Parameterized Resiliency Problems via Integer Linear Programming*

Jason Crampton[1], Gregory Gutin[1], Martin Koutecký[2], and Rémi Watrigant[3]

[1] Royal Holloway, University of London, Egham, Surrey, TW20 0EX, UK
{jav.crampton, greg.gutin}@gmail.com
[2] Department of Applied Mathematics, Charles University, Prague, Czech Republic
koutecky@kam.mff.cuni.cz
[3] INRIA Sophia-Antipolis, 06902 Sophia-Antipolis Cedex, France
rwatriga@gmail.com

Abstract. We introduce a framework in parameterized algorithms whose purpose is to solve resiliency versions of decision problems. In resiliency problems, the goal is to decide whether an instance remains positive after *any* (appropriately defined) perturbation has been applied to it. To tackle these kinds of problems, some of which might be of practical interest, we introduce a notion of resiliency for Integer Linear Programs (ILP). We prove that ILP Resiliency is fixed-parameter tractable (FPT) under a certain parameterization.
To demonstrate the utility of our result, we prove that resiliency versions of several concrete problems are FPT under natural parameterizations. Our first result, for a problem which is of interest in access control, subsumes several FPT results and solves an open question from Crampton *et al.* (AAIM 2016). The second concerns the Closest String problem, for which we identify and solve two different resiliency problems, extending an FPT result of Gramm *et al.* (2003). We also consider problems in the fields of scheduling and social choice. We believe that many other problems can be tackled by our framework.

1 Introduction

Questions of ILP feasibility are typically answered by finding an integral assignment of variables x satisfying $Ax \leq b$. By Lenstra's theorem [17], this problem can be solved in $O^*(f(n)) := O(f(n)L^{O(1)})$ time and space, where f is a function of the number of variables n only, and L is the size of the ILP (subsequent research has obtained an algorithm of the above running time with $f(n) = n^{O(n)}$ and using polynomial space [11, 16]). In the language of parameterized complexity, this means that ILP Feasibility is fixed-parameter tractable (FPT) parameterized by the number of variables. Note that there are a number of parameterized problems for which the only (known) way to prove fixed-parameter

* Gutin's research was supported by Royal Society Wolfson Research Merit Award. Koutecký's research was supported by projects 14-10003S of GA ČR and 338216 of GA UK.

tractability is to use Lenstra's theorem[1] [6]. For more details on this topic, we refer the reader to [6, 8].

The notion of *resiliency* measures the extent to which a system can tolerate modifications to its configuration and still satisfy given criteria. An organization might, for example, wish to know whether it will still be able to continue functioning, even if some of its staff become unavailable. In the language of decision problems, we would like to know whether an instance is still positive after *any* (appropriately defined) modification. Intuitively, the resiliency version of a problem is likely to be harder than the problem itself; a naive algorithm would consider every allowed modification of the input, and then see whether a solution exists.

In this paper, we introduce a framework for dealing with resiliency problems, and study their computational complexity through the lens of fixed-parameter tractability. We define resiliency for Integer Linear Programs (ILP) and prove fixed-parameter tractability of this new problem under a certain parameter. Our proof uses the main result of [9]. To illustrate the fact that our approach might be useful in different situations, we apply our framework to several concrete problems.

Crampton *et al.* analyzed the parameterized tractability of the Resiliency Checking Problem (RCP) [4], which has practical applications in the context of access control [18]. RCP can be seen as a resiliency version of a generalization of the Set Cover problem, and has five parameters n, p, s, d and t (described in more detail in Section 3), among which n is assumed to be large in practice, relative to the other four parameters [18]. Thus, it is natural to consider only p, s, d and t as parameters of RCP [4, 5].

Using well-known tools in parameterized algorithms, Crampton *et al.* [4] were able[2] to determine the complexity of RCP (FPT, XP, W[2]-hard, para-NP-hard or para-coNP hard) for all but two combinations of p, s, d and t (these two combinations are p and p, t)[3]. In particular, in the case where $s = 0$ (when no resiliency is considered), they proved, using Lenstra's theorem, that RCP is FPT parameterized by p (and thus by p, t). However, they could not extend this result to the case of any s, and thus the complexity of RCP parameterized by p was left open. We settle this case in this paper by showing that, in general, RCP is FPT parameterized by p. This result gives the complete picture of the parameterized complexity of RCP depending on the considered parameter.

We introduce an extension of the Closest String problem, a problem arising in computational biology. Informally, Closest String asks whether there exists a string that is "sufficiently close" to each member of a set of input strings.

[1] Lenstra's theorem allows us to prove a mainly classification result, i.e. the FPT algorithm is unlikely to be efficient in practice, nevertheless Lenstra's theorem indicates that efficient FPT algorithms are a possibility, at least for subproblems of the problem under considerations.

[2] We have also established that certain sub-cases of RCP are FPT using reductions to the Workflow Satisfiability Problem [5].

[3] By definition, a problem with several parameters $p_1, \dots, p_\ell$ is the problem with one parameter, the sum $p_1 + \cdots + p_\ell$.

We modify the problem so that the input strings may be unreliable – due to transcription errors, for example – and show that this resiliency version of Closest String called Global Resiliency Closest String is FPT when parameterized by the number of input strings. We consider another extension of Closest String when only specific positions of the strings can be affected and prove that this version is also FPT when parameterized by the number of input strings. Our resiliency results on Closest String are generalizations of a result of Gramm *et al.* for Closest String which was proved using Lenstra's theorem [12][4].

We introduce a resiliency version of the scheduling problem of makespan minimization on unrelated machines. We prove that this version is FPT when parameterized by the number of machines, the number of job types and the total expected downtime, generalizing a result of Mnich and Wiese [20] provided the jobs processing times are upper-bounded by a number given in unary.

Finally, we introduce a resilient swap bribery problem in the field of social choice and prove that it is FPT when parameterized by the number of candidates.

The remainder of the paper is structured in the following way. Section 2 introduces ILP resiliency and proves that it is FPT under a certain parameterization. We then apply our framework to a number of concrete problems. We establish the fixed-parameter tractability of RCP parameterized by p in Section 3. In Section 4 and Appendix B, we introduce two resiliency versions of Closest String Problem and prove that they are FPT. We study resiliency versions of scheduling and social choice problems in Sections 5 and 6. We conclude the paper in Section 7, where we discuss related literature. Proofs of all results marked by [$\star$] are given in Appendix A.

2 ILP resiliency

Recall that questions of ILP feasibility are typically answered by finding an integral assignment of variables x satisfying $Ax \leq b$. Let us introduce resiliency for ILP as follows. We add another set of variables z, which can be seen as "resiliency variables". We then consider the following ILP[5] denoted by $\mathcal{R}$:

$$Ax \leq b \tag{1}$$

$$Cx + Dz \leq e \tag{2}$$

$$Fz \leq g \tag{3}$$

The idea is that inequalities (1) and (2) represent the intrinsic structure of the problem, among which inequalities (2) represent how the resiliency variables modify the instance. Inequalities (3), finally, represent the structure of the resiliency part. The goal of ILP Resiliency is to decide whether $\mathcal{R}$ is *z-resilient*,

[4] Although not being strictly the first problem proved to be FPT using Lenstra's theorem (see, [22] for instance), it is considered as the one which popularized this technique [6, 8].

[5] To save space, we will always implicitly assume that integrality constraints are part of every ILP of this paper.

i.e. whether for *any* integral assignment of variables z satisfying inequalities (3), there exists an integral assignment of variables x satisfying (1) and (2).

In $\mathcal{R}$, we will assume that all entries of matrices in the left hand sides and vectors in the right hand sides are rational numbers. The dimensions of the vectors x and z will be denoted by n and p, respectively, and the total number of rows in A and C will be denoted by m. Let $\kappa(\mathcal{R}) := n + p + m$.

Our main result establishes that ILP Resiliency is FPT when parameterized by $\kappa(\mathcal{R})$, provided that part of the input is given in unary. Our method offers a generic framework to capture many situations. Firstly, it applies to ILP, a general and powerful model for representing many combinatorial problems. Secondly, the resiliency part of each problem can be represented as a whole ILP with its own variables and constraints, instead of, say, a simple additive term. Hence, we believe that our method can be applied to many other problems, as well as many different and intricate definitions of resiliency.

To prove our main result we will use the work of Eisenbrand and Shmonin [9]. For a rational polyhedron $Q \subseteq \mathbb{R}^{m+p}$, define $Q/\mathbb{Z}^p := \{h \in \mathbb{Q}^m : (h, \alpha) \in Q \text{ for some } \alpha \in \mathbb{Z}^p\}$. The Parametric Integer Linear Programming (PILP) problem takes as input a rational matrix $J \in \mathbb{Q}^{m \times n}$ and a rational polyhedron $Q \subseteq \mathbb{R}^{m+p}$, and asks whether the following expression is true:

$$\forall h \in Q/\mathbb{Z}^p \quad \exists x \in \mathbb{Z}^n : \quad Jx \leq h$$

Eisenbrand and Shmonin [9, Theorem 4.2] proved that PILP is solvable in polynomial time if the number of variables $n + p$ is fixed. A deeper analysis of their algorithm allows us to obtain a fined-grained version of their result:

Theorem 1. PILP *can be solved in time* $O^*(g(n, m, p)\varphi^{f(n,m,p)})$*, where* $\varphi \geq 2$ *is an upper bound on the encoding length of entries of* J *and* f *and* g *are some computable functions.*

Complexity remark. The polyhedron Q in Theorem 1 can be viewed as being defined by a system $Rh + S\alpha \leq t$, where $h \in \mathbb{R}^m$ and $\alpha \in \mathbb{Z}^p$. Then the algorithm of the theorem runs in time polynomial in the encoding lengths of R, S, and in m (the "continuous dimension"), and is FPT with respect to p (the "integer dimension").

Corollary 1. *If* n, m *and* p *are the parameters and all entries of* J *are given in unary, then* PILP *is FPT.*

Proof. We may assume that there is an upper bound $N \geq 2$ on the absolute values of entries of A and N is given in unary. Thus, the running time of the algorithm of Theorem 1 is $O^*(g(n, m, p)(\log N)^F)$, where $F = f(n, m, p)$.

It was shown in [3] that $(\log N)^F \leq (2F \log F)^F + N/2^F$, which concludes the proof. □

We now prove the main result of our framework, which will be applied in the next sections to two concrete problems.

Theorem 2. ILP Resiliency *is FPT when parameterized by* $\kappa(\mathcal{R})$ *provided the entries of matrices* A *and* C *are given in unary.*

Proof. We will reduce ILP Resiliency to Parametric Integer Linear Programming. Let us first define J and Q. Let $h = (h^1, h^2)$ with h^1 and h^2 being m_1 and m_2 dimensional vectors, respectively. Then the polyhedron Q is defined as follows: $h^1 = b, h^2 = e - D\alpha, F\alpha \leq g$. Furthermore, J is defined as: $Ax \leq h^1,\ Cx \leq h^2$.

Recall that $h^1 = b$ and $h^2 = e - D\alpha$ and α satisfies $F\alpha \leq g$, so for all $h \in Q/\mathbb{Z}^p$ there exists an integral x satisfying the above if and only if for all z satisfying $Fz \leq g$, there is an integral x satisfying (1) and (2). Moreover, the dimension of x is n, the integer dimension of Q is p and the number of inequalities of J is $m_1 + m_2 = m$, so applying Corollary 1 indeed yields the required FPT algorithm. □

3 Resiliency in Access Control

Access control is an important topic in computer security and is typically achieved by enforcing a policy that specifies which users are authorized to access which resources. Authorization policies are frequently augmented by additional policies, articulating concerns such as separation of duty and resiliency. The Resiliency Checking Problem (RCP) was introduced by Li *et al.* [18] and asks whether it is always possible to allocate authorized users to teams, even if some users are unavailable.

3.1 Definition of the Problem

Given a set of users U and set of resources R, an *authorization policy* is a relation $UR \subseteq U \times R$; we say u is *authorized* for resource r if $(u, r) \in UR$. For a user $u \in U$, we define $N_{UR}(u) = \{r \in R : (u, r) \in UR\}$, the *neighborhood* of u; by extension, for $V \subseteq U$, we define $N_{UR}(V) = \bigcup_{u \in V} N_{UR}(u)$, the *neighborhood* of V. Thus $N_{UR}(u)$ represents the resources for which u is authorized, and $N_{UR}(V)$ represents the resources for which the users in V are collectively authorized. We will omit the subscript UR if the authorization policy is clear from the context.

Given an authorization policy $UR \subseteq U \times R$, an instance of the Resiliency Checking Problem (RCP) is defined by a resiliency policy $\mathsf{res}(P, s, d, t)$, where $P \subseteq R$, $s \geq 0$, $d \geq 1$ and $t \geq 1$. We say that UR *satisfies* $\mathsf{res}(P, s, d, t)$ if and only if for every subset $S \subseteq U$ of at most s users, there exist d pairwise disjoint subsets of users $V_1, \ldots, V_d$ such that for all $i \in \{1, \ldots, d\}$:

$$V_i \cap S = \emptyset, \tag{4}$$

$$|V_i| \leq t \text{ and } N(V_i) \supseteq P. \tag{5}$$

In other words, UR satisfies $\mathsf{res}(P, s, d, t)$ if we can find d disjoint groups of users, even if up to s users are unavailable, such that each group contains no more than

t users and the users in each group are collectively authorized for the resources in P (observe that the particular case in which $s = 0$ and $d = 1$ is equivalent to the well-known SET COVER problem) Thus, we define RCP as follows:

RESILIENCY CHECKING PROBLEM (RCP)
Input: $UR \subseteq U \times R$, $P \subseteq R$, $s \geq 0$, $d \geq 1$, $t \geq 1$.
Question: Does UR satisfy $\mathsf{res}(P, s, d, t)$?

In the remainder of this section, we set $p = |P|$. Given an instance of RCP, we say that a set of d pairwise disjoint subsets of users $V = \{V_1, \ldots, V_d\}$ satisfying conditions (5) is a *set of teams*. For such a set of teams, we define $\mathcal{U}(V) = \bigcup_{i=1}^{d} V_i$. Given $U' \subseteq U$, the *restriction* of UR to U' is defined by $UR|_{U'} = UR \cap (U' \times R)$. Finally, a set of users $S \subseteq U$ is called a *blocker set* if for every set of teams $V = \{V_1, \ldots, V_d\}$, we have $\mathcal{U}(V) \cap S \neq \emptyset$. Equivalently, observe that S is a blocker set if and only if $UR|_{U \setminus S}$ does not satisfy $\mathsf{res}(P, 0, d, t)$.

3.2 Fixed-Parameter Tractability of RCP

In this section we prove that RCP is FPT parameterized by p. We first introduce some notation. In the following, $UR \subseteq U \times R$, $P \subseteq R$, $s \geq 0$, $d \geq 1$ and $t \geq 1$ will denote an input of RCP. Without loss of generality, we may assume $P = R$ and $N(u) \neq \emptyset$ for all $u \in U$. For all $N \subseteq P$, let $U_N = \{u \in U : N(u) = N\}$ (notice that we may have $U_N = \emptyset$ for some $N \subseteq P$).

Roughly speaking, the idea is that in order to construct a set of teams or a blocker set, it is sufficient to know the size of its intersection with U_N, for every $N \subseteq P$. We first define the set of *configurations*.

$$\mathcal{C} = \left\{ \{N_1, \ldots, N_b\} : b \leq t, N_i \subseteq P, i \in [b], \bigcup_{i=1}^{b} N_i = P \right\}.$$

Then, for any $N \subseteq P$, we denote the set of configurations involving N by $\mathcal{C}_N$. That is

$$\mathcal{C}_N = \{c = \{N_1, \ldots, N_{b_c}\} \in \mathcal{C} : N = N_i \text{ for some } i \in [b_c]\}$$

Observe that since we assume $t \leq p$, we have $|\mathcal{C}| = O(2^{p^2})$. The link between sets of teams and configurations comes from the following definition: given a set of teams V, we say that a team $T \in V$ *has configuration* $c \in \mathcal{C}$ if $c = \{N(u), u \in T\}$. In other words, c represents the distinct neighborhoods of users of T in P.

We define an ILP $\mathcal{L}$ over the set of variables $x = (x_c : c \in \mathcal{C})$ and $z = (z_N : N \subseteq P)$, with the following inequalities:

$$\textstyle\sum_{c \in \mathcal{C}} x_c \geq d \qquad (6)$$

$$\textstyle\sum_{N \subseteq P} z_N \leq s \qquad (7)$$

$$\textstyle\sum_{c \in \mathcal{C}_N} x_c \leq |U_N| - z_N \quad \text{for every } N \subseteq P \qquad (8)$$

$$0 \leq z_N \leq |U_N| \quad \text{for every } N \subseteq P \qquad (9)$$

$$0 \leq x_c \leq d \quad \text{for every } c \in \mathcal{C} \qquad (10)$$

Observe that $\kappa(\mathcal{L})$ is upper bounded by a function of p only. The idea behind this model is to represent a set S of at most s users by variables z (by deciding how many users to take for each set of users U_N, $N \subseteq P$), and to represent a set of teams by variables x (by deciding how many teams will have configuration $c \in \mathcal{C}$). Then, inequalities (8) will ensure that the set of teams does not intersect with the chosen set S. However, while we would be able to solve $\mathcal{L}$ in FPT time parameterized by p by using, *e.g.*, Lenstra's ILP Theorem, the reader might realize that doing so would not solve RCP directly. Nevertheless, the following result establishes the crucial link between this system and our problem.

Lemma 1. [$\star$] $\mathsf{res}(P, s, d, t)$ *is satisfiable if and only if* $\mathcal{L}$ *is* z*-resilient.*

Since, as we observed earlier, $\kappa(\mathcal{L})$ is bounded by a function of p only, combining Lemma 1 with Theorem 2, we obtain the following:

Theorem 3. RCP *is FPT parameterized by* p*.*

4 Closest String Problem

In the Closest String problem, we are given a collection of k strings $s_1, \ldots, s_k$ of length L over a fixed alphabet Σ, and a non-negative integer d. The goal is to decide whether there exists a string s (of length L) such that $d_H(s, s_i) \leq d$ for all $i \in [k]$, where $d_H(s, s_i)$ denotes the Hamming distance between s and s_i. If such a string exists, then it will be called a *d-closest string*.

It is common to represent an instance of the problem as a matrix C with k rows and L columns (*i.e.* where each row is a string of the input); hence, in the following, the term *column* will refer to a column of this matrix. As Gramm *et al.* [12] observe, as the Hamming distance is measured column-wise, one can identify some columns sharing the same structure. Let $\Sigma = \{\varphi_1, \ldots, \varphi_{|\Sigma|}\}$. Gramm *et al.* show [12] that after a simple preprocessing of the instance, we may assume that for every column c of C, φ_i is the i^{th} character that appears the most often (in c), for $i \in \{1, \ldots, |\Sigma|\}$ (ties broken *w.r.t.* the considered ordering of Σ). Such a preprocessed column will be called *normalized*, and by extension, a matrix consisting of normalized columns will be called *normalized*. One can observe that after this preprocessing, the number of different columns (called *column type*) is bounded by a function of k only, namely by the k^{th} Bell number $\mathcal{B}_k = O(2^{k \log_2 k})$. The set of all column types is denoted by T. Using this observation, Gramm *et al.* [12] prove that Closest String is FPT parameterized by k, using an ILP with a number of variables depending on k only, and then applying Lenstra's theorem.

4.1 Adding Resiliency

The motivation for studying resiliency with respect to this problem is the introduction of experimental errors, which may change the input strings [21]. While a solution of the Closest String problem tests whether the input strings are consistent, a resiliency version asks whether these strings will remain consistent after some small changes. However, there exist several ways to define how these changes will modify the input. We consider two versions of resiliency for Closest String. The first one, called Global Resiliency Closest String, allows at most m changes to appear anywhere in the matrix C. In the second version, called Column Resiliency Closest String and studied in Appendix B, we allow changes to appear column-wise. This situation might be useful if experimental errors occur more often at, say, the beginning or end of the string. We prove that both problems remain FPT parameterized by the number of input strings, generalizing in two different ways the result of Gramm *et al.* [12].

4.2 Global Resiliency Closest String

The most natural way of defining a notion of resiliency in the context of Closest String is to allow changes at any places of the matrix C. The only constraint is thus an upper bound on the number of total changes. To represent this, we simply use the Hamming distance between two matrices.

> Global Resiliency Closest String
> Input: C, a $k \times L$ normalized matrix of elements of Σ, $d \in \mathbb{N}$, $m \leq kL$.
> Question: For every C', $k \times L$ normalized matrix of elements of Σ such that the Hamming distance of C and C' is at most m, does C' admit a d-closest string?

ILP formulation. Let $\#_t$ be the number of columns of type t in C. For two types $t, t' \in T$ let $\delta(t, t')$ be their Hamming distance (the number of different elements). Let $z_{t,t'}$, for all $t, t' \in T$, be a variable meaning "how many columns of type t in C are changed to type t' in C'" (we allow $t = t'$). Thus we have the following constraints:

$$\sum_{t' \in T} z_{t,t'} = \#_t \qquad \forall t \in T \tag{11}$$

$$\sum_{t,t' \in T} \delta(t, t') z_{t,t'} \leq m \tag{12}$$

These constraints clearly capture all possible scenarios of how the input strings can be modified in at most m places. Then let $\#'_t$ be a variable meaning "how many columns of C' are of type t", and let $x_{t,\varphi}$ represent the number of columns of type t in C' whose corresponding character in the solution is set to φ. Finally let $\Delta(t, \varphi)$ be the number of characters of t which are different from φ. As the remaining constraints correspond to our formulation of ILP Resiliency, we have:

$$\sum_{t\in T} z_{t,t'} = \#'_{t'} \qquad \forall t' \in T \tag{13}$$

$$\sum_{\varphi\in\Sigma} x_{t,\varphi} = \#'_t \qquad \forall t \in T \tag{14}$$

$$\sum_{t\in T}\sum_{\varphi\in\Sigma} \Delta(t,\varphi) x_{t,\varphi} \le d \tag{15}$$

This is the standard ILP for Closest String [12], except that $\#'_t$ are now variables, and there exists a solution x exactly when there is a string at distance at most d from the modified strings given by the variables $\#'$. Let $\mathcal{L}$ denote the ILP composed of constraints (11), (12), (13), (14) and (15). Finally, let $\mathcal{Z}$ denote variables $z_{t,t'}$ and $\#'_t$ for every $t, t' \in T$.

Lemma 2. [$\star$] *The instance is satisfiable if and only if $\mathcal{L}$ is $\mathcal{Z}$-resilient.*

It remains to observe that for the above system of constraints $\mathcal{L}$, $\kappa(\mathcal{L})$ is bounded by a function of k (since $|T| = O(2^{k \log_2 k})$. We thus get the following:

Theorem 4. Global Resiliency Closest String *is FPT parameterized by k.*

5 Resilient Scheduling

A fundamental scheduling problem is makespan minimization on unrelated machines, where we have m machines and n jobs, and each job has a vector of processing times with respect to machines $p_j = (p_j^1, \ldots, p_j^m)$, $j \in [n]$. If the vectors p_j and $p_{j'}$ are identical for two jobs j, j', we say these jobs are of the same *type*. Here we consider the case when m and the number of types θ are parameters and the input is given as θ numbers $n_1, \ldots, n_\theta$ of job multiplicities. A *schedule* is an assignment of jobs to machines. For a particular schedule, let n_t^i be the number of jobs of type t assigned to machine i. Then, the *completion time* of machine i is $C^i = \sum_{t\in[\theta]} p_t^i n_t^i$ and the largest C^i is the *makespan* of the schedule, denoted C_{max}.

The parameterization by θ and m might seem very restrictive, but note that when m alone is a parameter, the problem is W[1]-hard even when the machines are identical and the job lengths are given in unary [15]. Also, Asahiro et al. [1] show that it is strongly NP-hard already for *restricted assignment* when there is a number p_j for each job such that for each machine i, $p_j^i \in \{p_j, \infty\}$ and all $p_j \in \{1, 2\}$ and for every job there are exactly two machines where it can run. Mnich and Wiese [20] proved that the problem is FPT with parameters θ and m.

A natural way to introduce resiliency is when we consider unexpected delays due to repairs, fixing software bugs, etc., but we have an upper bound K on the total expected downtime. We assume that the execution of jobs can be resumed after the machine becomes available again, but cannot be moved to another machine, that is, we assume preemption but not migration. Under these assumptions it does not matter when specifically the downtime happens, only the total downtime of each machine. Given m machines, n jobs and $C_{max}, K \in \mathbb{N}$, we say that a scheduling instance has a *K-tolerant makespan C_{max}* if, for every $d_1, \ldots, d_m \in \mathbb{N}$ such that $\sum_{i=1}^{m} d_i \leq K$, there exists a schedule where each machine $i \in [m]$ finishes by the time $C_{max} - d_i$. We obtain the following problem:

RESILIENCY MAKESPAN MINIMIZATION ON UNRELATED MACHINES
Input: m machines, θ job types $p_1, \ldots, p_\theta \in \mathbb{N}^m$, job multiplicities $n_1, \ldots, n_\theta$, and $K, C_{max} \in \mathbb{N}$.
Question: Does this instance have a K-tolerant makespan C_{max} ?

Let x_t^i be a variable expressing how many jobs of type t are scheduled to machine i. We have the following constraints, with the first constraint describing the feasible set of delays, and the subsequent constraints assuring that every job is scheduled on some machine and that every machine finishes by time $C_{max} - d_i$:

$$\sum_{i=1}^{m} d_i \leq K$$

$$\sum_{i=1}^{m} x_i^t = n_t \qquad \forall t \in [\theta]$$

$$\sum_{t=1}^{\theta} x_t^i p_t^i \leq C_{max} - d_i \qquad \forall i \in [m]$$

Theorem 2 and the system of constraints above implies the following result related to the above-mentioned result of Mnich and Wiese [20].

Theorem 5. [$\star$] RESILIENCY MAKESPAN MINIMIZATION ON UNRELATED MACHINES *is FPT when parameterized by θ, m and K and with* $\max_{t \in [\theta], i \in [m]} p_i^t \leq N$ *for some number N given in unary.*

6 Resilient Swap Bribery

The field of computational social choice is concerned with computational problems associated with voting in elections. SWAP BRIBERY, where the goal is to find the cheapest way to bribe voters such that a preferred candidate wins, has received considerable attention. This problem models not only actual bribery, but also processes designed to influence voting (such as campaigning). It is natural to consider the case where an adversarial counterparty first performs their bribery, where we only have an estimate on their budget. The question becomes if, for each such bribery, it is possible, within a given budget, to bribe the election such that our preferred candidate still wins. The number of candidates is a well studied parameter [2, 7]. In this section we will show that the resilient version of

Swap Bribery with unit costs (unit costs are a common setting, cf. Dorn and Schlotter [7]) is FPT using our framework. Let us now give formal definitions.

Elections. An election $E = (C, V)$ consists of a set C of m candidates $c_1, \ldots, c_m$ and a set V of voters (or votes). Each voter i is a linear order $\succ_i$ over the set C. For distinct candidates a and b, we write $a \succ_i b$ if voter i prefers a over b. We denote by $\text{rank}(c, i)$ the position of candidate $c \in C$ in the order $\succ_i$. The preferred candidate is c_1.

Swaps. Let (C, V) be an election and let $\succ_i \in V$ be a voter. A *swap* $\gamma = (a, b)_i$ in preference order $\succ_i$ means to exchange the positions of a and b in $\succ_i$; denote the resulting order by $\succ_i^\gamma$; the *cost* of $(a, b)_i$ is $\pi_i(a, b)$ (in the problem studied in this paper, we have $\pi_i(a, b) = 1$ for every voter i and candidates a, b). A swap $\gamma = (a, b)_i$ is *admissible in* $\succ_i$ if $\text{rank}(a, i) = \text{rank}(b, i) - 1$. A set Γ of swaps is *admissible in* $\succ_i$ if they can be applied sequentially in $\succ_i$, one after the other, in some order, such that each one of them is admissible. Note that the obtained vote, denoted by $\succ_i^\Gamma$, is independent from the order in which the swaps of Γ are applied. We also extend this notation for applying swaps in several votes and denote it V^Γ.

Voting rules. A voting rule $\mathcal{R}$ is a function that maps an election to a subset of candidates, the set of winners. We will show our example for rules which are scoring protocols, but following the framework of so-called "election systems described by linear inequalities" [7] it is easily seen that the result below holds for many other voting rules. With a scoring protocol $s = (s_1, \ldots, s_m) \in \mathbb{N}^m$, a voter i gives s_1 points to his most preferred candidate, s_2 points to his second most preferred candidate and so on. The candidate with most points wins.

> Resiliency Unit Swap Bribery
> Input: An election $E = (C, V)$ with each swap of unit cost and with a scoring protocol $s \in \mathbb{N}^m$, the adversary's budget B_a, our budget B.
> Question: For every adversarial bribery Γ_a of cost at most B_a, is there a bribery Γ of cost at most B such that $E = (C, (V^{\Gamma_a})^\Gamma)$ is won by c_1?

Theorem 6. [$\star$] Resiliency Unit Swap Bribery *with a scoring protocol is FPT when parameterized by the number of candidates m.*

7 Discussion

For some time, Lenstra's theorem was the only approach in parameterized algorithms and complexity based on integer programming. Recently other tools based on integer programming have been introduced: the use of Graver bases for the n-fold integer programming problem [13], ILP approaches in kernelization [14], and an integer quadratic programming analog of Lenstra's theorem [19]. Our approach is a new addition to this powerful arsenal.

References

1. Y. Asahiro, J. Jansson, E. Miyano, H. Ono, and K. Zenmyo. Approximation algorithms for the graph orientation minimizing the maximum weighted outdegree. In *AAIM*, LNCS 4508, pages 167–177, 2007.
2. R. Bredereck, P. Faliszewski, R. Niedermeier, P. Skowron, and N. Talmon. Elections with few candidates: Prices, weights, and covering problems. In *ADT'15*, LNCS 9346, pages 414–431, 2015.
3. R. Chitnis, M. Cygan, M. Hajiaghayi, and D. Marx. Directed subset feedback vertex set is fixed-parameter tractable. *ACM Trans. Alg.*, 11(4):28:1–28:28, 2015.
4. J. Crampton, G. Gutin, and R. Watrigant. A multivariate approach for checking resiliency in access control. In *AAIM 2016*, LNCS 9778, pages 173–184, 2016.
5. J. Crampton, G. Gutin, and R. Watrigant. Resiliency policies in access control revisited. In *Proc. SACMAT'16*, pages 101–111. ACM, 2016.
6. M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.
7. B. Dorn and I. Schlotter. Multivariate complexity analysis of swap bribery. *Algorithmica*, 64(1):126–151, 2012.
8. R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.
9. F. Eisenbrand and G. Shmonin. Parametric integer programming in fixed dimension. *Math. Oper. Res*, 33(4):839–850, 2008.
10. E. Elkind, P. Faliszewski, and A. Slinko. Swap bribery. In *Proc. SAGT 2009*, volume 5814 of *LNCS*, pages 299–310, 2009.
11. András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.
12. J. Gramm, R. Niedermeier, and P. Rossmanith. Fixed-parameter algorithms for CLOSEST STRING and related problems. *Algorithmica*, 37(1):25–42, 2003.
13. Raymond Hemmecke, Shmuel Onn, and Lyubov Romanchuk. n-fold integer programming in cubic time. *Mathematical Programming*, 137(1):325–341, 2013.
14. B. M. P. Jansen and S. Kratsch. A structural approach to kernels for ILPs: Treewidth and total unimodularity. In *ESA*, pages 779–791, 2015.
15. K. Jansen, S. Kratsch, D. Marx, and I. Schlotter. Bin packing with fixed number of bins revisited. *Journal of Computer and System Sciences*, 79(1):39–49, 2013.
16. R. Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, August 1987.
17. H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.
18. N. Li, M. V. Tripunitara, and Q. Wang. Resiliency policies in access control. *ACM Trans. Inf. Syst. Secur.*, 12(4), 2009.
19. D. Lokshtanov. Parameterized integer quadratic programming: Variables and coefficients. *CoRR*, abs/1511.00310, 2015.
20. M. Mnich and A. Wiese. Scheduling and fixed-parameter tractability. *Mathematical Programming*, 154(1):533–562, 2014.
21. P. Pevzner. *Computational Molecular Biology: An Algorithmic Approach*. MIT Press, 2000.
22. A. Sebö. Integer plane multiflows with a fixed number of demands. *J. Comb. Theory, Ser. B*, 59(2):163–171, 1993.

A Omitted Proofs

Lemma 1 $\mathsf{res}(P, s, d, t)$ *is satisfiable if and only if* $\mathcal{L}$ *is* z*-resilient.*

Proof. Let us denote by $\mathcal{L}_z$ the ILP consisting only of inequalities involving variables z, *i.e.* inequalities (7) and (9). Suppose first that $\mathsf{res}(P, s, d, t)$ is satisfiable, and let σ_z be an integral assignment for z such that σ_z satisfies $\mathcal{L}_z$.

We now define a set of users S by picking, in an arbitrary manner, $\sigma_z(z_N)$ users in U_N, for each $N \subseteq P$ (since $\sigma_z(z_N) \leq \min\{s, |U_N|\}$, such a set S must exist). Since S is a set of at most s users, there exists a set of teams $V = \{T_1, \ldots, T_d\}$ such that $\mathcal{U}(V) \cap S = \emptyset$. Then, for each $c \in \mathcal{C}$, let $\sigma_x(x_c)$ be the number of teams of V having configuration c. Clearly we have $\sigma_x(x_c) \in \{0, \ldots, d\}$ and $\sum_{c \in \mathcal{C}} \sigma_x(x_c) = d$, and thus inequalities (6) and (10) are satisfied. Then, for all $N \subseteq P$, we may assume w.l.o.g. that $|T_i \cap U_N| \leq 1$ for all $i \in \{1, \ldots, d\}$. Hence $\sum_{c \in \mathcal{C}_N} \sigma_x(x_c)$ equals $|\mathcal{U}(V) \cap U_N|$, which is the number of users of U_N involved in some teams of V. Since $\mathcal{U}(V) \cap S = \emptyset$, we have $\sum_{c \in \mathcal{C}_N} \sigma_x(x_c) \leq |U_N| - \sigma_z(z_N)$, and thus inequalities (8) are also satisfied for every $N \subseteq P$. Consequently, $\sigma_x \cup \sigma_x$ satisfies $\mathcal{L}$.

Conversely, let $S \subseteq U$, $|S| \leq s$. For each $N \subseteq P$, define $\sigma_z(z_N) = |S \cap U_N|$, which is thus an integral assignment of variables z satisfying $\mathcal{L}_Z$. Hence, there exists a valid assignment σ_x such that $\sigma_z \cup \sigma_x \models \mathcal{L}$. Then, for $c = \{N_1, \ldots, N_b\} \in \mathcal{C}$, $b \leq t$, consider a set of users T consisting of a user chosen arbitrarily in U_{N_i} for each $i \in [b]$. By definition of a configuration, T is a team. Then, since for all $N \subseteq P$, we have, by inequalities (8), that it is possible to construct $\sigma_x(x_c)$ pairwise disjoint such team for each $c \in \mathcal{C}$, each having an empty intersection with S. In other words, for every $S \subseteq U$, $|S| \leq s$, there exists a set of teams V (and V contains at least d teams, thanks to inequality (6)) such that $\mathcal{U}(V) \cap S = \emptyset$, and thus $\mathsf{res}(P, s, d, t)$ is satisfiable. □

Lemma 2 *The instance is satisfiable if and only if* $\mathcal{L}$ *is* $\mathcal{Z}$*-resilient.*

Proof. Constraints involving variables $\mathcal{Z}$ are (11), (12) and (13). Suppose first that the instance is satisfiable, and let $\sigma_{\mathcal{Z}}$ be an integral assignment for $\mathcal{Z}$. We construct C' from C by turning, in an arbitrary way, $\sigma_{\mathcal{Z}}(z_{t,t'})$ columns of type t to columns of type t'. By constraints (11), C' is well-defined, and by constraints (12), the Hamming distance between C and C' is at most m. Then, by constraint (13), matrix C' contains $\sigma_{\mathcal{Z}}(\#'_t)$ columns of type t, for every $t \in T$. Since the instance is satisfiable, there exists a d-closest string s of C'. For $t \in T$ and $\varphi \in \Sigma$, define $\sigma_x(x_{t,\varphi})$ as the number of columns of type t in C' whose corresponding character in s is φ. Since, as we said previously, C' has exactly $\sigma_{\mathcal{Z}}(\#'_t)$ columns of type t, constraint (14) is satisfied for every $t \in T$. Then, since s is a d-closest string for C', constraint (15) is also satisfied.

Conversely, suppose that $\mathcal{L}$ is $\mathcal{Z}$-resilient, and let us consider C', a $k \times L$ normalized matrix of elements of Σ such that the Hamming distance of C and C' is at most m. In polynomial time, we construct $\sigma_{\mathcal{Z}}(z_{t,t'})$ for every $t, t' \in T$ such that (11) and (13) are satisfied. By definition of C', constraint (12) is satisfied. Thus, there exists an integral assignment σ_x satisfying (14) and (15). We now construct s as a string having, for every column type $t \in T$ in C', $\sigma_x(x_{t,\varphi})$ occurence(s) of character φ, for every $\varphi \in \Sigma$ (columns chosen arbitrarily among those of type t in C'). Because of constraint (14), and since $\#'_t$ is the number of columns of type t in C', s is well-defined. Finally, observe that constraint (15) ensures that s is a d-closest string of C', which concludes the proof. ◻

Theorem 5 Resiliency Makespan Minimization on Unrelated Machines *is FPT when parameterized by θ, m and K.*

Proof. We recall the following ILP, denoted by $\mathcal{L}$:

$$\sum_{i=1}^{m} d_i \leq K \tag{16}$$

$$\sum_{i=1}^{m} x_i^t = n_t \qquad \forall t \in [\theta] \tag{17}$$

$$\sum_{t=1}^{\theta} x_t^i p_t^i \leq C_{max} - d_i \qquad \forall i \in [m] \tag{18}$$

We prove that the instance is satisfiable (*i.e.* has a K-tolerant makespan C_{max}) if and only if $\mathcal{L}$ is d-resilient. Suppose first that the instance is satisfiable, and let σ_d be an integral assignment of variables d_i satisfying constraint (16), that is, we have a scenario of delays $\sigma_d(d_1), \ldots, \sigma_d(d_m)$ with total delay at most K. Thus, there exists a schedule where each machine $i \in [m]$ finished by time C_{max} with expected delay $\sigma_d(d_i)$. By defining, for every machine $i \in [m]$ and every type $t \in [\theta]$, $\sigma_x(x_i^t)$ to be the number of jobs of type t assigned to machine i, we obtain an integral assignment for variables x_i^t satisfying constraints (17) and (18). That is, $\mathcal{L}$ is d-resilient.

Conversely, suppose that $\mathcal{L}$ is d-resilient, and let us consider a scenario of delays $d_1, \ldots, d_m$ with total delay at most K, or, equivalently, an integral assignment σ_d of variables d satisfying constraint (16). Since $\mathcal{L}$ is d-resilient, there exists an assignment σ_x of variables x_i^t satisfying (17) and (18). Using the same arguments as above, there exists a schedule where each machine $i \in [m]$ finishes by time $C_{max} - d_i$.

Finally, observe that $\kappa(\mathcal{L})$ is bounded by a function of θ, m and K only. ◻

Theorem 6 Resiliency Unit Swap Bribery *with a scoring protocol is FPT when parameterized by the number of candidates m.*

Proof. A standard way of looking at an election when the number of candidates m is a parameter is as given by *multiplicities* of *voter types*: there are at most $m!$ total orders on C, so we count them and output numbers $n_1, \ldots, n_{m!}$. Observe that for two orders $\succ, \succ'$, the admissible set of swaps Γ such that $\succ' = \succ^{\Gamma}$ is uniquely given as the set of pairs (c_i, c_j) for which either $c_i \succ c_j \wedge c_j \succ' c_i$ or $c_j \succ c_i \wedge c_i \succ' c_j$ (cf. [10, Proposition 3.2]). Thus it is possible to define the price $\pi(i, j)$, for $i, j \in [m!]$, of bribing a voter of type i to become of type j (since every swap is of unit cost, it does not depend on the users). Moreover, we can extend our notation rank(a, i) to denote the position of a in the order of type i.

Similarly to our Global Resiliency Closest String approach, let z_{ij}, for all $i, j \in [m!]$, be a variable representing the number of voters of type i bribed to become of type j, and let y_i, $i \in [m!]$, represent the election $E = (C, V^{\Gamma_a})$ after the first bribery. These constraints describe all possible adversarial briberies:

$$\sum_{j=1}^{m!} z_{ij} = n_i \qquad \forall i \in [m!]$$

$$\sum_{i=1}^{m!} z_{ij} = y_j \qquad \forall j \in [m!]$$

$$\sum_{i=1}^{m!} \sum_{j=1}^{n} \pi(i, j) z_{ij} \le B_a$$

The rest of the ILP is standard; variables x_{ij} will describe the second bribery in the same way as z_{ij} and variables w will describe the election after this bribery, on which we will impose a constraint which is satisfied when c_1 is a winner:

$$\sum_{j=1}^{m!} x_{ij} = y_i \qquad \forall i \in [m!]$$

$$\sum_{i=1}^{m!} x_{ij} = w_j \qquad \forall j \in [m!]$$

$$\sum_{i=1}^{m!} \sum_{j=1}^{m!} \pi(i, j) x_{ij} \le B$$

$$\sum_{k=1}^{m} \sum_{i:\mathrm{rank}(c_1,i)=k} w_i s_k > \sum_{k=1}^{m} \sum_{i:\mathrm{rank}(c_j,i)=k} w_i s_k \qquad \forall j = 2, \ldots, m$$

The rest of the proof is similar to the proof of Lemma 2. □

B Column Resiliency Closest String

Before defining formally the problem, we need to introduce some notation.

In the following, C denotes a normalized $k \times L$ matrix of elements of Σ, *i.e.* an instance of the problem. Let L_a be a non-negative integer, and A be a normalized $k \times L_a$ matrix of elements of Σ. We denote by $C \oplus A$ the $k \times (L + L_a)$ matrix obtained by appending the columns of A to those of C (in other words, the first L columns of $C \oplus A$ are from C, while the L_a last columns are from A). Then, suppose that $L_a \le L$, and let $I = \{\ell_1, \ldots, \ell_{L_a}\} \subseteq [L]$. We will denote by $C \otimes_I A$ the matrix obtained by replacing, for each $i \in \{1, \dots, L_a\}$, the ℓ_i^{th} column of C by the i^{th} column of A. We are now introduce our second resiliency version of Closest String.

Column Resiliency Closest String
Input: A $k \times L$ normalized matrix of elements of Σ, $d \in \mathbb{N}$, $I \subseteq [L]$, $L_a \in \mathbb{N}$.
Question: For every $I' \subseteq I$, for every $k \times |I'|$ normalized matrix M, for every $k \times L_a$ normalized matrix A, does $(C \otimes_{I'} M) \oplus A$ admits a d-closest string ?

Given an instance of Column Resiliency Closest String, we define three sets of variables X, A and M as follows:

$$X = \{x_{t,\varphi} : t \in T, \varphi \in \Sigma\},$$
$$A = \{a_t : t \in T\},$$
$$M = \{m_t^r : t \in T\} \cup \{m_t^a : t \in T\}.$$

Note that the following ILP is an extension of the one used to solve Closest String in FPT time parameterized by k [12]. Given $t \in T$ and $\varphi \in \Sigma$, variable $x_{t,\varphi}$ represents the number of columns of type t (in C) whose corresponding character in the solution is set to φ. The idea of our extension is to model the resiliency part by variables in A and M, i.e. these variables form z. More precisely, for a type $t \in T$, a_t will represent the number of columns of type t in A, while m_t^r and m_t^a will denote respectively the number of columns of type t in C that will be replaced, and the number of columns of type t in M that will be added instead, from the operation $(C \otimes_{I'} M)$. Given $t \in T$, we denote by $\#_t$ and $\#_t^I$ the number of columns of type t in C, and the number of columns of type t in C among those of I, respectively. The key observation is that $\#_t - m_t^r + m_t^a + a_t$ is the number of columns of type t in $(C \otimes_{I'} M) \oplus A$. Finally, for $t \in T$ and $i \in [k]$, $\varphi_{t,i}$ denotes the alphabet symbol at the i^{th} entry of column type t. Let $\mathcal{L}$ be the following ILP:

$$\sum_{\varphi \in \Sigma} x_{t,\varphi} = \#_t - m_t^r + m_t^a + a_t \quad \text{for all} \quad t \in T \tag{19}$$

$$\sum_{t \in T} \sum_{\varphi \in \Sigma \setminus \{\varphi_{t,i}\}} x_{t,\varphi} \leq d \quad \text{for all} \quad i \in [k] \tag{20}$$

$$\sum_{t \in T} m_t^r - m_t^a = 0 \quad \text{for all} \quad t \in T \tag{21}$$

$$0 \leq m_t^r \leq \#_t^I \quad \text{for all} \quad t \in T \tag{22}$$

$$\sum_{t \in T} a_t \leq L_a \tag{23}$$

Constraint (19) requires that a solution string can indeed be constructed from an assignment of x, and constraint (20) ensures the solution will be a d-closest string. Constraints (21) and (22) ensure that we will remove from C columns from I only, while we replace them by the same number of columns (those from M). Finally, constraint (23) requires the addition of at most L_a columns (those from matrix A).

In the following, we will say that the instance (of Closest String) is *satisfiable* if and only if for every $I' \subseteq I$, for every $k \times |I'|$ normalized matrix M, for every $k \times L_a$ normalized matrix A, the instance whose matrix is $(C \otimes_{I'} M) \oplus A$ admits a d-closest string.

Lemma 3. *The instance is satisfiable if and only if $\mathcal{L}$ is $(A \cup M)$-resilient.*

Proof. Observe that the constraints of $\mathcal{L}$ involving only variables from $A \cup M$ are (21), (22) and (23). Let us denote by $\mathcal{L}_{AM}$ the restriction of $\mathcal{L}$ to these constraints.

Suppose first that the instance is satisfiable, and let σ_{AM} be an integral assignment for $A \cup M$ such that $\sigma_{AM} \models \mathcal{L}_{AM}$. We construct A as a matrix having exactly $\sigma_{AM}(a_t)$ columns of type t, for every $t \in T$. Because of constraint (23), A has L_a columns (and k rows). On the other hand, construct M as a matrix having exactly $\sigma_{AM}(m_t^a)$ columns of type t, for every $t \in T$. Then, constraint (22) ensures that for every $t \in T$, there exists at least $\sigma_{AM}(m_t^r)$ columns of type t among those of I. Hence, it is possible to construct $I' \subseteq I$ as the union, for every $t \in T$, of $\sigma_{AM}(m_t^r)$ column indices of I having type t, chosen arbitrarily. Let $C' = (C \otimes_{I'} M) \oplus A$. Observe that by constraint (21), matrix M has exactly $|I'|$ columns. Moreover, observe that in C', the number of columns of type t is exactly $\#_t - \sigma_{AM}(m_t^r) + \sigma_{AM}(m_t^a) + \sigma_{AM}(a_t)$, for every $t \in T$. Since the instance is satisfiable, C' admits a d closest string r. For $t \in T$ and $\varphi \in \Sigma$, define $\sigma_X(x_{t,\varphi})$ as the number of columns of type t in C' whose corresponding character in r is φ. Since, as said previously, C' has exactly $\#_t - \sigma_{AM}(m_t^r) + \sigma_{AM}(m_t^a) + \sigma_{AM}(a_t)$ columns of type t, constraint (19) is satisfied for every $t \in T$. Then, since r is a d-closest string, constraint (20) is also satisfied for every $i \in [k]$. We thus have $(\sigma_{AM} \cup \sigma_X) \models \mathcal{L}$.

Conversely, suppose that $\mathcal{L}$ is resilient, and let us consider $I' \subseteq I$, a normalized $k \times |I'|$ matrix M, and a normalized $k \times L_a$ matrix A. Let $C' = (C \otimes_{I'} M) \oplus A$. We define, for every $t \in T$, $\sigma_{AM}(a_t)$ as the number of columns of A of type t, $\sigma_{AM}(m_t^a)$ as the number of columns of M of type t, and finally $\sigma_{AM}(m_t^r)$ as the number of columns of C of type t among those of I'. Using similar arguments as previously, we can argue that $\sigma_{AM} \models \mathcal{L}_{AM}$. Thus, there exists an integral assignment σ_X of X such that $(\sigma_{AM} \cup \sigma_X) \models \mathcal{L}$. We now construct r as a string having, for every column type $t \in T$ in C', $\sigma_X(x_{t,\varphi})$ occurrence(s) of character φ, for every $\varphi \in \Sigma$ (columns chosen arbitrarily among those of type t in C'). Because of constraint (19), and since $\#_t - \sigma_{AM}(m_t^r) + \sigma_{AM}(m_t^a) + \sigma_{AM}(a_t)$ is the number of columns of type t in C', r is well defined. Finally, observe that constraint (20) ensures that r is a d-closest string of C', which concludes the proof. □

Now, observe that $\kappa(\mathcal{L})$ is bounded by a function of k only (since $|T| = O(2^{k \log_2(k)})$). Hence, combining the previous lemma with Theorem 2, we obtain the following result.

Theorem 7. Column Resiliency Closest String *is FPT parameterized by* k.