Fixed-Parameter Tractability of Error Correction in Graphical Linear Systems

Peter Damaschke[1], Ömer Eğecioğlu[2], and Leonid Molokov[1]

[1] Department of Computer Science and Engineering,
Chalmers University, 41296 Göteborg, Sweden
{ptr,molokov}@chalmers.se
[2] Department of Computer Science, University of California,
Santa Barbara, CA 93106-5110
omer@cs.ucsb.edu

Abstract. In an overdetermined and feasible system of linear equations $Ax = b$, let vector b be corrupted, in the way that at most k entries are off their true values. Assume that we can check in the restricted system given by any minimal dependent set of rows, the correctness of all corresponding values in b. Furthermore, A has only coefficients 0 and 1, with at most two 1s in each row. We wish to recover the correct values in b and x as much as possible. The problem arises in a certain chemical mixture inference application in molecular biology, where every observable reaction product stems from at most two candidate substances. After formalization we prove that the problem is NP-hard but fixed-parameter tractable in k. The FPT result relies on the small girth of certain graphs.

Keywords: sparse system of linear equations, error correction, girth, even cycle matroid, parameterized algorithm.

1 Introduction

Let $Ax = b$ be a system of m linear equations in n variables, over the real numbers. Suppose that $Ax = b$ was obtained from some feasible linear system $Ax = b'$ by changing at most k of the coefficients in the vector b'. That is, b differs from the unknown true vector b' in at most k positions, but we are not told which. The maximum number k of errors may or may not be known. Our goal is to recover the correct values in b (and x) as far as possible, using a certain correctness criterion for entries of b that will be introduced below.

We were led to the problem by an application where we wish to infer the amounts of chemical compounds in an unknown mixture. We can only measure amounts of products of chemical split reactions each of which can stem from one or more candidate substances. This is modelled by a system of linear equations $Ax = b$. Each row (equation) corresponds to a measured substance, and each column of A (resp. variable of x) corresponds to a candidate compound. A with entries a_{ij} is the incidence matrix, that is, $a_{ij} = 1$ if split product i appears in compound j, and $a_{ij} = 0$ else. The a_{ij} are known, and b_i is the measured amount of product i. Typically every split product is contained in very few candidate

S.K. Ghosh and T. Tokuyama (Eds.): WALCOM 2013, LNCS 7748, pp. 245–256, 2013.

compounds, hence the rows of A contain very few 1s. We are particularly interested in the reconstruction of protein mixtures after enzymatic digestion into peptides which can be identified and measured. Most peptides come from only one or two candidate proteins, and simulated protein digestion data suggest that already equations with at most two variables suffice to infer most of the protein amounts, provided that all measurements are correct [2]. A practical issue is that, as a result of experimental errors, some of the measured values in b may be corrupted. Without errors we would merely have to solve the linear system, which is even overdetermined. But in the presence of errors it is clear that any inference algorithm needs some assumptions about the number or the nature of errors, as well as the manner by which they can be detected. Here we will adopt what we call the *independent errors assumption*; see below.

For any set R of rows, let $A[R]$ be the matrix A restricted to R, and let $b[R]$ be the vector b restricted to the corresponding entries, in the rows of R. Our systems are overdetermined. Consider any subset C of rows that is linearly dependent and minimal with this property, that is, every proper subset of C is linearly independent. Following the terminology of matroid theory we call such C a *circuit*. (Note that circuits can be of any size up to $rank(A)+1$.) Every row of a circuit C is a unique linear combination of the other rows of C. It follows that, if $b[C]$ has exactly one false entry, then the system $A[C]x = b[C]$ is not feasible. However, if $b[C]$ has several false entries, these errors are unlikely to cancel out each other such that $A[C]x = b[C]$ remains feasible by chance. Since the false entries in b result from independent measurement errors, deviations follow some continuous probability distribution and are uncorrelated. Hence errors in a circuit lead almost surely to infeasibility. This motivates the following

Independent Errors Assumption: *Whenever C is a circuit and $A[C]x = b[C]$ is feasible (we also call the circuit C* balanced*), then all values in $b[C]$ are true.*

Note that minimality of C is essential here. The assumption trivially extends to unions of circuits, but not to arbitrary dependent sets. If some row in a dependent set D is not contained in any circuit with other rows of D, the row is independent of the rest of D, and then arbitrary changes of the corresponding entry of vector b will keep the system feasible. Our assumption is of similar spirit as the very common "general position" assumptions in computational geometry (e.g., no three points are on the same line, no three lines intersect in a point). The setting also resembles the combinatorial group testing problem where elements of a set can be faulty or clean, and one can test pools of elements, with the result that either the entire pool is clean, or some faults are present. But when faults are present, the test does not tell us what the faulty elements are. In our variant however, pools are restricted to circuits of some matroid. For ease of presentation we assume an idealized computational model with precise real numbers, in practice we must allow small tolerances when we check two numbers for equality. Our problem is now preliminarily stated as follows:

Balanced Circuits Recovery: Given a linear system $Ax = b$ where some unknown subset of the entries in b are faulty, identify all entries of vector b that can be confirmed true under the independent errors assumption.

After that, it only remains to solve the linear system restricted to the rows with confirmed entries of b. Two questions arise: Which entries of b can we recover, if we can recover any at all, and how difficult is this algorithmically?

We focus on the case where all coefficients in A are 0 or 1, and every row of A contains at most two entries that are 1. This problem is not as limited and specialized as it might seem. The restriction to $0,1$-matrices with sparse rows is immediately motivated by applications as above, and the independent errors assumption is not restrictive at all; loosely speaking it just says that random errors will not collectively appear correct by pure chance.

Organization of the Paper: First we have to put some work into the formal problem statement and terminology. In Section 2 we characterize the entries in b that are recoverable under the independent errors assumption. In the case of two variables per equation, our problem can be formalized as a graph labeling problem where the entries of x and b are turned into vertices and edges, respectively. We introduce some notions and facts about matroids from graphs, which are needed in the following (see also [11]). The graph formulation is essential for our algorithm. In Section 3 we prove NP-hardness of Balanced Circuits Recovery in graphs. Section 4 gives a preparation for an FPT algorithm, with the number k of faulty edges as the parameter: we show that, after some preprocessing, we always find a circuit of logarithmic size. This extends the known fact that graphs of constant minimum degree have logarithmic girth, but since circuits and cycles are different objects, the matter requires some care. What we have here is the *even cycle matroid* of the associated graph, for which the circuits are different than the cycles of the ordinary cycle matroid. Based on the girth we give our FPT result in Section 5, by constructing a kernel of $O(k \log k)$ vertices. In Section 6 we obtain an $O^*((\log k)^k)$ time bound. Section 7 lists some open questions. Due to space limitations some proofs and straightworward algorithm details are omitted.

Related Literature: In [2] we considered an error model where all b_i are changed by at most some small ϵ, and we gave graph-theoretic and LP methods for controlling the error in the solution vector x. The motivation for the present study is that, besides general measurement inaccuracies, a number k of measured amounts may be totally wrong and should be detected first. Only for ease of presentation we assume here that all non-faulty b_i are accurate. If they are slightly disturbed, then unbalanced circuits remain unbalanced, and "nearly" balanced circuits within some tolerance have to be considered as balanced.

Approximability and parameterized complexity of finding maximal feasible subsystems of linear systems is studied, e.g., in [4,5]. Like ours, this problem is of special interest in the case of graphs (e.g., for some models in statistical physics), and it can be generalized to so-called gain graphs where vertices and edges are labeled with elements of a group. The minimum number of unsatisfied edges is known as the *frustration index*, and its computation is NP-hard already in special cases. We refer to the extensive annotated bibliography in [15]. However, Balanced Circuits Recovery differs from this suite of problems in that

we made an additional mild assumption on the confirmation of correct edge labels. Next, the matroid circuits we have to deal with are even cycles and connected pairs of odd cycles (see details in Section 2), which loosely relates our problem to both feedback set problems [12,7,3] and odd cycle transversals (OCT) [13,7,9] whose parameterized complexity is well investigated. Remarkably, [8] uses matroids for kernelization, too. LP techniques as applied to OCT and other problems in [10] do not seem to be immediately applicable to our problem. In [1] we enumerated solutions with minimal support in linear systems with a constant number of nonzeros per row, however in an error-free setting.

2 Characterizations and Formalization

Remember that our input is a linear system $Ax = b$, where some entries of the observed b are faulty but obey the independent errors assumption.

Definition 1. *A row i is* correct *if b_i has its true value, otherwise it is* faulty. *A set of rows is* correct *if every row in that set is correct. A row i is* recoverable *if a unique value b_i is consistent with the independent errors assumption.*

Note that we cannot directly "see" which rows are correct, rather, we learn them only by checking circuits for being balanced. Row i being recoverable means informally that we could infer the true b_i, given enough computation time. Clearly, rows in balanced circuits are recoverable, and an obvious question is whether there exist more recoverable rows.

Definition 2. *We inductively define* reachable *rows as follows. A row in a balanced circuit is reachable; a single row in a circuit where all other rows are reachable is reachable, too; and no other rows are reachable.*

As mentioned, all rows in balanced circuits are recoverable. Next, any row i that appears in some circuit C where all other rows are recoverable, is recoverable, too. This follows from the fact that row i of matrix A is a unique linear combination of the other rows of C: since the true values in $b[C]$, perhaps except b_i, can be determined, we can finally determine the true b_i as well, thereby even ignoring the given value. In summary, all reachable rows are recoverable. The converse is also true, but due to space limitations we skip the proof.

Theorem 1. *The recoverable rows are exactly the reachable rows.* □

From now on we deal with the announced "graphical" case.

Definition 3. *Let $Ax = b$ be a system of m linear equations in n variables, with at most two variables per equation, that appear with coefficient* 1. *We represent it as a graph with n vertices and m edges as follows. Its vertices are the variables in x. For every equation (row) $x_u + x_v = b_i$, the graph has an edge uv with* edge label *$b_{uv} := b_i$. For every trivial equation $2x_u = b_i$ with only one variable, the graph has a loop with edge label $b_{uu} := b_i$. Variable x_v is also called the* vertex label *of v. The graph may comprise parallel edges and also several loops at the same vertex, since the given matrix A may have identical rows.*

We have multiplied the trivial equations by 2 (and doubled b_i) to give all equations the same form. Despite possible parallel edges the notation b_{uv} will not cause confusion, as it will be clear from context which edge we refer to.

Due to the correspondence established in the Definitions and Theorem 1 we can use the terms *row* and *edge* interchangeably and speak of *recoverable (reachable) edges*. Moreover we can state the following graph problem, whose complexity with respect to parameter k will be studied here.

Balanced Circuits Recovery in graphs: Given a graph $G = (V, E)$, possibly with parallel edges and loops, and an edge labeling b, identify all reachable edges. As for the parameter k, the following is assumed: There exists a labeling b' that differs from b on at most k edges called the faulty edges; G with labeling b' has only balanced circuits; and in G with labeling b, no circuit containing faulty edges is balanced.

Of course, it is essential to know which edge sets in the graph correspond to the circuits, i.e., minimal dependent sets of rows in the linear system.

Definition 4. *A path or cycle in a graph is* simple *if it does not cross itself, that is, every vertex appears at most once on it. The* length *of a path or cycle is the number of edges. We consider a loop as a simple odd cycle of length* 1*. A* bow tie *is either a pair of vertex-disjoint simple odd cycles connected by a simple path whose inner vertices do not appear in any of the two cycles, or a pair of simple odd cycles with exactly one common vertex.*

The following is implicit in earlier literature [14,6]. (In [6] one can also find an interesting treatment of the algebra of the even cycle matroid.)

Theorem 2. *The circuits are exactly the simple even cycles and bow ties.* □

The two types of circuits behave differently when it comes to the vertex labels. In a bow tie, the vertex labels are uniquely determined. This is because the incidence matrix of a simple odd cycle has a nonzero determinant. As opposed to this, the incidence matrix of a simple even cycle has determinant zero, and the possible vectors of vertex labels form a 1-dimensional space: We can alternatingly add/subtract some free value to/from the vertex labels.

For our algorithm we will need some "technical" generalization of graphs (which is well established in matroid theory, cf. signed graphs and gain graphs).

Definition 5. *A* signed graph *is a graph where every edge also has a sign, besides the edge label. A sign is even or odd. Signs can be added modulo 2 where even=0 and odd=1. Vertex and edge labels are related as follows. The label b_{uv} of an odd edge uv satisfies $b_{uv} = x_u + x_v$; note that $b_{uv} = b_{vu}$. The label d_{uv} of an even edge uv satisfies $d_{uv} = x_u - x_v$. Note that $d_{uv} = -d_{vu}$, that is, the orientation of an even edge matters.*

The operation of merging *edges in a signed graph works as follows. Let w be some vertex of degree 2 with neighbors u and v, where possibly $u = v$. We replace w and edges wu and wv with a new edge uv whose sign is the sum of signs of wu and wv. The label of uv is built according to these rules:*

If both wu and wv are odd, then uv is even, and
$d_{uv} = x_u - x_v = b_{uw} - x_w + x_w - b_{vw} = b_{uw} - b_{wv}$.
If both wu and wv are even, then uv is even, and
$d_{uv} = x_u - x_v = d_{uw} + x_w - x_w - d_{vw} = d_{uw} + d_{wv}$.
If wu is odd and wv is even, then uv is odd, and
$b_{uv} = x_u + x_v = b_{uw} - x_w + x_w + d_{vw} = b_{uw} - d_{wv}$.
If wu is even and wv is odd, then uv is odd, and
$b_{uv} = x_u + x_v = b_{vw} - x_w + x_w + d_{uw} = b_{vw} - d_{wu}$.

Note that:

(1) In terms of the linear system, merging just means to eliminate the variable x_w by combining the equations for the labels of wu and wv.

(2) We can consider the original graph as a signed graph where all signs are odd. After a sequence of mergings, an odd (even) edge can represent a path of odd (even) length with inner vertices of degree 2 in the original graph. It is straightforward to prove associativity: The label of an edge does not depend on the order the merge steps are applied to consecutive edges in a path. Also the notions of circuit and balanced circuit can now be lifted to signed graphs in a straightforward way. In particular we have:

Corollary 1. *The circuits in signed graphs are exactly the simple even cycles and bow ties, with the modification that the sign of a cycle (odd or even) is now the sum of signs of its edges.* □

3 NP-Hardness of Determining the Recoverable Edges

Theorem 3. Balanced Circuits Recovery *in graphs is NP-hard.*

Proof. We will demonstrate that the following decision problem is NP-complete: Given a graph with edge signs and edge labels and a specific edge, is this edge recoverable? Then the Theorem follows, because even edges are only used as a shorthand for a path of two odd edges. (An instance of Balanced Circuits Recovery has odd edges only.)

The well-known NP-complete Subset Sum takes as input $n+1$ real numbers $a_1, \ldots, a_n; s$ and asks whether $\sum_{i \in A} a_i = s$ holds for some subset $A \subseteq \{1, \ldots, n\}$. Given an instance of Subset Sum, we construct in linear time an instance of Balanced Circuit Recovery as follows (consult Fig.1 for an example of the reduction graph):

Create two vertices u and v, each with a loop with odd sign. The loop at u gets label 0, and the loop at v gets label $2s$. For $i = 1$ create vertices u_1, v_1 and two edges uu_1 and uv_1. For each item a_i, $i > 1$, from the sequence, create two vertices u_i, v_i and four edges $u_{i-1}u_i$, $u_{i-1}v_i$, $v_{i-1}u_i$, $v_{i-1}v_i$. Also create two edges u_nv, v_nv. All these non-loop edges are even. Edges get the following labels. Every edge u_iu_{i+1} and v_iu_{i+1} gets label 0, and every edge u_iv_{i+1} and v_iv_{i+1} gets label $-a_{i+1}$. Similarly, edge uu_1 gets label 0, and edge uv_1 gets label $-a_1$. Both u_nv and v_nv get label 0.

The idea is that every even edge, so to speak, shifts the vertex label by either a_{i+1} or 0 when we proceed from u to v. Based on this, we will show the following equivalence (remember what *recoverable* means, from Definition 1).

Claim. The two loops are recoverable if and only if the SUBSET SUM instance is a YES instance.

Proof of Claim. Trivially, the graph without the loops at u and v contains only even cycles. Due to Theorem 2, the circuits containing a loop are exactly the bow ties connecting the loops at u and v by a simple path. That is, the two loops can only appear together in circuits. By the inductive definition of reachable edges (rows) and Theorem 1, we can only infer edge labels in balanced circuits, and one further true edge label at a time, in a circuit where all other edges are already recovered. But we cannot infer two new edge labels in a circuit simultaneously. Hence the loops are recoverable if and only if they appear in a balanced circuit.

The 2^n shortest simple paths from u to v go through the vertices u_i or v_i strictly in the order of indices $i = 1, \ldots, n$. On a simple path from u to v we may also go from index $i+1$ back to index i, but then we have to return immediately to $i+1$, to avoid repeated visits of a vertex. Any such zig zag path of three edges can be replaced with the one edge between its end vertices with indices i and $i+1$. The shift of labels on the zig zag path and the single edge is the same; this is easy to verify by our choice of even edge labels and their effect on the vertex labels. (Note that the direct edge is the result of merging the three edges in the zig zag path.) Hence it suffices to consider only bow ties with shortest paths from u to v. Now, going through v_i means to add a_i to the solution A, and going through u_i means not to add a_i to A. Since the loop at v has label $2s = s + s$, the total shift must be exactly s, and the claimed equivalence is established. □

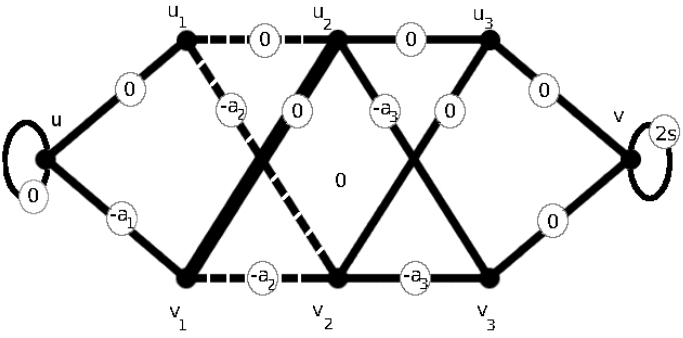

Fig. 1. The reduction graph (Theorem 3) for 3 items. The dashed lines show a zig zag path that may be replaced with the bold edge in a bow tie.

4 Girth of Reduced Signed Graphs

In view of the NP-hardness result in Theorem 3 it is natural to study the parameterized version of Balanced Circuits Recovery in graphs, with the maximum number k of faulty edges as the parameter. The following is a building block of our parameterized algorithm.

Given any signed graph with real-valued edge labels we construct the *reduced* signed graph by repeatedly applying the following steps as long as possible:

0-rule: If vertex w has a single loop with odd sign, and w is not adjacent with further edges, delete w and the loop.
1-rule: If w is a vertex of degree 1 with neighbor u, delete w and the edge wu.
2-rule: If w is a vertex of degree 2 with neighbors u and v (where possibly $u = v$), merge the edges wu and wv.

It is easy to establish that these rules can be applied in any order and yield a uniquely determined reduced signed graph. We also need:

Lemma 1. *In a signed graph G with n vertices and minimum degree $d \geq 3$ we find a circuit with at most $4\log_{d-1} n+2$ edges in polynomial time. Consequently, in a reduced signed graph with n vertices we find a circuit with at most $4\log_2 n+2$ edges in polynomial time.*

Proof. We only sketch the construction, details are omitted due to space limitations. By breadth-first search we easily find a cycle C of logarithmic length. If C is even, we are done. If C is odd, we shrink C to a super vertex and run BFS again, to find another cycle and a path connecting it to C, both of logarithmic length. Finally note Corollary 1. □

5 Parameterized Algorithm for the Recoverable Edges

Suppose that we have identified a balanced circuit C. Then the edges in C are recovered, due to the independent errors assumption. Furthermore we can *contract* C by successively contracting its edges.

Edge Contraction: Once an edge uv is recognized as correct, we can *contract* it and obtain an equivalent smaller problem instance. The principle is simply to eliminate one of the variables, say x_v, which is possible since we know that the edge label is correct. For clarity we discuss all details as needed later.

If the considered edge is a loop ($u = v$), we have the following cases. If the loop is even, then $d_{uu} = 0$ (otherwise the loop would not be correct), and we can simply delete the loop. If the loop is odd, then $x_u = b_{uu}/2$ is enforced, thus we keep the loop as an indicator that the x_u has been determined. In the following we suppose $u \neq v$ and discuss the real edge contraction. We keep vertex u and eliminate v; this choice is arbitrary. Let w denote any further vertex. When we contract uv, all edges uw ($w \neq v$) are unchanged. We merge uv with every edge vw ($w \neq v$) exactly as described earlier in Section 2. Note that at least one such edge exists, since otherwise uv is in no circuit (by Theorem 2), hence it would

never be confirmed as correct. It remains to consider odd loops at v (whereas even loops are meaningless, as seen before). Any odd loop at v, with label b_{vv}, is transformed into an odd loop at u with the following label b_{uu}.

If uv is odd, then $b_{uu} = 2x_u = 2b_{uv} - 2x_v = 2b_{uv} - b_{vv}$.
If uv is even, then $b_{uu} = 2x_u = 2d_{uv} + 2x_v = 2d_{uv} + b_{vv}$.

It may be helpful to notice what the final result of *contracting a balanced circuit* C is. If C is a simple even cycle, there remains one vertex without loop, that is, the label of that vertex is not "internally" determined by C, corresponding to the 1-dimensional solution space of a simple even cycle. If C is a bow tie, we eventually get one vertex with two odd loops attached, however they have equal edge labels as C is balanced, hence one of them is redundant. – We are ready to give a high-level description of our algorithm for Balanced Circuits Recovery in graphs.

Preprocessing: In the following we work with signed graphs. Initially, every edge gets an odd sign. First we apply the 1-rule as long as possible. From Theorem 1 and Corollary 1 we get: The removed edges do not appear in any circuit, hence they are not recoverable nor can they contribute to recovery of other edge labels. Clearly, we can ignore them henceforth. Next we also apply the 2-rule and 0-rule as long as possible, hence we obtain the reduced signed graph.

Main Loop of the Algorithm: We apply Lemma 1 to find a circuit C with at most $4\log_2 n + O(1)$ edges. We check whether C is balanced, by testing whether the linear system induced by the edges in C is feasible.

If C is balanced, we contract C. Contraction does not alter the degrees of vertices outside C, only some edges incident with C may get new end vertices and adjusted labels. (Remember that parallel edges are allowed, they are *not* removed from our graphs. While contraction of edges may render other edges parallel, these edges are still kept, hence the degrees of vertices outside C are *not* diminished.) In particular, the graph is still a reduced signed graph after contraction, hence no merging takes places among the remaining edges.

If C is not balanced, clearly C contains *some* faulty edge, but it is important to notice that we do not know which edges are faulty. Therefore, in this case we delete *all* edges of C (but not their vertices) from the graph, and then we reduce the remaining graph again, by exhaustively applying the three rules.

We iterate the process of circuit detection, contraction, deletion and reduction, until the remaining graph has some constant size $O(1)$.

Re-inserting Edges: As said in the beginning of this section, contraction only yields equivalent problem instances, and vertices eliminated during the contraction of balanced circuits have labels uniquely determined by the labels of the vertices we keep, hence we need not consider them any more. Roughly speaking, no information gets lost by contraction.

The situation is different for faulty circuits. Since we have removed edges that are only "potentially" faulty, we must eventually re-insert them one by one to guarantee an equivalent problem instance. The details of this step need

some more discussion. In the following, G denotes the input graph after initial exhaustive application of the 1-rule (hence G has minimum degree 2), and H denotes the graph obtained by our processing so far. Let $e = uv$ be some edge that we want to put back next. Recall that any vertex of H may stand for a vertex of G or represent a subset of original vertices of G identified due to edge contractions. Similarly, any edge of H may stand for an original edge of G or a path of merged edges from G. Thus, any end of e, say u, may lie in a vertex of H or on an edge of H, or u may even lie outside H, because the edges adjacent to u have been removed as well (e.g., by the 1-rule). If e is already on a path represented by an edge of H, clearly we need not re-insert e.

Now we treat the other cases. If u is in a vertex of H, we simply attach e to this vertex in the obvious sense. If u is on an edge f of H, we insert vertex u in H and subdivide f. If u is not in H, then due to minimum degree 2 another path in G different from edge uv starts in u and ends somewhere in H, or in v. We also re-insert such a path, but then exhaustively apply the 2-rule again. A few simple case inspections show that, in either case regarding the ends u and v of e, re-insertion of an edge e adds at most 2 vertices and 2 further edges to H. With at most k faulty edges, the size of H thus increases by $O(k \log n)$ in total, due to Lemma 1. (After each re-insertion we can also update the edge labels in H straightforwardly, since the original edge labels from G are still known.)

Analysis: By the last observation we can reduce a graph with n vertices to a kernel of $k \log n$ vertices (with some fixed logarithm base) in polynomial time. Thus an upper bound n on the kernel size is implicitly given by $n = k \log n$. In order to bound n in terms of k only, observe that $n = k \log n$ implies $k > \log n$ (for large enough n). Therefore $n = k \log n = k(\log k + \log \log n) = O(k \log k)$.

Work on the Kernel: The above procedure has computed, in polynomial time, a kernel of $O(k \log k)$ vertices, with the property that every edge outside the kernel is already recovered or not recoverable at all. Thus it only remains to solve the problem on the kernel. Even if we naively enumerate all circuits in the kernel, test them, and get the other recoverable edges inductively (Theorem 1), the time depends only on k. Thus we have finally shown:

Theorem 4. Balanced Circuits Recovery *in graphs parameterized by the number of faulty edges is in FPT.* □

6 Branching Strategies

Theorem 4 establishes our FPT result, however exhaustive enumeration of all circuits in the kernel would be wasteful. We consider more efficient strategies for this last phase separately. Before we solve the remaining problem on the kernel, we reduce this graph once more: Isolated odd loops removed by the 0-rule and edges removed by the 1-rule are not recoverable anyhow, and in every path P of edges merged by the 2-rule, either all or none of the edges in P are recoverable, such that we need not distinguish them. Also remember that contracting balanced circuits only removes edges that are already recovered

(since the independent errors assumption holds, and edges that became parallel are kept in the graph). In parameterized time bounds we use the O^* notation that omits polynomial factors, and log means the logarithm with a suitable base.

Theorem 5. Balanced Circuits Recovery *in graphs with n vertices and m edges, at most k of them faulty, is solvable within the minimum of the following time bounds: $O^*(2^m)$; $O^*(m^k/k!)$; and $O^*((\log n)^k)$.*

Proof. We may guess the subset of faulty edges and delete them. Clearly, these are at most 2^m and at most $m^k/k!$ subsets. In every branch we run the algorithm from Section 5, with the difference that we abort it when a faulty circuit is detected. Every branch needs polynomial time.

Alternatively, we may run the algorithm from Section 5, now with the difference that, when a faulty circuit C is detected, we branch on C, thereby guessing only one faulty edge and deleting it. Due to Lemma 1, C has $\log n$ edges. Since we can apply this step at most k times, we obtain a bounded search tree of size $(\log n)^k$. Clearly, if some branch is successful, we have found all recoverable edges, otherwise we report that more than k faulty edges were present.

A side remark is that the reachable edges that are not in balanced circuits (see the definition of reachable rows/edges and Theorem 1) are eventually detected, as they are linearly dependent from already recovered edges. □

Since we have $n = O(k \log k)$ as said above, we obtain $O^*((\log n)^k) = O^*((\log k)^k) = O^*(c^{k \log\log k})$ for some constant $c > 1$. Hence the latter method is faster unless $m = O(k \log\log k)$.

Corollary 2. Balanced Circuits Recovery *in graphs with at most k faulty edges is solvable in $O^*((\log k)^k)$ time.* □

7 Conclusions

We proved NP-hardness and gave an FPT algorithm for the problem of recovering the entries of vector b in a linear system $Ax = b$, where A is a 0,1-matrix with at most two 1s per row, assuming that b has at most k errors which are uncorrelated in a sense. The problem is motivated by the inference of chemical mixtures under measurement errors, and can be rephrased as a graph problem. Already membership in FPT is not trivial. Some obvious open questions are: Can we improve the FPT time bound, possibly by using stronger relations between edge number and girth? What about matrices with general nonzero coefficients, and with some more than two nonzeros per row? (Do the resulting hypergraph problems inherit some of the useful graph structure?) Can we generalize the FPT approach to gain graphs [15], provided that there exist natural applications? The complexity of approximation might be interesting, too, but this was not the scope of this paper. Finally, it would be worthwhile to apply the algorithms to real protein quantitation data and validate the error assumptions made in the parameterization.

Acknowledgment. This work has been supported by the Swedish Research Council (Vetenskapsrådet), grant no. 2010-4661, "Generalized and fast search strategies for parameterized problems". Early stages of the third author's work have also been supported by Devdatt Dubhashi through a Chalmers Bioscience Initiative grant. The work was done while the second author was visiting Chalmers during his sabbatical 2011–2012.

References

1. Damaschke, P.: Sparse Solutions of Sparse Linear Systems: Fixed-Parameter Tractability and an Application of Complex Group Testing. In: Marx, D., Rossmanith, P. (eds.) IPEC 2011. LNCS, vol. 7112, pp. 94–105. Springer, Heidelberg (2012); Extended version to appear in Theor. Comp. Sci.
2. Damaschke, P., Molokov, L.: Error Propagation in Sparse Linear Systems with Peptide-Protein Incidence Matrices. In: Bleris, L., Măndoiu, I., Schwartz, R., Wang, J. (eds.) ISBRA 2012. LNCS, vol. 7292, pp. 72–83. Springer, Heidelberg (2012)
3. Dehne, F.K., Fellows, M.R., Langston, M.A., Rosamond, F.A., Stevens, K.: An $O(2^{O(k)}n^3)$ FPT Algorithm for the Undirected Feedback Vertex Set Problem. Theory Comput. Syst. 41, 479–492 (2007)
4. Feige, U., Reichman, D.: On the Hardness of Approximating Max-Satisfy. Info. Proc. Lett. 97, 31–35 (2006)
5. Giannopoulos, P., Knauer, C., Rote, G.: The Parameterized Complexity of Some Geometric Problems in Unbounded Dimension. In: Chen, J., Fomin, F.V. (eds.) IWPEC 2009. LNCS, vol. 5917, pp. 198–209. Springer, Heidelberg (2009)
6. Grossman, J.W., Kulkarni, D.M., Schochetman, I.E.: Algebraic Graph Theory Without Orientation. Lin. Algebra and its Appl. 212/213, 289–307 (1994)
7. Guo, J., Gramm, J., Hüffner, F., Niedermeier, R., Wernicke, S.: Compression-Based Fixed-Parameter Algorithms for Feedback Vertex Set and Edge Bipartization. J. Comput. Syst. Sci. 72, 1386–1396 (2006)
8. Kratsch, S., Wahlström, M.: Compression via Matroids: A Randomized Polynomial Kernel for Odd Cycle Transversal. In: Rabani, Y. (ed.) SODA 2012, pp. 94–103. SIAM (2012)
9. Lokshtanov, D., Saurabh, S., Sikdar, S.: Simpler Parameterized Algorithm for OCT. In: Fiala, J., Kratochvíl, J., Miller, M. (eds.) IWOCA 2009. LNCS, vol. 5874, pp. 380–384. Springer, Heidelberg (2009)
10. Narayanaswamy, N.S., Raman, V., Ramanujan, M.S., Saurabh, S.: LP can be a Cure for Parameterized Problems. In: Dürr, C., Wilke, T. (eds.) STACS 2012. LIPIcs, vol. 14, pp. 338–349 (2012)
11. Oxley, J.: Matroid Theory, 2nd edn. Oxford Univ. Press (2011)
12. Raman, V., Saurabh, S., Subramanian, C.R.: Faster Fixed Parameter Tractable Algorithms for Finding Feedback Vertex Sets. ACM Trans. Algor. 2, 403–415 (2006)
13. Reed, B.A., Smith, K., Vetta, A.: Finding Odd Cycle Transversals. Oper. Res. Lett. 32, 299–301 (2004)
14. Tutte, W.T.: On Chain-Groups and the Factors of Graphs. Coll. Math. Societatis János Bolyai 25 (Algebraic Methods in Graph Theory, Szeged), 793–818 (1978)
15. Zaslavsky, T.: A Mathematical Bibliography of Signed and Gain Graphs and Allied Areas. Electron. J. Comb., Dynamic Surveys in Combinatorics, no. DS8 (1999)