Every dominating set in $G' = (\mathcal{R}, \mathcal{B}, E')$ has size at least k', and contains at least one vertex from each of the k' red blocks of $\mathfrak{R}$.*

Proof. Let $S \subseteq \mathcal{R}$ be a dominating set in G', and let $\mathcal{X} \in \mathfrak{R}$ be a red block. If $S \cap \mathcal{X} = \varnothing$, then the blue vertex $\beta \in \mathcal{B}$ created by the application of Γ_1 to $\mathcal{X}$ does not have a neighbor in S. □

Next, we need a definition. Let Γ be an operator; let A denote the union of the red blocks that either provide the arguments for the application of Γ, or that are created by the application of Γ, and let B denote the set of blue vertices created by the application of Γ.

Definition. *We say that the operator Γ is **locally perfect** if the following condition is satisfied for every subset S of A that contains exactly one vertex from each red block contained in A: if every vertex in B has at least one neighbor in S, then every vertex in B has a unique neighbor in S. In other words, if S contains one vertex from each red block and is "locally" a dominating set, then S is necessarily a "local" perfect code.*

The above definition is motivated by the observation that if every application of an operator in the construction of $G' = (\mathcal{R}, \mathcal{B}, E')$ is locally perfect, then property **P2** of Theorem 1 will hold. This observation is a corollary to the following lemma.

Lemma 2. *Every k'-element dominating set S in G' must contain exactly one vertex from each of the k' red blocks of $\mathfrak{R}$, and the last step in the construction of G' is locally perfect.*

Proof. Since each of the k' blocks of $\mathfrak{R}$ must contain at least one element of S by Lemma 1, each block must contain exactly one element of S. Local perfection of Γ_1 is trivial. □

The branch operator Γ_2. The arguments to the Γ_2 operator are two red blocks $\mathcal{X}$ and $\mathcal{X}'$, given together with a partition of $\mathcal{X}'$ into $|\mathcal{X}|$ or more non-empty classes, and an injective assignment to each element of $\mathcal{X}$ of a different class in the partition of $\mathcal{X}'$, that is

$$x \in \mathcal{X} \;\mapsto\; \mathcal{S}_x \subset \mathcal{X}'$$

so that $\mathcal{S}_x \cap \mathcal{S}_y = \varnothing$ for $x \neq y$. The result of applying Γ_2 to $\mathcal{X}$ and $\mathcal{X}'$ with this assignment is:

a. The creation of the set of $|\mathcal{X}|$ blue vertices

$$\{\, \beta(x) \;:\; x \in \mathcal{X} \,\}$$

b. For each blue vertex $\beta(x)$ in this set, the creation of edges connecting $\beta(x)$ to $x \in \mathcal{X}$ and to all the red vertices in $\mathcal{X}'$ that belong to $\mathcal{S}_y$ for some $y \neq x$ in $\mathcal{X}$.

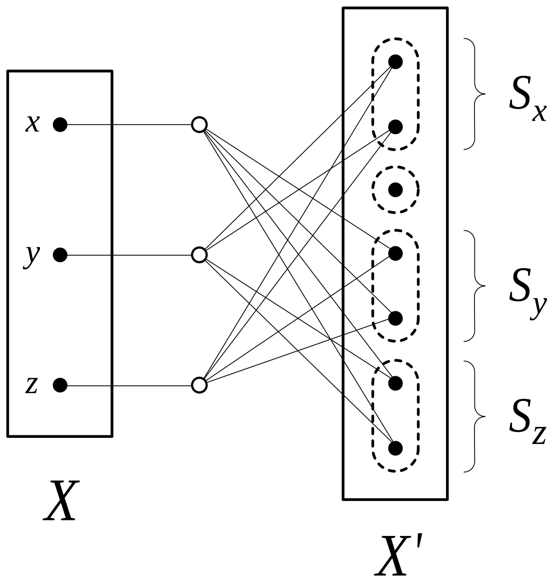

Figure 4: *The branch operator* Γ_2

Lemma 3. *The branch operator* Γ_2 *is locally perfect. In particular, suppose that this operator is applied to the red blocks* $\mathcal{X}$ *and* $\mathcal{X}'$*, and let* S *be a* k'*-element dominating set in* $G' = (\mathcal{R}, \mathcal{B}, E')$*. Then* $S \cap \mathcal{X} = \{x\}$*, and* $S \cap \mathcal{X}' = \{x'\}$ *for some* $x' \in \mathcal{S}_x$*.*

Proof. The fact that S intersects each of the blocks $\mathcal{X}$ and $\mathcal{X}'$ at a single vertex, say x and x', follows immediately from Lemma 2. The vertex $x \in \mathcal{X}$ is adjacent to the single blue vertex $\beta(x)$. If $x' \in \mathcal{X}'$ belongs to a partition class that is not assigned to a vertex of $\mathcal{X}$, then x' is not adjacent to any of the blue vertices created by Γ_2, and the set S dominates only the single vertex $\beta(x)$, a contradiction. Similarly, if $x' \in \mathcal{S}_y$ for some vertex $y \neq x$ in $\mathcal{X}$, then the blue vertex $\beta(y)$ is not adjacent to either x or x'. Hence $\beta(y)$ is not dominated by S, again a contradiction. Thus $x' \in \mathcal{S}_x$, which is the only remaining case. It now follows that every blue vertex created by Γ_2 has a unique neighbor in S. The unique neighbor of $\beta(x)$ is x, and x' is the unique neighbor of every other blue vertex. Thus, the branch operator Γ_2 is locally perfect. $\square$

A useful special case of the application of Γ_2 is as "equality" enforcer. For this, we assume that the arguments to Γ_2 are isomorphic sets, with the isomorphism given by the correspondence:

$$x \in \mathcal{X} \;\longleftrightarrow\; e(x) \in \mathcal{X}'$$

and in applying the operator Γ_2, we use the natural assignment $x \;\mapsto \mathcal{S}_x = \{e(x)\}$. In this situation, assuming that S is a k'-element dominating set in G', we observe that $S \cap \mathcal{X} = \{x\}$ and $S \cap \mathcal{X}' = \{y\}$ necessarily implies that $y = e(x)$, in view of Lemma 3. With Γ_2 at hand, we are finally ready to describe the first two steps of our construction.

Step 1. *For $i = 1, 2, \ldots, k-1$, apply the operator Γ_2 to the arguments $\mathcal{X} = \mathcal{A}(i)$ and $\mathcal{X}' = \mathcal{B}(i)$, with the assignment:*

$$a(i,u) \;\mapsto\; \{\, b(i,J) \;:\; J \in \mathcal{J}^*,\; \partial(J) = u \,\} \tag{10}$$

Also apply Γ_2 to the arguments $\mathcal{X} = \mathcal{A}(k)$ and $\mathcal{X}' = \mathcal{B}(k)$, with the assignment:

$$a(k,u) \;\mapsto\; \{\, b(k,J) \;:\; J \in \mathcal{J}^*,\; \partial(J) = u \;\text{ and }\; \partial'(J) = \infty \,\} \tag{11}$$

Step 2. *For $i = 2, 3, \ldots, k$, apply the operator Γ_2 to the arguments $\mathcal{X} = \mathcal{A}(i)$ and $\mathcal{X}' = \mathcal{B}(i-1)$, with the assignment:*

$$a(i,u) \;\mapsto\; \{\, b(i-1,J) \;:\; J \in \mathcal{J}^*,\; \partial'(J) = u \,\} \tag{12}$$

Also apply Γ_2 to the arguments $\mathcal{X} = \mathcal{A}(1)$ and $\mathcal{X}' = \mathcal{B}(0)$, with the assignment:

$$a(1,u) \;\mapsto\; \{\, b(0,J) \;:\; J \in \mathcal{J}^*,\; \partial'(J) = u \;\text{ and }\; \partial(J) = 0 \,\} \tag{13}$$

The first two steps of the construction operate on the vertices in R_1 and R_2 that represent, respectively, the choice of k vertices of G, and the intervals between these chosen vertices. The purpose of the these two steps is reflected in the following lemma.

Lemma 4. *Suppose that S is a k'-element dominating set in $G' = (\mathcal{R}, \mathcal{B}, E')$. Then the $k+(k+1)$ vertices of $S \cap R_1$ and $S \cap R_2$ consistently represent k vertices of $G = (V, E)$ and the intervals between these vertices, in the following way:*

- **a.** *If $S \cap \mathcal{A}(1) = \{a(1,u)\}$, then $S \cap \mathcal{B}(0) = \{b(0, J(0,u))\}$.*
- **b.** *For $i = 1, 2, \ldots, k-1$, if $S \cap \mathcal{A}(i) = \{a(i,u)\}$ and $S \cap \mathcal{A}(i+1) = \{a(i+1,v)\}$, then $S \cap \mathcal{B}(i) = \{b(i, J(u,v))\}$.*
- **c.** *If $S \cap \mathcal{A}(k) = \{a(k,u)\}$, then $S \cap \mathcal{B}(k) = \{b(k, J(u,\infty))\}$.*
- **d.** *For all $i < i'$, if $S \cap \mathcal{A}(i) = \{a(i,u)\}$ and $S \cap \mathcal{A}(i') = \{a(i',u')\}$, then $u < u'$.*

Proof. Referring to the last application of the operator Γ_2 in Step 2, it follows from Lemma 3 that if $S \cap \mathcal{A}(1) = \{a(1,u)\}$, then $S \cap \mathcal{B}(0)$ consists of a vertex x' which belongs to the set assigned to $a(1,u)$ on the righ-hand side of (13). But this set consists of the single element $b(0, J(0,u))$, which establishes part (**a**). Part (**c**) follows in a similar fashion from Lemma 3 and (11). Part (**b**) follows from Lemma 3 along with the combination of (10) and (12). Indeed, assuming the condition of part (**b**), it follows from Lemma 3 that $S \cap \mathcal{B}(i) = \{x'\}$, where x' belongs to the intersection of the set assigned to $a(i,u)$ in (10) with the set assigned to $a(i+1,v)$ in (12). Thus $x' = b(i,J)$, where $\partial(J) = u$ and $\partial'(J) = v$, that is $J = J(u,v)$. Part (**d**) follows immediately from part (**b**). $\square$

The product operator Γ_3. The arguments for the operator are three red blocks $\mathcal{X}_1$, $\mathcal{X}_2$, and $\mathcal{Z}$, where $\mathcal{Z}$ is isomorphic to the product $\mathcal{X}_1 \times \mathcal{X}_2$, together with a one-to-one correspondence:

$$z \in \mathcal{Z} \;\longleftrightarrow\; (x_1, x_2) \in \mathcal{X}_1 \times \mathcal{X}_2 \tag{14}$$

We write $z = z(x_1, x_2)$ to denote the element of $\mathcal{Z}$ that corresponds to the pair $(x_1, x_2) \in \mathcal{X}_1 \times \mathcal{X}_2$ in (14). An application of Γ_3 augments $G' = (\mathcal{R}, \mathcal{B}, E')$ in the following ways:

a. Two auxiliary blocks of red vertices are created:

$$\begin{aligned} \mathcal{P}_1 &= \{\, \rho_1(x_1,x_2) \;:\; x_1 \in \mathcal{X}_1,\; x_2 \in \mathcal{X}_2 \,\} \\ \mathcal{P}_2 &= \{\, \rho_2(x_1,x_2) \;:\; x_1 \in \mathcal{X}_1,\; x_2 \in \mathcal{X}_2 \,\} \end{aligned}$$

b1. The operator Γ_2 is applied to the arguments $\mathcal{X} = \mathcal{X}_1$ and $\mathcal{X}' = \mathcal{P}_1$ with the assignment:

$$x \in \mathcal{X}_1 \;\mapsto\; \{\, \rho_1(x,y) \;:\; y \in \mathcal{X}_2 \,\} \tag{15}$$

b2. The operator Γ_2 is applied to the arguments $\mathcal{X} = \mathcal{X}_2$ and $\mathcal{X}' = \mathcal{P}_2$ with the assignment:

$$y \in \mathcal{X}_2 \;\mapsto\; \{\, \rho_2(x,y) \;:\; x \in \mathcal{X}_1 \,\} \tag{16}$$

c. The operator Γ_2 is applied as an equality enforcer to the arguments $\mathcal{X} = \mathcal{P}_1$ and $\mathcal{X}' = \mathcal{P}_2$ with the assignment:

$$\rho_1(x_1,x_2) \;\mapsto\; \{\, \rho_2(x_1,x_2) \,\} \tag{17}$$

d1. The operator Γ_2 is applied as an equality enforcer to the arguments $\mathcal{X} = \mathcal{P}_2$ and $\mathcal{X}' = \mathcal{Z}$ with the assignment:

$$\rho_2(x_1,x_2) \;\mapsto\; \{\, z(x_1,x_2) \,\} \tag{18}$$

d2. The operator Γ_2 is applied as an equality enforcer to the arguments $\mathcal{X} = \mathcal{Z}$ and $\mathcal{X}' = \mathcal{P}_1$ with the assignment:

$$z(x_1,x_2) \;\mapsto\; \{\, \rho_1(x_1,x_2) \,\} \tag{19}$$

Lemma 5. *The product operator Γ_3 is locally perfect. In particular, suppose that this operator is applied to the three red blocks $\mathcal{X}_1, \mathcal{X}_2$, and $\mathcal{Z}$, and let S be a k'-element dominating set in $G' = (\mathcal{R}, \mathcal{B}, E')$, with $S \cap \mathcal{X}_1 = \{x\}$ and $S \cap \mathcal{X}_2 = \{y\}$. Then $S \cap \mathcal{Z} = \{z(x,y)\}$.*

Proof. It follows from Lemma 2 that S intersects each of the red blocks $\mathcal{P}_1, \mathcal{P}_2$, and $\mathcal{Z}$ at a single vertex, say $\rho_1(x',y') \in \mathcal{P}_1$, $\rho_2(x'',y'') \in \mathcal{P}_2$, and $z \in \mathcal{Z}$, respectively. We can now argue a series of simple observations regarding $\rho_1(x', y'), \rho_2(x'', y'')$, and z, which follow from Lemma 3 in conjunction with the assignments in (15) – (19). In view of (15), respectively (16), we have that $x' = x$, respectively $y'' = y$. By the equality enforcing assignment in (17), we have $x' = x''$ and $y' = y''$. Thus $x'' = x' = x$ and $y'' = y' = y$. It now follows from either (18) or (19) that $z = z(x, y)$.

Since the product operator Γ_3 is composed from five applications of the branch operator Γ_2, the local perfection of Γ_3 follows from the local perfection of Γ_2 established in Lemma 3. Alternatively, this can be proved directly by verifying that each of the $|\mathcal{X}_1| + |\mathcal{X}_2| + 3\,|\mathcal{X}_1||\mathcal{X}_2|$ blue vertices created by Γ_3 is adjacent to one, and only one, of the five red vertices $x \in \mathcal{X}_1$, $y \in \mathcal{X}_2$, $\rho_1(x, y) \in \mathcal{P}_1$, $\rho_2(x, y) \in \mathcal{P}_2$, and $z(x, y) \in \mathcal{Z}$, regardless of the choice of $x \in \mathcal{X}_1$ and $y \in \mathcal{X}_2$. $\square$

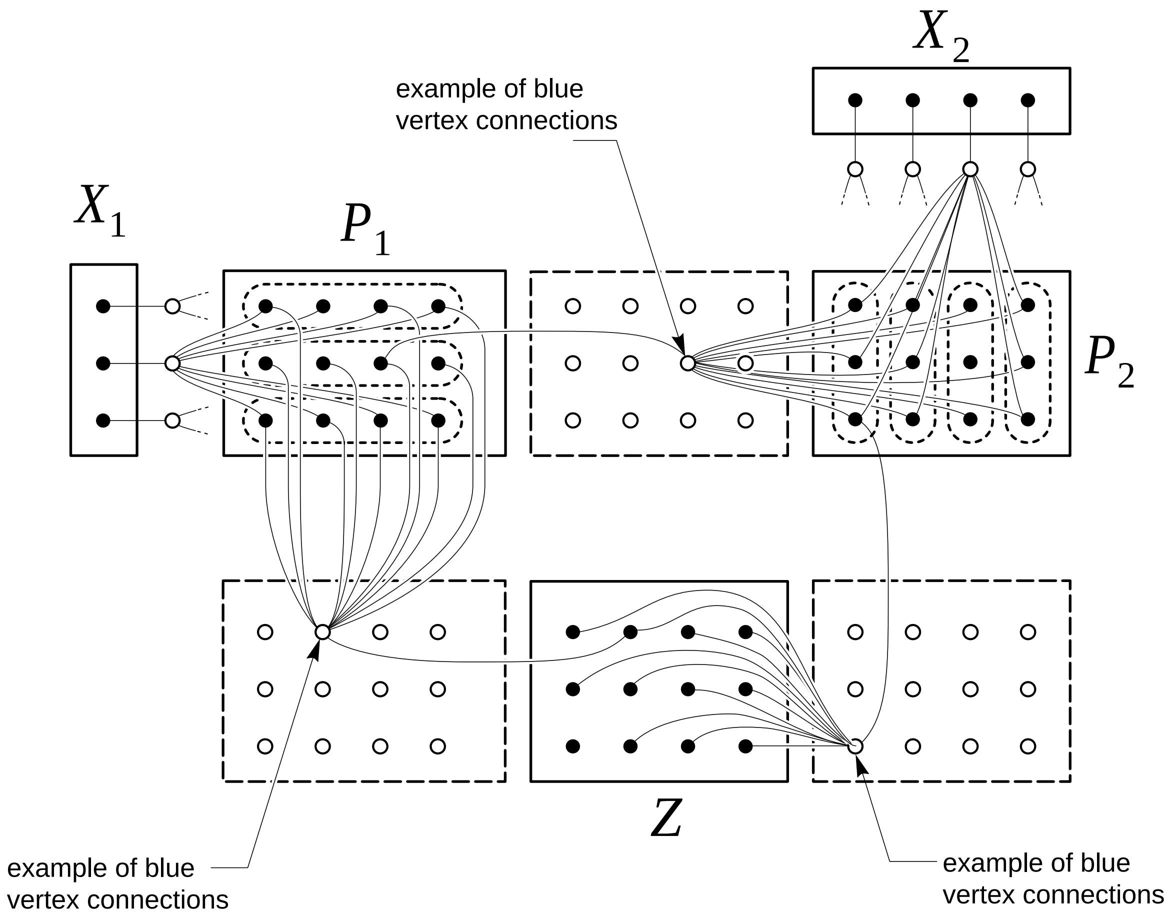
X_2
example of blue vertex connections
X_1
P_1
P_2
Z
example of blue vertex connections
example of blue vertex connections

Figure 5: *The product operator* Γ_3

With the product operator Γ_3 at hand, we can now describe the next three steps in our construction of $G' = (\mathcal{R}, \mathcal{B}, E')$. These three steps establish a relation between the blocks in R_3, R_4, R_5 and the blocks of R_1, R_2 which represent the choice of some k vertices of G according to Lemma 4.

Step 3. *For each pair (i, i') with $i, i' \in [k]^*$, apply the operator Γ_3 to the arguments $\mathcal{X}_1 = \mathcal{B}(i)$, $\mathcal{X}_2 = \mathcal{B}(i')$ and $\mathcal{Z} = \mathcal{C}(i, i')$ with the product correspondence:*

$$c(i, i', J, J') \in \mathcal{C}(i, i') \;\longleftrightarrow\; (b(i, J), b(i', J')) \in \mathcal{B}(i) \times \mathcal{B}(i')$$

Step 4. *For each pair (i, i') with $i \in [k]$ and $i' \in [k]^*$, apply the operator Γ_3 to the arguments $\mathcal{X}_1 = \mathcal{A}(i)$, $\mathcal{X}_2 = \mathcal{B}(i')$ and $\mathcal{Z} = \mathcal{C}'(i, i')$ with the product correspondence:*

$$c'(i, i', u, J) \in \mathcal{C}'(i, i') \;\longleftrightarrow\; (a(i, u), b(i', J)) \in \mathcal{A}(i) \times \mathcal{B}(i')$$

Step 5. *For each pair (i, i') with $i \in [k]^*$ and $i' \in [k]$, apply the operator Γ_3 to the arguments $\mathcal{X}_1 = \mathcal{B}(i)$, $\mathcal{X}_2 = \mathcal{A}(i')$ and $\mathcal{Z} = \mathcal{C}''(i, i')$ with the product correspondence:*

$$c''(i, i', J, u) \in \mathcal{C}''(i, i') \;\longleftrightarrow\; (b(i, J), a(i', u)) \in \mathcal{B}(i) \times \mathcal{A}(i')$$

At this stage of the construction, the red/blue graph $G' = (\mathcal{R}, \mathcal{B}, E')$ is highly structured, but the adjacency structure of G' is *independent* of the adjacency structure of $G = (V, E)$. The following operator will finally establish the connection between $G = (V, E)$ and $G' = (\mathcal{R}, \mathcal{B}, E')$.

The G-adjacency operator Γ_4. This operator takes as arguments all the vertices of R_1, that is, all the $\mathcal{A}$-blocks $\mathcal{A}(1), \mathcal{A}(2), \ldots, \mathcal{A}(k)$. A set of $n = |V|$ blue vertices

$$\{\, \beta(u) \;:\; u \in V \,\}$$

is created. These vertices are connected as follows. For each $i = 1, 2, \ldots, k$, the blue vertex $\beta(u)$ is connected to $a(i, u)$, and to $a(i, v)$ for all $v \in V$ that are adjacent to u in G.

It is easy to see that an application of Γ_4 essentially amounts to replicating k times the original graph $G = (V, E)$ as a red/blue bipartite graph with the set of red vertices $\mathcal{A}(i)$ and the set of blue vertices $\{\beta(u) : u \in V\}$, both isomorphic to V. This is precisely what we do next.

Step 6. *Apply the G-adjacency operator Γ_4 to all of the $\mathcal{A}$-blocks $\mathcal{A}(1), \mathcal{A}(2), \ldots, \mathcal{A}(k)$.*

Let S be a k'-element dominating set in $G' = (\mathcal{R}, \mathcal{B}, E')$. We know from Lemma 2 that S intersects each of the red blocks, in particular each of the blocks $\mathcal{A}(1), \mathcal{A}(2), \ldots, \mathcal{A}(k)$, at a single vertex. Thus for each $i = 1, 2, \ldots, k$, we can define v_i to be the unique vertex of V, such that $S \cap \mathcal{A}(i) = \{a(i, v_i)\}$. With this notation, let $V(S) = \{v_1, v_2, \ldots, v_k\}$.

Lemma 6. *The set $V(S)$ is a k-element dominating set in G. Furthermore, if S is a perfect code in $G' = (\mathcal{R}, \mathcal{B}, E')$, then $V(S)$ a k-element perfect code in G.*

Proof. It follows from Lemma 4 that $v_1, v_2, \ldots, v_k$ are all distinct (in fact $v_1 < v_2 < \cdots < v_k$). Hence $|V(S)| = k$. The rest is immediate from the definition of Γ_4: a vertex $u \in V$ has a (unique) neighbor in $V(S)$ if and only if the vertex $\beta(u)$ created by Γ_4 has a (unique) neighbor in S. □

Notice that the G-adjacency operator Γ_4 is not locally perfect, unless $V(S)$ is actually a perfect code in $G = (V, E)$ for every k'-element dominating set S in $G' = (\mathcal{R}, \mathcal{B}, E')$. However, the following operator ensures that the latter condition is always satisfied.

We say that two distinct vertices u and u' in $G=(V,E)$ are *close* if there is a path length 1 or 2 between them (notice that a vertex u is not considered close to itself). It is easy to see that a dominating set in G is also a perfect code if and only if it does not contain close vertices.

The perfection operator Γ_5. This operator takes as arguments all of the $\mathcal{C}, \mathcal{C}'$, and $\mathcal{C}''$ blocks. An application of the operator results in:

a. The creation of a set of blue vertices, with one vertex for each ordered pair (u,u') of close vertices in G, that is:

$$\{\, \beta(u,u') \ : \ u \text{ is close to } u' \text{ in } G \ \} \tag{20}$$

b. The creation of edges connecting each blue vertex $\beta(u,u')$ in this set to all the red vertices contained in one of the following three sets

$$\begin{aligned}
\mathbb{C}(u,u') &= \{\, c(i,i',J,J') \ : \ i,i' \in [k]^*,\ u \in J,\ u' \in J' \,\} \\
\mathbb{C}'(u,u') &= \{\, c'(i,i',u,J) \ : \ i \in [k],\ i' \in [k]^*,\ u' \in J \,\} \\
\mathbb{C}''(u,u') &= \{\, c''(i,i',J,u') \ : \ i \in [k]^*,\ i' \in [k],\ u \in J \,\}
\end{aligned}$$

The idea of the perfection operator Γ_5 is to ensure that if $V(S)$ contains close vertices then S cannot be a dominating set in $G'=(\mathcal{R},\mathcal{B},E')$. This is accomplished in the following step.

Step 7. *Apply the perfection operator* Γ_5 *to all of the* $\mathcal{C}, \mathcal{C}'$, *and* $\mathcal{C}''$ *blocks.*

Next, we need some more notation. We again let S denote a k'-element dominating set in G', and define the sets $J_0, J_1, \ldots, J_k$ as follows:

$$J_i \quad = \quad J(u,v) \subset V \qquad \text{so that } S \cap \mathcal{B}(i) = \{b(i,J(u,v))\} \qquad \text{for } i = 0,1,\ldots,k$$

It follows from Lemma 4 that the sets $J_0, J_1, \ldots, J_k$ are well defined, and the sequence of sets $J_0, \{v_1\}, J_1, \{v_2\}, J_2, \ldots, J_{k-1}, \{v_k\}, J_k$ constitutes a partition of the vertex set V of G.

Lemma 7. *The perfection operator* Γ_5 *is locally perfect, and the set* $V(S)=\{v_1,v_2,\ldots,v_k\}$ *does not contain close vertices.*

Proof. With the notation just defined, it follows from Lemma 5 along with Steps 3, 4, and 5 of our construction that

$$\begin{aligned}
S \cap \mathcal{C}(i,i') &= \{\, c(i,i',J_i,J_{i'}) \,\} && \text{for all } i \in [k]^* \text{ and } i' \in [k]^* \\
S \cap \mathcal{C}'(i,i') &= \{\, c'(i,i',v_i,J_{i'}) \,\} && \text{for all } i \in [k] \text{ and } i' \in [k]^* \\
S \cap \mathcal{C}''(i,i') &= \{\, c''(i,i',J_i,v_{i'}) \,\} && \text{for all } i \in [k]^* \text{ and } i' \in [k]
\end{aligned}$$

Now consider a pair (u,u') of close vertices in G. Since $J_0, \{v_1\}, J_1, \{v_2\}, J_2, \ldots, J_{k-1}, \{v_k\}, J_k$ is a partition of the vertex set V of G, exactly one statement is true on the following list, describing where u is to be found and where u' is to be found.

Case 1: $u \in J_i$ and $u' \in J_{i'}$

In this case $\beta(u,u')$ in (20) is adjacent to a single red vertex in S. Specifically $\beta(u,u')$ is connected to $c(i,i',J_i,J_{i'}) \in \mathbb{C}(u,u')$, where $\{\, c(i,i',J_i,J_{i'}) \,\} = S \cap \mathcal{C}(i,i')$.

Case 2: $u = v_i$ and $u' \in J_{i'}$

In this case $\beta(u, u')$ in (20) is adjacent to a single red vertex in S. Specifically $\beta(u, u')$ is connected to $c'(i, i', v_i, J_{i'}) \in \mathbb{C}'(u, u')$, where $\{ c'(i, i', v_i, J_{i'}) \} = S \cap \mathcal{C}'(i, i')$.

Case 3: $u \in J_i$ and $u' = v_{i'}$

In this case $\beta(u, u')$ in (20) is adjacent to a single red vertex in S. Specifically $\beta(u, u')$ is connected to $c''(i, i', J_i, v_{i'}) \in \mathbb{C}''(u, u')$, where $\{ c''(i, i', J_i, v_{i'}) \} = S \cap \mathcal{C}''(i, i')$.

Case 4: $u = v_i$ and $u' = v_{i'}$

In this case $\beta(u, u')$ is **not** adjacent to any of the vertices of S.

In each case, we see that a blue vertex $\beta(u, u')$ created by Γ_5 is adjacent to at most one vertex of S, which implies that Γ_5 is locally perfect. Furthermore, since S is a dominating set by assumption, Case 4 above cannot happen. In other words, there is no pair of close vertices in the set $V(S)$ "chosen" by a k'-element dominating set S in $G' = (\mathcal{R}, \mathcal{B}, E')$. □

Lemma 8. *The set $V(S)$ is a k-element perfect code in $G = (V, E)$.*

Proof. By Lemma 6, the set $V(S)$ is a k-element dominating set in G, and by Lemma 7 it does not contain close vertices. Hence $V(S)$ is a perfect code. □

We can now complete our construction, and complete the proof of Theorem 1. As mentioned in the beginning of this section, the last step in the construction is:

Step 8. *Apply the block guard operator Γ_1 to every red block constructed thus far.*

Theorem 1 now follows from a series of easy observations. Property **P1** of Theorem 1 is established in Lemma 1. The fact that $V(S)$ is necessarily a perfect code in G, established in Lemma 8, further implies that the G-adjacency operator Γ_4 is locally perfect. Thus *all* the operators used in our construction are locally perfect, and property **P2** of Theorem 1 holds. The "if" part of property **P3** then follows directly from Lemma 8. The reader can now verify the few remaining details to see that the forward translation of a k-element perfect code in $G = (V, E)$ to a k'-element perfect code in $G' = (\mathcal{R}, \mathcal{B}, E')$, outlined in (5) – (9), works correctly.

4. Membership in the parametrized complexity class $W[2]$

All of the hardness results for parameterized complexity that we derive in this paper are by reduction from the Perfect Code problem. This particular problem has eluded exact classification in the $W[t]$ hierarchy for a number of years. What is known [DF95b] is that Perfect Code is hard for $W[1]$ and belongs to $W[2]$. It may be a representative of a natural parameterized complexity degree, which is intermediate between the $W[1]$ and $W[2]$ complexity classes. This remains an interesting open problem in the structure of the parametric complexity classes, especially in view of connections to one-per-clause satisfiability problems.

In this section we indicate how some of the problems considered in this paper, namely Maximum-Likelihood Decoding, Weight Distribution, and Minimum Distance, can be shown to belong to the parameterized complexity class $W[2]$. A wide variety of problems are known to be complete or hard for $W[2]$, including DOMINATING SET, BANDWIDTH, LENGTH-k FACTORIZATION OF MONOIDS, LONGEST COMMON SUBSEQUENCE FOR k-SEQUENCES, and k-PROCESSOR PRECEDENCE CONSTRAINED SCHEDULING — see [BFH94, CCDF96, BDFW95, BF95].

To establish the necessary background for Theorem 10 below, we now briefly recall the definition of the classes of the $W[t]$ hierarchy based on problems about bounded-depth circuits. We will generally follow the exposition in [DF95a]. We first define circuits in which some logic gates have bounded fan-in and some have unrestricted fan-in. It is assumed that fan-out is never restricted.

Definition. *A Boolean circuit is of* ***mixed type*** *if it consists of circuits having gates of the following kinds:*

small gates: $\neg$ *gates,* $\wedge$ *gates and* $\vee$ *gates with bounded fan-in; we will usually assume that the bound on fan-in is* 2 *for* $\wedge$ *gates and* $\vee$ *gates, and* 1 *for* $\neg$ *gates*

large gates: $\wedge$ *gates and* $\vee$ *gates with unrestricted fan-in*

The *depth* of a circuit $\mathcal{C}$ is defined to be the maximum number of gates (small or large) on an input-output path in $\mathcal{C}$. The *weft* of a circuit $\mathcal{C}$ is the maximum number of large gates on an input-output path in $\mathcal{C}$. We say that a family of decision circuits F has *bounded depth* if there is a constant h such that every circuit in the family F has depth at most h. We say that F has *bounded weft* if there is constant t such that every circuit in the family F has weft at most t.

Let F be a family of mixed type decision circuits. We allow that F may have many different circuits with a given number of inputs. To F we associate the following parameterized circuit problem $L_F = \{ (\mathcal{C}, k) \ : \ \mathcal{C} \in F$ accepts some input vector of weight $k\}$, where the (Hamming) weight of a Boolean vector x is the number of 1's in the vector.

Definition. *A parameterized language L belongs to $W[t]$ if L reduces to the parameterized circuit problem $L_{F(t,h)}$ for the family $F(t,h)$ of mixed type decision circuits of weft at most t, and depth at most h, for some constant h.*

Definition. *A parameterized language L belongs to $W^*[t]$ if it belongs to $W[t]$ with the definition of* a small gate *being revised to allow fan-in bounded by a fixed arbitrary function of k, and where the depth of a circuit is allowed to be a function of k as well.*

It is an important open problem whether $W^*[t] = W[t]$ for all t. The significance of this question is that for purposes of establishing membership in the $W[t]$ hierarchy, we would like to have the most generous possible characterization of $W[t]$ available to work with. It is shown that $W^*[1] = W[1]$ and $W^*[2] = W[2]$, in [DFT96] and [DF97b], respectively. For $t \geq 3$ the question is still open. Our argument here makes essential use of the result of [DF97b] that $W^*[2] = W[2]$.

Theorem 10. WEIGHT DISTRIBUTION *belongs to $W[2]$.*

Proof. Given an instance H and k of WEIGHT DISTRIBUTION, we describe how to compute a pair (E, k'), consisting of a Boolean expression E and a positive integer k', such that the circuit corresponding to E has a form allowed by the definition of $W^*[2] = W[2]$, and such that H, k is a yes-instance of WEIGHT DISTRIBUTION if and only if E is satisfied by a weight k' truth assignment. In order for this to be a parametric reduction, we must have k' computed purely as a function of k. Our reduction is simple in this regard: we take $k' = k$.

Suppose that H is an $m \times n$ binary matrix, and let $h_1, h_2, \ldots, h_n$ denote the columns of this matrix. For $j = 1, 2, \ldots, m$, we will write $h_i[j]$ to denote the j-th component of h_i. The set V of Boolean variables for E is:

$$V \;=\; \{\, v[b,i] \;:\; b = 1, 2, \ldots, k \text{ and } i = 1, 2, \ldots, n \,\}$$

Intuition can be served by viewing V as consisting of k "choice blocks," each of size n. The expression E is constructed so that any satisfying truth assignment must make exactly one variable in each of these blocks `true`, and will in this way indicate a set of k columns of H.

Let $\mathcal{E}$ denote the set of all subsets of $\{1, 2, \ldots, k\}$ of even cardinality. For each $j \in \{1, 2, \ldots, m\}$ and $\sigma \in \mathcal{E}$, define the Boolean expressions:

$$\begin{aligned}
E^{+}(\sigma, j) &= \bigwedge_{b \in \sigma} \; \bigwedge_{i:\, h_i[j] = 0} \neg v[b,i] \\
E^{-}(\sigma, j) &= \bigwedge_{b \notin \sigma} \; \bigwedge_{i:\, h_i[j] = 1} \neg v[b,i] \\
E(\sigma, j) &= E^{+}(\sigma, j) \;\wedge\; E^{-}(\sigma, j)
\end{aligned}$$

Then the expression E has the form:

$$E \;=\; E_1 \;\wedge\; E_2 \;\wedge\; E_3$$

where:

$$\begin{aligned}
E_1 &= \bigwedge_{1 \le b < b' \le k} \; \bigwedge_{i=1}^{n} (\neg v[b,i] \;\vee\; \neg v[b',i]) \\
E_2 &= \bigwedge_{b=1}^{k} \left(\bigvee_{i=1}^{n} v[b,i] \right) \\
E_3 &= \bigwedge_{j=1}^{m} \; \bigvee_{\sigma \in \mathcal{E}} E(\sigma, j)
\end{aligned}$$

It is easy to see that if the definition of a small gate allows fan-in bounded by a function of k, then each of the subexpressions E_1 and E_2 has weft one, while the subexpression E_3 has weft two. The depth of E_1, E_2, and E_3 is 4, 2, and 6, respectively, so that the depth of E itself is 7. Thus E belongs to a family of weft-two circuits allowed by the definition of $W^*[2]$. We next argue the correctness of the reduction. Note the validity of the following easy claims.

Claim 1. *The subexpression E_2 is satisfied by a weight k truth assignment to the variables of V if and only if exactly one variable in each of the k blocks is assigned the value* `true`.

Claim 2. *The subexpression $E' = E_1 \wedge E_2$ is satisfied by a weight k truth assignment to the variables of V if and only if exactly one variable in each of the k blocks is assigned the value* `true`, *in such a way that the second indices of the* `true` *variables are all distinct.*

Let τ be a weight k truth assignment that satisfies E'. It follows from Claim 1, that for each $b \in \{1, 2, \ldots, k\}$, there is a unique $i \in \{1, 2, \ldots, n\}$ such that $v[b, i]$ is assigned the value `true` by τ. Thus τ and b together specify the unique i-th column of H. For $j = 1, 2, \ldots, m$, we let $\alpha(b, j, \tau)$ denote the binary value to be found in the j-th row of this column, that is $\alpha(b, j, \tau) = h_i[j]$.

Claim 3. *The expression E_3 is satisfied by a weight k truth assignment $\tau : V \to \{\texttt{true}, \texttt{false}\}$ that also satisfies $E' = E_1 \wedge E_2$, if and only if for each $j \in \{1, 2, \ldots, m\}$ there is some $\sigma \in \mathcal{E}$ such that the following equality holds: $\sigma = \{b : \alpha(b, j, \tau) = 1\}$.*

Now suppose that H, k is a yes-instance of WEIGHT DISTRIBUTION, and let $h_{i_1}, h_{i_2}, \ldots, h_{i_k}$ be the k columns of H that sum to the all-zero vector. Let τ be the truth assignment that assigns the variables $v[1, i_1], v[2, i_2], \ldots, v[k, i_k]$ to be `true`, and assigns all the other variables in V the value `false`. Clearly τ has weight k. It follows from Claim 2 that τ satisfies $E' = E_1 \wedge E_2$. For $j = 1, 2, \ldots, m$, let σ_j be the subset of $\{1, 2, \ldots, k\}$ consisting of all b such that $h_{i_b}[j] = 1$. Since

$$h_{i_1}[j] + h_{i_2}[j] + \cdots + h_{i_k}[j] \;=\; 0 \pmod 2$$

for all j, it follows that the number of such indices b must be even. Hence $\sigma_j \in \mathcal{E}$ for all j. It is not difficult to see that for the truth assignment τ specified above, σ_j is precisely the set $\{b : \alpha(b, j, \tau) = 1\}$. Hence by Claim 3, τ also satisfies E_3, and therefore E itself.

Conversely, suppose that E has a truth assignment τ of weight k. By Claim 2, there are k distinct indices $i_1, i_2, \ldots, i_k$ such that τ assigns to the k variables $v[1, i_1], v[2, i_2], \ldots, v[k, i_k]$ the value `true`, and assigns all other variables in V the value `false`. By Claim 3, the number of elements in the set $\{b : \alpha(b, j, \tau) = 1\} = \{b : h_{i_b}[j] = 1\}$ must be even for each $j = 1, 2, \ldots, m$. Hence the k columns $h_{i_1}, h_{i_2}, \ldots, h_{i_k}$ of H sum to the all-zero vector. □

The above argument can be easily modified to show that MAXIMUM-LIKELIHOOD DECODING and MINIMUM DISTANCE also belong to $W^*[2] = W[2]$. We conjecture that all of these problems are equivalent to the PERFECT CODE problem.

5. Concluding discussion and open problems

We have shown that four of six fundamental computational problems in the domains of linear codes and integer lattices are NP-complete and hard for the parametrized complexity class $W[1]$. The obvious outstanding open problems are:

- Is the MINIMUM DISTANCE problem, recently proved to be NP-complete in [Var97b], also hard for $W[1]$?
- Is the SHORTEST VECTOR problem hard for NP and $W[1]$?

A consequence of the proof in [Var97b] that the MINIMUM DISTANCE problem is NP-hard is that EVEN SET is NP-hard. This leaves us in the curious situation that the only known proof of this seemingly quite combinatorial result is by means of sophisticated algebraic techniques, deeply rooted in coding theory. Is there a direct combinatorial proof? Understanding this issue may shed some light on whether the combinatorial methods used here to show NP and $W[1]$ hardness for THETA SERIES can be extended to the SHORTEST VECTOR problem.

A. Appendix: short survey of parameterized complexity

Over the past several years, it has become increasingly clear that classical complexity frameworks such as NP-completeness and PSPACE-completeness are not adequate to address intractability questions for problems that are naturally parameterized, and for which the important applications are covered by parameter values of, say $k \leq 50$.

For example, consider the VERTEX COVER problem, which asks whether a graph G on n vertices has a vertex cover of size at most k. This is one of the six NP-complete problems singled out for attention by Garey and Johnson [GJ79]. The best known result, presently, is that VERTEX COVER can be solved in time $O(kn+(4/3)^k k^2)$, with a very small hidden constant [BFR96]. This means that although the problem is NP-complete, it is well-solved for input graphs of *unlimited* size, as long as k is at most 70 or so. Strong tractability results such as this seem to be not uncommon when problems are qualitatively classifiable as fixed-parameter tractable, meaning that they belong to the parametrized complexity class FPT, formally defined below. Three of the six basic NP-complete problems considered by Garey and Johnson in [GJ79, Chapter 3] are fixed-parameter tractable.

In general, a variety of metrics can be applied to the input of a computational problem. The total length of the input is one basic measurement, but it is by no means the only important one. It is natural to try to understand how different input measurements interact in determining problem complexity. Furthermore, it is essential to understand such interactions, in order to exploit the opportunities for designing algorithms that are sensitive to natural input distributions.

A generic example of a parameterization is provided by the many well-known decision problems concerning graphs, that take as input a graph G and a positive integer k. The parameter k appears to contribute to the complexity of such problems in two qualitatively different ways. GRAPH GENUS, MIN-CUT LINEAR ARRANGEMENT, VERTEX COVER, and FEEDBACK VERTEX SET FOR UNDIRECTED GRAPHS (see, for example [GJ79], for definitions) can all be solved in time $O(f(k)n^c)$, where c is a constant independent of k, and $f(\cdot)$ is some (arbitrary) function. This "good behavior" is termed *fixed-parameter tractability* in the theory introduced in [DF95a]. As is the case with polynomial-time complexity, the exponent c is typically small. One can equivalently define fixed-parameter tractability to mean solvability in time $O(f(k)+n^c)$, that is, with only an *additive* contribution from the parameter [CCDF97]. There is a rich collection of distinctive techniques for devising FPT algorithms (see [DF95c, DF97b, KST94, LeC97, Ste92]).

Contrasting complexity behavior is exhibited by the naturally parameterized problems such as CLIQUE, DOMINATING SET, and BANDWIDTH, for which the best known algorithms have running times $O(n^{ck})$. These problems have been shown to be complete or hard for the various levels of the W hierarchy of parameterized complexity

$$W[1] \subseteq W[2] \subseteq \cdots \subseteq W[P] \subseteq \cdots \subseteq XP$$

which can be taken as evidence that they are unlikely to be fixed-parameter tractable [BFH94]. With these problems, we seem to hit a natural "wall" requiring brute force effort, much as is typically the case with NP-complete problems. For example, essentially no better algorithm is known for the k-DOMINATING SET problem than checking all k-subsets.

As in the theory of NP-completeness, there are roughly two kinds of arguments that can be offered for believing that parameterized problems that are complete or hard for $W[1]$ are not likely to be fixed-parameter tractable. The first kind of argument is, roughly speaking, sociological.

So many different kinds of problems stand or fall together that the combination of efforts expended unsuccessfully from the various vantages compels a belief in inherent intractability. The second kind of argument is some form of direct intuition concerning the nature of the computations that define the issue — e.g., nondeterministic as opposed to deterministic polynomial-time.

For parameterized complexity, both kinds of arguments can be made. Although the amount of unsuccessful effort that has been expended in an attempt to show fixed-parameter tractability for $W[1]$-hard problems is much less than the total effort expended to date in attempting to develop polynomial-time algorithms for NP-complete problems, it is still considerable and accumulating. The parameterized complexity class $W[1]$ is particularly interesting and important, for several reasons: there is a substantial list of natural and useful computational problems that are precisely $W[1]$-complete. This list includes CLIQUE, INDEPENDENT SET, VAPNIK-CHERVONENKIS DIMENSION, MONOTONE DATA COMPLEXITY FOR RELATIONAL DATABASES, SQUARE TILING, k-STEP DERIVATION FOR CONTEXT SENSITIVE GRAMMARS, m-LENGTH COMMON SUBSEQUENCE FOR k SEQUENCES, and k-LENGTH POST CORRESPONDENCE, among other problems.

Direct intuition about $W[1]$ is also available. It is shown in [DFKHW94, CCDF96] that the k-step halting problem for nondeterministic Turing machines is $W[1]$-complete. This problem is formally defined as follows.

Problem: SHORT TURING MACHINE ACCEPTANCE
Instance: A nondeterministic Turing machine M and a positive integer k.
Question: Does M have a computation path accepting the empty string in at most k steps?
Parameter: k

This is a problem so generic and opaque that it is hard to imagine that there is any algorithm for it that radically improves on simply exploring the n-branching depth-k tree of allowed transitions exhaustively. This is essentially the same intuition as the belief that Cook's Theorem provides a basis for the intractability of NP-complete problems.

For a definition of the $W[t]$ complexity classes and the fundamentals of parameterized complexity, we refer the reader to § 4 and [DF95a, DF95b, DF95c, DF97b]. Here, we will briefly review the basic definitions of a parametrized problem and fixed-parameter tractability.

Definition. *A* ***parameterized problem*** *is a set $L \subseteq \Sigma^* \times \Sigma^*$, where Σ is a fixed alphabet. For convenience, we can think of a parameterized problem as a subset L of $\Sigma^* \times N$, where N is the set of nonnegative integers.*

Definition. *We say that parameterized problem L is (uniformly)* ***fixed-parameter tractable*** *if there is a constant α and an algorithm Φ, such that Φ decides if $(x,k) \in L$ in time $f(k)|x|^\alpha$ where $f : N \rightarrow N$ is an arbitrary function.*

Let A and B be parameterized problems. We say that A is (uniformly many : 1) *reducible* to B if there is an algorithm Φ which transforms $(x,k) \in \Sigma^* \times N$ into $(x', g(k))$ in time $f(k)|x|^\alpha$, where $f, g : N \rightarrow N$ are arbitrary functions and α is a constant independent of k, so that $(x,k) \in A$ if and only if $(x', g(k)) \in B$. Such an algorithm Φ may be called a parametric transformation. It is easy to see that if A reduces to B, and B is fixed-parameter tractable, then so too is A.

References

[ABSS93] S. Arora, L. Babai, J. Stern, and Z. Sweedyk, The hardness of approximate optima in lattices, codes, and systems of linear equations, in *Proc. 34-th Annual Symp. Found. Computer Science*, Palo Alto, CA, (1993), 724–733.

[Ajt97] M. Ajtai, The shortest vector problem in L_2 is NP-hard for randomized reductions, personal communication, May 1997.

[BFR96] R. Balasubramanian, M.R. Fellows, and V. Raman, An improved fixed-parameter algorithm for vertex cover, preprint, 1996.

[Bar94] A. Barg, Some new NP-complete coding problems, *Probl. Peredachi Informatsii*, **30**, (1994), 23–28, (in Russian).

[BDFW95] H. Bodlaender, R.G. Downey, M.R. Fellows, and H.T. Wareham, The parameterized complexity of the longest common subsequence problem, *Theoretical Computer Science*, **147**, (1995), 31–54.

[BF95] H. Bodlaender and M.R. Fellows, On the complexity of k-processor scheduling, *Operations Research Letters*, **18**, (1995), 93–98.

[BFH94] H. Bodlaender, M.R. Fellows, and M.T. Hallett, Beyond NP-completeness for problems of bounded width: hardness for the W hierarchy, in *Proc. 26-th Annual ACM Symp. Theory of Computing*, (1994), 449–458.

[BMvT78] E.R. Berlekamp, R.J. McEliece, and H.C.A. van Tilborg, On the inherent intractability of certain coding problems, *IEEE Trans. Inform. Theory*, **24**, (1978), 384–386.

[BN90] J. Bruck and M. Naor, The hardness of decoding linear codes with preprocessing, *IEEE Trans. Inform. Theory*, **36**, (1990), 381–385.

[LeC97] L. Cai, Fixed-parameter tractability of graph modification problems for hereditary properties, *Inform. Processing Letters*, to appear.

[CC93] L. Cai and J. Chen, Fixed parameter tractability and approximability of NP-hard optimization problems, in *Proc. 2-nd Israel Symp. Theory of Computing Systems*, (1993), 118–126.

[CCDF96] L. Cai, J. Chen, R.G. Downey, and M.R. Fellows, On the parameterized complexity of short computation and factorization, *Arch. Math. Logic*, to appear.

[CCDF97] L. Cai, J. Chen, R.G. Downey, and M.R. Fellows, Advice classes of parameterized tractability, *Annals Pure and Applied Logic*, **84**, (1997), 119–138.

[DF95a] R.G. Downey and M.R. Fellows, Fixed parameter tractability and completeness I: basic theory, *SIAM J. Computing*, **24**, (1995), 873–921.

[DF95b] R.G. Downey and M.R. Fellows, Fixed parameter tractability and completeness II: completeness for $W[1]$, *Theoretical Comp. Science*, **141**, (1995), 109–131.

[DF95c] R.G. Downey and M.R. Fellows, Parameterized computational feasibility, in *Feasible Mathematics*, P. Clote and J. Remmel (Eds.), Birkhauser (1995), 219–244.

[DF97a] R.G. Downey and M.R. Fellows, Threshold dominating sets and an improved characterization of $W[2]$, *Theoretical Computer Science*, **189**, (1997), to appear.

[DF97b] R.G. Downey and M.R. Fellows, *Parameterized Complexity.* Springer-Verlag (1997), to appear.

[DFKHW94] R.G. Downey, M.R. Fellows, B.M. Kapron, M.T. Hallett, and H.T. Wareham, The parameterized complexity of some problems in logic and linguistics, in *Lect. Notes Comp. Science*, Springer-Verlag, **813**, (1994), 89–100.

[DFR97] R.G. Downey, M.R. Fellows, and K.R. Regan, Parameterized circuit complexity and the W hierarchy, *Theoretical Comp. Science*, **189**, (1997), to appear.

[DFT96] R.G. Downey, M.R. Fellows, and U. Taylor, The parameterized complexity of relational database queries and an improved characterization of $W[1]$, in *Combinatorics, Complexity, and Logic*, D. Bridges, C. Calude, J. Gibbons, S. Reeves, and I. Witten (Eds.), Springer-Verlag (1996), 194-213,

[GJ79] M.R. Garey and D.S. Johnson, *Computers and Intractability: Guide to the Theory of NP-completeness*, Freeman, San Francisco, 1979.

[Kra94] J. Kratochvíl, Regular codes in regular graphs are difficult, *Discrete Math.*, **133**, (1994), 191–205.

[KK88] J. Kratochvíl and M. Křivánek, On the computational complexity of codes in graphs., in *Lect. Notes Comp. Science*, Springer-Verlag, **324**, (1988), 396–404.

[KST94] H. Kaplan, R. Shamir, and R.E. Tarjan, Tractability of parameterized completion problems on chordal and interval graphs: minimum fill-in and DNA physical mapping, in *Proc. 35-th Annual Symp. Found. Computer Science*, (1994), 780–891.

[SS96] M. Sipser and D.A. Spielman, Expander codes, *IEEE Trans. Inform. Theory*, **42**, (1996), 1710–1722.

[Ste92] M. Steel, The complexity of reconstructing trees from qualitative characters and subtrees, *J. Classification*, **9**, (1992), 91–116.

[Ste93] J. Stern, Approximating the number of error locations within a constant ratio is NP-complete, *Lect. Notes Comp. Science*, Springer-Verlag, **673**, (1993), 325–331.

[Tan81] R.M. Tanner, A recursive approach to low-complexity codes, *IEEE Trans. Inform. Theory*, **27**, (1981), 533–547.

[vEB80] P. van Emde Boas, Another NP-complete partition problem and the complexity of computing short vectors in a lattice, Tech. Report 81–04, Dept. of Mathematics, Univ. of Amsterdam, 1980.

[Var97a] A. Vardy, Algorithmic complexity in coding theory and the minimum distance problem, in *Proc. 29-th Annual ACM Symp. Theory of Computing*, El Paso, TX., (1997), 92–109.

[Var97b] A. Vardy, The intractability of computing the minimum distance of a code, *IEEE Trans. Inform. Theory*, **43**, no. 6, (1997), to appear.