# Parameterized Algorithms for Matrix Completion with Radius Constraints

## Tomohiro Koana 🆔
Technische Universität Berlin, Faculty IV, Algorithmics and Computational Complexity, Germany
tomohiro.koana@tu-berlin.de

## Vincent Froese
Technische Universität Berlin, Faculty IV, Algorithmics and Computational Complexity, Germany
vincent.froese@tu-berlin.de

## Rolf Niedermeier 🆔
Technische Universität Berlin, Faculty IV, Algorithmics and Computational Complexity, Germany
rolf.niedermeier@tu-berlin.de

## Abstract

Considering matrices with missing entries, we study NP-hard matrix completion problems where the resulting completed matrix should have limited (local) radius. In the pure radius version, this means that the goal is to fill in the entries such that there exists a "center string" which has Hamming distance to all matrix rows as small as possible. In stringology, this problem is also known as CLOSEST STRING WITH WILDCARDS. In the local radius version, the requested center string must be one of the rows of the completed matrix.

Hermelin and Rozenberg [CPM 2014, TCS 2016] performed a parameterized complexity analysis for CLOSEST STRING WITH WILDCARDS. We answer one of their open questions, fix a bug concerning a fixed-parameter tractability result in their work, and improve some running time upper bounds. For the local radius case, we reveal a computational complexity dichotomy. In general, our results indicate that, although being NP-hard as well, this variant often allows for faster (fixed-parameter) algorithms.

## 1 Introduction

In many applications data can only be partially measured, which leads to incomplete data records with missing entries. A common problem in data mining, machine learning, and computational biology is to infer these missing entries. In this context, matrix completion problems play a central role. Here the goal is to fill in the unknown entries of an incomplete data matrix such that certain measures regarding the completed matrix are optimized. Ganian et al. [9] recently studied parameterized algorithms for two variants of matrix completion problems; their goal was to minimize either the rank or the number of distinct rows in the completed matrix. In this work, we focus our study on another variant, namely, minimizing the "radius" of the completed matrix. Indeed, this is closely related to the topic of consensus (string) problems, which received a lot of attention in stringology and particularly with respect to parameterized complexity studies [16, 15, 3]. Indeed, radius minimization for

| | | | | |
|---|---|---|---|---|
| 1 | 1 | * | 0 | 1 |
| * | 0 | 2 | * | 2 |
| 1 | * | 1 | * | 1 |
| 0 | 2 | 2 | 0 | 0 |

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 2 | 0 | 1 |
| 1 | 0 | 2 | 0 | 2 |
| 1 | 2 | 1 | 0 | 1 |
| 0 | 2 | 2 | 0 | 0 |

| | | | | |
|---|---|---|---|---|
| 0 | 2 | 2 | 0 | 1 |

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 2 | 0 | 1 |
| 1 | 0 | 2 | 0 | 2 |
| 1 | 0 | 1 | 0 | 1 |
| 0 | 2 | 2 | 0 | 0 |

| | | | | |
|---|---|---|---|---|
| 1 | 0 | 2 | 0 | 2 |
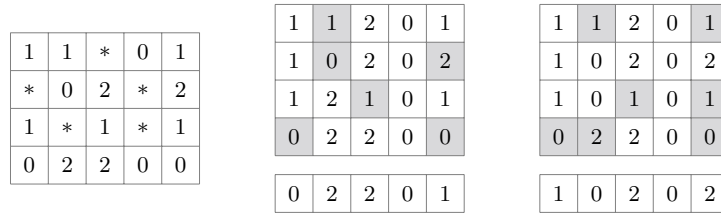
**Figure 1** An example of MINRMC and MINLRMC. The incomplete input matrix is depicted in the left (a $*$ denotes a missing entry). Optimal solutions for MINRMC ($d = 2$) and MINLRMC ($d = 3$) are shown in the middle and right. Entries differing from the solution vector are marked gray.

incomplete matrices is known in the stringology community under the name CLOSEST STRING WITH WILDCARDS [12], a generalization of the frequently studied CLOSEST STRING problem. Herein, an incomplete matrix shall be completed such that there exists a vector that is not too far from each row vector of the completed matrix in terms of the Hamming distance (that is, the completed matrix has small *radius*). We consider this radius minimization problem and also a local variant where all row vectors of the completed matrix must be close to a row vector in the matrix.

Given the close relation to CLOSEST STRING, which is NP-hard already for binary strings [8], all problems we study in this work are NP-hard in general. However, we provide several positive algorithmic results, namely fixed-parameter tractability with respect to natural parameters or even polynomial-time solvability for special cases. Formally, we study the following problems (see Figure 1 for illustrative examples):

MINIMUM RADIUS MATRIX COMPLETION (MINRMC)

**Input:**      An incomplete matrix $\mathbf{S} \in (\Sigma \cup \{*\})^{n \times \ell}$ and $d \in \mathbb{N}$.
**Question:** Is there a "completion" $\mathbf{T} \in \Sigma^{n \times \ell}$ of $\mathbf{S}$ such that $\delta(v, \mathbf{T}) \leq d$ for some vector $v \in \Sigma^\ell$?

MINIMUM LOCAL RADIUS MATRIX COMPLETION (MINLRMC)

**Input:**      An incomplete matrix $\mathbf{S} \in (\Sigma \cup \{*\})^{n \times \ell}$ and $d \in \mathbb{N}$.
**Question:** Is there a "completion" $\mathbf{T} \in \Sigma^{n \times \ell}$ of $\mathbf{S}$ such that $\delta(v, \mathbf{T}) \leq d$ for some vector $v \in \mathbf{T}$?

Here, missing entries are denoted by $*$. A completion of an incomplete matrix $\mathbf{S} \in \Sigma \cup \{*\}$ is a matrix obtained by replacing each missing entry with some symbol from $\Sigma$. We denote the Hamming distance by $\delta$. The Hamming distance between a vector $v$ and a matrix $\mathbf{T}$ is the maximum Hamming distance between $v$ and a vector in $\mathbf{T}$. In fact, our results for MINRMC also hold for the following more general problem:

CONSTRAINT RADIUS MATRIX COMPLETION (CONRMC)

**Input:**      An incomplete matrix $\mathbf{S} \in (\Sigma \cup \{*\})^{n \times \ell}$ and $d_1, \ldots, d_n \in \mathbb{N}$.
**Question:** Is there a "completion" $\mathbf{T} \in \Sigma^{n \times \ell}$ of $\mathbf{S}$ such that for some vector $v \in \Sigma^\ell$, $\delta(v, \mathbf{T}[i]) \leq d_i$ for each $i \in [n]$?

### Related work

Our most important reference point is the work of Hermelin and Rozenberg [12] who analyzed the parameterized complexity of MINRMC (under the name CLOSEST STRING WITH WILDCARDS) with respect to several problem-specific parameters (also see Table 1). In

■ **Table 1** Overview of known results and our new results for MINRMC and MINLRMC. Notation: $n$ – number of rows, $\ell$ – number of columns, $|\Sigma|$ – alphabet size, $d$ – distance bound, $k$ – maximum number of missing entries in any row vector. Note that all our results for MINRMC also hold for CONRMC.

| Parameter | MINRMC | Reference | MINLRMC | Reference |
|---|---|---|---|---|
| $n$ | $O^*(2^{2^{O(n \log n)}})$ | [12] | $O^*(2^{O(n^2 \log n)})$ | Corollary 3 |
| | $O^*(2^{O(n^2 \log n)})$ | [13] | | |
| $\ell$ | $O^*(2^{\ell^2/2})$ | [12] | $O^*(\ell^\ell)$ | Corollary 7 |
| | $O^*(\ell^\ell)$ | Corollary 6 | | |
| $d = 1$ | $O(n\ell^2)$ for $|\Sigma| = 2$ | [12] | $O(n^2\ell)$ | Corollary 2 |
| | $O(n\ell)$ | Theorem 1 | | |
| $d = 2$ | NP-hard for $|\Sigma| = 2$ | [12] | NP-hard for $|\Sigma| = 2$ | [12] |
| $k$ | NP-hard for $k = 0$ | [8] | $O^*(k^k)$ | Corollary 7 |
| $d + k$ | $O^*((d+1)^{d+k})$ | Theorem 9 | | |
| $d + k + |\Sigma|$ | $O^*(|\Sigma|^k \cdot d^d)$ | [12] | $O^*(|\Sigma|^k)$ | trivial |
| | $O^*(2^{4d+k} \cdot |\Sigma|^{d+k})$ | Theorem 12 | | |

particular, they provided fixed-parameter tractability results for the parameters number $n$ of rows, number $\ell$ of columns, and radius $d$ combined with maximum number $k$ of missing entries per row. However, we will show that their fixed-parameter tractability result for the combined parameter $d + k$ is flawed. Moreover, they showed a computational complexity dichotomy for binary inputs between radius 1 (polynomial time) and radius 2 (NP-hard) (see Table 1 for a complete overview).

As mentioned before, Ganian et al. [9] started research on the parameterized complexity of two related matrix completion problems (minimizing rank and minimizing number of distinct rows). Very recently, Eiben et al. [7] studied generalizations of our problems which demand that the completed matrix be clustered into several submatrices of small (local) radius – basically, our work studies the case of a single cluster. They proved fixed-parameter tractability (problem kernels of superexponential size) with respect to the combined parameter $(c, d, r)$. Here, $c$ is the number of clusters and $r$ is the minimum number of rows and columns covering all missing entries. On the negative side, they proved that dropping any of $c$, $d$, or $r$ results in parameterized intractability even for binary alphabet. Amir et al. [1] showed that the clustering version of MINLRMC on complete matrices with unbounded alphabet size is NP-hard when restricted to only two columns. Note that fixed-parameter tractability for the clustering variant implies fixed-parameter tractability for MINRMC and MINLRMC with respect to $d + r$. Indeed, to reach for (more) practical algorithms, we consider an alternative parameterization by the maximum number $k$ of missing entries in any row vector (which can be much smaller than $r$). Finally, let us also mention our companion work [14], in which we studied a matrix completion problem with diametrical constraints – all pairwise row distances must be within a specified range in the completion.

## Our contributions

We survey our and previous results in Table 1. Notably, all of our results for MINRMC indeed also hold for the more general CONRMC when setting $d := \max\{d_1, d_2, \ldots, d_n\}$.

Let us highlight a few of our new results in comparison with previous work. For MINRMC, we give a linear-time algorithm for the case radius $d = 1$ and arbitrary alphabet. This answers an open question of Hermelin and Rozenberg [12]. We also show fixed-parameter tractability

with respect to the combined parameter $d + k$, which was already claimed by Hermelin and Rozenberg [12] but was flawed, as we will point out by providing a counter-example to their algorithm. Lastly, inspired by known results for CLOSEST STRING, we give a more efficient algorithm for small alphabet size.

Regarding MINLRMC, we show that it can be solved in polynomial time when $d = 1$. This yields a computational complexity dichotomy since MINLRMC is NP-hard for $d = 2$. Moreover, we show that MINLRMC is fixed-parameter tractable with respect to the maximum number $k$ of missing entries per row. Remarkably, this stands in sharp contrast to MINRMC, which is NP-hard even for $k = 0$.

## 2 Preliminaries

For $m \leq n \in \mathbb{N}$, let $[m, n] := \{m, \ldots, n\}$ and $[n] := [1, n]$.

Let $\mathbf{T} \in \Sigma^{n \times \ell}$ be an $(n \times \ell)$-matrix over a finite alphabet $\Sigma$. Let $i \in [n]$ and $j \in [\ell]$. We use $\mathbf{T}[i, j]$ to denote the character in the $i$-th row and $j$-th column of $\mathbf{T}$. We use $\mathbf{T}[i, :]$ (or $\mathbf{T}[i]$ in short) to denote the *row vector* $(\mathbf{T}[i, 1], \ldots, \mathbf{T}[i, \ell])$ and $\mathbf{T}[:, j]$ to denote the *column vector* $(\mathbf{T}[1, j], \ldots, \mathbf{T}[n, j])^T$. For any subsets $I \subseteq [n]$ and $J \subseteq [\ell]$, we write $\mathbf{T}[I, J]$ to denote the submatrix obtained by omitting rows in $[n] \setminus I$ and columns in $[\ell] \setminus J$ from $\mathbf{T}$. We abbreviate $\mathbf{T}[I, [\ell]]$ and $\mathbf{T}[[n], J]$ as $\mathbf{T}[I, :]$ (or $\mathbf{T}[I]$ for short) and $\mathbf{T}[:, J]$, respectively. We use the special character $*$ for a *missing* entry. A matrix $\mathbf{S} \in (\Sigma \cup \{*\})^{n \times \ell}$ that contains a missing entry is called *incomplete*. We say that $\mathbf{T} \in \Sigma^{n \times \ell}$ is a *completion* of $\mathbf{S} \in (\Sigma \cup \{*\})^{n \times \ell}$ if either $\mathbf{S}[i, j] = *$ or $\mathbf{S}[i, j] = \mathbf{T}[i, j]$ holds for all $i \in [n]$ and $j \in [\ell]$.

Let $v, v' \in (\Sigma \cup \{*\})^\ell$ be row vectors and let $\sigma \in \Sigma \cup \{*\}$. We write $P_\sigma(v)$ to denote the set $\{j \in [\ell] \mid v[j] = \sigma\}$ of column indices where the corresponding entries of $v$ are $\sigma$. We write $Q(v, v')$ to denote the set $\{j \in [\ell] \mid v[j] \neq v'[j] \wedge v[j] \neq * \wedge v'[j] \neq *\}$ of column indices where $v$ and $v'$ disagree (not considering positions with missing entries). The *Hamming distance* between $v$ and $v'$ is $\delta(v, v') := |Q(v, v')|$. For $\mathbf{S} \in (\Sigma \cup \{*\})^{n \times \ell}$ and $v \in (\Sigma \cup \{*\})^\ell$, let $\delta(v, \mathbf{S}) := \max_{i \in [n]} \delta(v, \mathbf{S}[i])$. The binary operation $v \oplus v'$ replaces the missing entries of $v$ with the character in $v'$ in the corresponding position, given that $v'$ contains no missing entry. We sometimes use string notation $\sigma_1 \sigma_2 \sigma_3$ to represent the row vector $(\sigma_1, \sigma_2, \sigma_3)$.

**Parameterized Complexity**

We sometimes use the $O^*$-notation which suppresses polynomial factors in the input size. A *parameterized problem* $\Pi$ is a set of instances $(I, k) \in \Sigma^* \times \mathbb{N}$, where $k$ is called the *parameter* of the instance. A parameterized problem is *fixed-parameter tractable* if $(I, k) \in \Pi$ can be determined in $f(k) \cdot |I|^{O(1)}$ time for an arbitrary computable function $f$. An algorithm with such a running time is called a *fixed-parameter algorithm*.

## 3 Linear-time algorithm for radius $d = 1$

Hermelin and Rozenberg [12, Theorem 6] gave a reduction from MINRMC to 2-SAT for the case $|\Sigma| = 2$ and $d = 1$, resulting in an $O(n\ell^2)$-time algorithm. We provide a more efficient reduction to 2-SAT, exploiting the compact encoding $C_{\leq 1}$ of the "at-most-one" constraint by Sinz [20]. Let $L = \{l_1, \ldots, l_m\}$ be a set of $m$ literals. The encoding uses $m - 1$ additional

variables $r_1, \ldots, r_{m-1}$ and it is defined as follows:

$$C_{\leq 1}(L) = (\neg l_1 \vee r_1) \wedge (\neg l_m \vee \neg r_{m-1})$$
$$\wedge \bigwedge_{2 \leq j \leq m-1} ((\neg l_j \vee \neg r_{j-1}) \wedge (\neg l_j \vee r_j) \wedge (\neg r_{j-1} \vee r_j)).$$

Note that if $l_j$ is true for some $j \in [m]$, then $r_j, \ldots, r_{m-1}$ are all true and $r_1, \ldots, r_{j-1}$ are all false. Hence, at most one literal in $L$ can be true.

Actually, our algorithm solves CONRMC, the generalization of MINRMC where the distance bound can be specified for each row vector individually.

▶ **Theorem 1.** *If* $\max_{i \in [n]} d_i = 1$, *then* CONRMC *can be solved in* $O(n\ell)$ *time.*

**Proof.** We reduce CONRMC to 2-SAT. Let $I_d := \{i \in [n] \mid d_i = d\}$ be the row indices for which the distance bound is $d$ for $d \in \{0, 1\}$. We define a variable $x_{j,\sigma}$ for each $j \in [\ell]$ and $\sigma \in \Sigma$. The intuition behind our reduction is that the $j$-th entry of the solution vector $v$ becomes $\sigma$ when $x_{j,\sigma}$ is true. We give the construction of a 2-CNF formula $\phi$ in three parts $\phi_1, \phi_2, \phi_3$ (that is, $\phi = \phi_1 \wedge \phi_2 \wedge \phi_3$).

▬ Let $X_j = \{x_{j,\sigma} \mid \sigma \in \Sigma\}$ for each $j \in [\ell]$. The first subformula will ensure that at most one character is assigned to each entry of the solution vector $v$:

$$\phi_1 = \bigwedge_{j \in [\ell]} C_{\leq 1}(X_j).$$

▬ Subformula $\phi_2$ handles distance-0 constraints:

$$\phi_2 = \bigwedge_{i \in I_0} \bigwedge_{\substack{j \in [\ell] \\ \mathbf{S}[i,j] \neq *}} (x_{j,\mathbf{S}[i,j]}).$$

▬ Finally, subformula $\phi_3$ guarantees that the solution vector $v$ deviates from each row vector of $\mathbf{S}[I_1]$ in at most one position.

$$\phi_3 = \bigwedge_{i \in I_1} C_{\leq 1}(\{\neg x_{j,\mathbf{S}[i,j]} \mid j \in [\ell], \mathbf{S}[i,j] \neq *\}).$$

Note that our construction uses $O(|\Sigma| \cdot \ell)$ variables and $O((n + |\Sigma|) \cdot \ell) = O(n\ell)$ clauses. We prove the correctness of the reduction.

($\Rightarrow$) Suppose that there exists a vector $v \in \Sigma^\ell$ such that $\delta(v, \mathbf{S}[i]) \leq d_i$ holds for each $i \in [n]$. For each $j \in [\ell]$ and $\sigma \in \Sigma$, we set $x_{j,\sigma}$ to true if $v[j] = \sigma$, and false otherwise. It is easy to see that this truth assignment satisfies $\phi$.

($\Leftarrow$) Suppose that there exists a satisfying truth assignment $\varphi$. Let $J^*$ denote the column indices $j \in [\ell]$ such that $\varphi(x_{j,\sigma}) = 0$ for all $\sigma \in \Sigma$. Note that at most one variable in $X_j$ is set to true in $\varphi$ for each $j \in [\ell]$. It follows that, for each $j \in [\ell] \setminus J^*$, there exists exactly one character $\sigma_j \in \Sigma$ satisfying $\varphi(x_{j,\sigma_j}) = 1$ and we assign $v[j] = \sigma_j$. For each $j \in J^*$, we set $v[j] = \sigma^*$ for some arbitrary character $\sigma^* \in \Sigma$. The formula $\phi_2$ ensures that $\delta(v, \mathbf{S}[i]) = 0$ holds for each $i \in I_0$. Moreover, $\phi_3$ ensures that there is at most one column index $j \in [\ell]$ such that $\mathbf{S}[i,j] \neq *$ and $\mathbf{S}[i,j] \neq v[j]$ for each $i \in I_1$. ◀

Note that MINRMC (and thus CONRMC) is NP-hard for $|\Sigma| = 2$ and $d = 2$ [12]. Thus, our result implies a complete complexity dichotomy regarding $d$. We remark that this dichotomy also holds for MINLRMC since there is a simple reduction from MINLRMC to CONRMC. To solve an instance $(\mathbf{S}, d)$ of MINLRMC, we solve $n$ instances of ConRMC: For

■ **Algorithm 1** Improved algorithm for CONRMC (based on Hermelin and Rozenberg [12]).

---

**Input:** An incomplete matrix $\mathbf{S} \in (\Sigma \cup \{*\})^{n \times \ell}$ and $d_1, \ldots, d_n \in \mathbb{N}$.
**Task:** Decide whether there exists a row vector $v \in \Sigma^\ell$ with $d(v, \mathbf{S}[i]) \le d_i$ for all $i \in [n]$.
 1: **if** $d_i < 0$ for some $i \in [n]$ **then return No**.
 2: **if** $\ell - |P_*(\mathbf{S}[i])| \le d_i$ for all $i \in [n]$ **then return Yes**.
                                       $\triangleright$ $|P_*(\mathbf{S}[i])|$ is the number of missing entries in $\mathbf{S}[i]$
 3: Choose any $i \in [n]$ such that $\ell - |P_*(\mathbf{S}[i])| > d_i$.
 4: Choose any $R \subseteq [\ell] \setminus P_*(\mathbf{S}[i])$ with $|R| = d_i + 1$.
 5: **for all** $j \in R$ **do**
 6:      Let $\mathbf{S}' = \mathbf{S}[:, [\ell] \setminus \{j\}]$ and $d'_{i'} = d_i - \delta(\mathbf{S}[i, j], \mathbf{S}[i', j])$ for each $i' \in [n]$.
 7:      **if** recursion on $(\mathbf{S}', d'_1, \ldots, d'_n)$ returns **Yes then return Yes**.
 8: **return No**.

---

each $i \in [n]$, we solve the instance $(\mathbf{S}, d_1, \ldots, d_n)$ where $d_{i'} = d$ for each $i' \in [n] \setminus \{i\}$ and $d_i = 0$. Clearly, $(\mathbf{S}, d)$ is a **Yes**-instance if and only if at least one CONRMC-instance is a **Yes**-instance. This yields the following.

▶ **Corollary 2.** *MINLRMC can be solved in $O(n^2 \ell)$ time when $d = 1$.*

Since CONRMC is solvable in $O^*(2^{O(n^2 \log n)})$ time [13], we also obtain the following result, where the running time bound only depends on the number $n$ of rows.

▶ **Corollary 3.** *MINLRMC can be solved in $O^*(2^{O(n^2 \log n)})$ time.*

Finally, we remark that CONRMC can be solved in linear time for binary alphabet $\Sigma = \{0, 1\}$ if $d_i \ge \ell - 1$ for all $i \in [n]$ (the problem remains NP-hard in the case of unbounded alphabet size [17] even if $d_i \ge \ell - 1$ for all $i \in [n]$): First, we remove each row vector with distance bound $\ell$. We also remove every row vector with at least one missing entry since it has distance at most $\ell - 1$ from any vector of length $\ell$. We then remove every duplicate row vector. This can be achieved in linear time: We sort the row vectors lexicographically using radix sort and we compare each row vector to the adjacent row vectors in the sorted order. We return **Yes** if and only if there are at most $2^\ell - 1$ row vectors, because each distinct row vector $u \in \{0, 1\}^\ell$ excludes exactly one row vector $\overline{u} \in \{0, 1\}^\ell$ where $\overline{u}[j] = 1 - u[j]$ for each $j \in [\ell]$. Summarizing, we arrive at the following.

▶ **Proposition 4.** *If $\Sigma = \{0, 1\}$ and $d_i \ge \ell - 1$ for all $i \in [n]$, then CONRMC can be solved in linear time.*

## 4   Parameter number $\ell$ of columns

Hermelin and Rozenberg [12, Theorem 3] showed that one can solve MINRMC in $O(2^{\ell^2/2} \cdot n\ell)$ time using a search tree algorithm. We use a more refined recursive step to obtain a better running time (see Algorithm 1). In particular we employ a trick used by Gramm et al. [11] in order to reduce the search space to $d + 1$ subcases. Note that for nontrivial instances clearly $d < \ell$.

▶ **Theorem 5.** *For $d := \max_{i \in [n]} d_i$, CONRMC can be solved in $O((d + 1)^\ell \cdot n\ell)$ time.*

**Proof.** We prove that Algorithm 1 is correct by induction on $\ell$. More specifically, we show that it returns **Yes** if there exists a vector $v \in \Sigma^\ell$ that satisfies $\delta(\mathbf{S}[i], v) \le d_i$ for all $i \in [n]$. It is easy to see that the algorithm is correct for the base case $\ell = 0$, because it returns

**Yes** if $d_i$ is nonnegative for all $i \in [n]$ and **No** otherwise (Lines 1 and 2). Consider the case $\ell > 0$. The terminating conditions in Lines 1 and 2 are clearly correct. We show that branching on $R$ is correct in Lines 4 and 5. If $v[j] \neq \mathbf{S}[i,j]$ holds for all $j \in R$, then we have a contradiction $\delta(v, \mathbf{S}[i]) \geq |R| > d_i$. Thus the branching on $R$ leads to a correct output. Now the induction hypothesis ensures that the recursion on $\mathbf{S}[:, [\ell] \setminus \{j\}]$ (notice that it has exactly one column less) returns a desired output. This implies that our algorithm is correct.

As regards the time complexity, note that each node in the search tree has at most $d+1$ children. Moreover, the depth of the search tree is at most $\ell$ because the number of columns decreases for each recursion. Since each recursion step only requires linear (that is, $O(n\ell)$) time, the overall running time is in $O((d+1)^\ell \cdot n\ell)$. ◄

Since $d < \ell$ for nontrivial input instances, Theorem 5 yields "linear-time fixed-parameter tractability" with respect to $\ell$, meaning an exponential speedup over the previous result due to Hermelin and Rozenberg [12].

▶ **Corollary 6.** *CONRMC can be solved in $O(n\ell^{\ell+1})$ time.*

**Proof.** For each $i \in [n]$, we construct an CONRMC-instance, where the input matrix is $\mathbf{S}_i = \mathbf{S}[:, P_*(\mathbf{S}[i])]$ and $d_{i,i'} = d - \delta(\mathbf{S}[i], \mathbf{S}[i'])$ for each $i' \in [n]$. We return **Yes** if and only if there is a **Yes**-instance $(\mathbf{S}_i, d_{i,1}, \ldots, d_{i,n})$ of CONRMC. Each CONRMC-instance requires $O(n\ell)$ time to construct, and $O(nk^{k+1})$ time to solve, because $\mathbf{S}_i$ contains at most $k$ columns. ◄

We remark that this algorithm cannot be significantly improved assuming the ETH.[1] It is known that there is no $\ell^{o(\ell)} \cdot n^{O(1)}$-time algorithm for the special case CLOSEST STRING unless the ETH fails [17]. The running time of our algorithm matches this lower bound (up to a constant in the exponent) and therefore there is presumably no substantially faster algorithm with respect to $\ell$.

As a consequence of Corollary 6, we obtain a fixed-parameter algorithm for MINLRMC with respect to the maximum number $k$ of missing entries per row in the input matrix.

▶ **Corollary 7.** *MINLRMC can be solved in time $O(n^2\ell + n^2k^{k+1})$.*

## 5  Combined parameter $d + k$

In this section we generalize two algorithms (one by Gramm et al. [11] and one by Ma and Sun [18]) for the special case of MINRMC in which the input matrix is complete (known as the CLOSEST STRING problem) to the case of incomplete matrices. We will describe both algorithms briefly. In fact, both algorithms solve the special case of CONRMC, referred to as NEIGHBORING STRING (generalizing CLOSEST STRING by allowing row-individual distances), where the input matrix is complete.

NEIGHBORING STRING
**Input:**   A matrix $\mathbf{T} \in \Sigma^{n \times \ell}$ and $d_1, \ldots, d_n \in \mathbb{N}$
**Question:** Is there a row vector $v \in \Sigma^\ell$ such that $\delta(v, \mathbf{T}[i]) \leq d_i$ for each $i \in [n]$?

The algorithm of Gramm et al. [11] is given in Algorithm 2. First, it determines whether the first row vector $\mathbf{T}[1]$ is a solution. If not, then it finds another row vector $\mathbf{T}[i]$ that differs from $\mathbf{T}[1]$ on more than $d_i$ positions and branches on the column positions $Q(\mathbf{T}[1], \mathbf{T}[i])$ where $\mathbf{T}[1]$ and $\mathbf{T}[i]$ disagree.

---

[1] The Exponential Time Hypothesis asserts that 3-SAT cannot be solved in $O^*(2^{o(n+m)})$ time for a 3-CNF formula with $n$ variables and $m$ clauses.

◼ **Algorithm 2** Algorithm for NEIGHBORING STRING by Gramm et al. [11].

---

**Input:** A matrix $\mathbf{T} \in (\Sigma \cup \{*\})^{n \times \ell}$ and $d_1, \ldots, d_n \in \mathbb{N}$.
**Task:** Decide whether there exists a row vector $v \in \Sigma^\ell$ with $d(v, \mathbf{T}[i]) \leq d_i$ for all $i \in [n]$.
1: **if** $d_i < 0$ for some $i \in [n]$ **then return No**.
2: **if** $\delta(\mathbf{T}[1], \mathbf{T}[i]) \leq d_i$ for all $i \in [n]$ **then return Yes**.
3: Choose any $i \in [n]$ such that $\delta(\mathbf{T}[1], \mathbf{T}[i]) > d_i$.
4: Choose any $Q' \subseteq Q(\mathbf{T}[1], \mathbf{T}[i])$ with $|Q'| = d_i + 1$.
5: **for all** $j \in Q'$ **do**
6:     Let $\mathbf{T}' = \mathbf{T}[:, [\ell] \setminus \{j\}]$ and $d'_{i'} = d_i - \delta(\mathbf{T}[i, j], \mathbf{T}[i', j])$ for each $i' \in [n]$.
7:     **if** recursion on $(\mathbf{T}', d'_1, \ldots, d'_n)$ returns **Yes then return Yes**.
8: **return No**.

---

Using a search variant of Algorithm 2, Hermelin and Rozenberg [12, Theorem 4] claimed that MINRMC is fixed-parameter tractable with respect to $d + k$. Here, we reveal that their algorithm is in fact not correct. The algorithm chooses an arbitrary row vector $\mathbf{S}[i]$ and calls the algorithm by Gramm et al. [11] with input matrix $\mathbf{S}' = \mathbf{S}[:, [\ell] \setminus P_*(\mathbf{S}[i])]$. This results in a set of row vectors $v$ satisfying $\delta(v, \mathbf{S}') \leq d$. Then, the algorithm constructs an instance of CONRMC where the input matrix is $\mathbf{S}[P_*(\mathbf{S}[i])]$ and the distance bound is given by $d_{i'} = d - \delta(v, \mathbf{S}'[i'])$ for each $i' \in [n]$. The correctness proof was based on the erroneous assumption that the algorithm of Gramm et al. [11] finds *all* row vectors $v$ satisfying $\delta(v, \mathbf{S}') \leq d$ in time $O((d+1)^d \cdot n\ell)$. Although Gramm et al. [11] noted that this is indeed the case when $d$ is optimal, it is not always true. In fact, it is generally impossible to enumerate all solutions in time $O((d+1)^d \cdot n\ell)$ because there can be $\Omega(\ell^d)$ solutions. We use the following simple matrix to illustrate the error in the algorithm of Hermelin and Rozenberg [12]:

$$\mathbf{S} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ * & 0 & 0 \end{bmatrix}.$$

We show that the algorithm may output an incorrect answer for $d = 2$. If the algorithm chooses $i = 3$, then the algorithm by Gramm et al. [11] returns only one row vector 00. Then the algorithm of Hermelin and Rozenberg [12] constructs an instance of CONRMC with $\mathbf{S}' = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$ and $d_1 = d_2 = 0$, resulting in **No**. However, the row vector $v = 001$ satisfies $\delta(v, \mathbf{S}) = 2$ and thus the correct output is **Yes**. To remedy this, we give a fixed-parameter algorithm for MINRMC, adapting the algorithm by Gramm et al. [11].

Before presenting our algorithm, let us give an observation noted by Gramm et al. [11] for the case of no missing entries. Suppose that the input matrix $\mathbf{S} \in (\Sigma \cup \{*\})^{n \times \ell}$ contains more than $nd$ *dirty* columns (a column is said to be dirty if it contains at least two distinct symbols from the alphabet). Clearly, we can assume that every column is dirty. For any vector $v \in \Sigma^\ell$, there exists $i \in [n]$ with $\delta(v, \mathbf{S}[i]) \geq d$ by the pigeon hole principle and hence we can immediately conclude that it is a **No**-instance. It is easy to see that this argument also holds for MINRMC and thus CONRMC.

▶ **Lemma 8.** *Let $(\mathbf{S}, d_1, \ldots, d_n)$ be a CONRMC instance, where $\mathbf{S} \in (\Sigma \cup \{*\})^{n \times \ell}$ and $d_1, \ldots, d_n \in \mathbb{N}$. If $\mathbf{S}$ contains more than $nd$ dirty columns for $d = \max_{i \in [n]} d_i$, then there is no row vector $v \in \Sigma^\ell$ with $\delta(v, \mathbf{S}[i]) \leq d_i$ for all $i \in [n]$.*

Our algorithm is given in Algorithm 3. It generalizes Algorithm 2 and finds the solution vector even if the input matrix is incomplete. In contrast to NEIGHBORING STRING, the output cannot be immediately determined even if $d_1 = 0$. We use Algorithm 1 to overcome

◼ **Algorithm 3** Algorithm for CONRMC (generalizing Algorithm 2).

---

**Input:** An incomplete matrix $\mathbf{S} \in (\Sigma \cup \{*\})^{n \times \ell}$ and $d_1, \ldots, d_n \in \mathbb{N}$.
**Task:** Decide whether there exists a row vector $v \in \Sigma^\ell$ with $d(v, \mathbf{S}[i]) \leq d_i$ for all $i \in [n]$.

1: **if** $d_1 = 0$ **then**
2:     Let $\mathbf{S}' = \mathbf{S}[[2, n], P_*(\mathbf{S}[1])]$ and $d_i' = d_i - \delta(\mathbf{S}[1], \mathbf{S}[i])$ for each $i \in [2, n]$.
3:     **return** the output of Algorithm 1 on $(\mathbf{S}', d_2', \ldots, d_n')$.
4: Let $R_i = (P_*(\mathbf{S}[1]) \setminus P_*(\mathbf{S}[i])) \cup Q(\mathbf{S}[1], \mathbf{S}[i])$ for each $i \in [2, n]$.
5: **if** $|R_i| \leq d_i$ for all $i \in [2, n]$ **then return Yes**.
6: Choose any $i \in [n]$ with $|R_i| > d_i$.
7: Choose any $R \subseteq R_i$ with $|R| = d_i + 1$.
8: **for all** $j \in R$ **do**
9:     Let $\mathbf{S}' = \mathbf{S}[:, [\ell] \setminus \{j\}]$ and $d_{i'}' = d_i - \delta(\mathbf{S}[i, j], \mathbf{S}[i', j])$ for each $i' \in [n]$.
10:     **if** recursion on $(\mathbf{S}', d_1', \ldots, d_n')$ returns **Yes then return Yes**.
11: **return No**.

---

this issue (Line 3). Algorithm 3 also considers the columns where the first row vector has missing entries (recall that $P_*(\mathbf{S}[1])$ denotes column indices $j$ with $\mathbf{S}[1, j] = *$) in the branching step (Line 8), and not only the columns where $\mathbf{S}[1]$ and $\mathbf{S}[i]$ disagree. Again, we restrict the branching to $d_i + 1$ subcases (Line 7). This reduces the size of the search tree significantly. We show the correctness of Algorithm 3 and analyze its running time in the proof of the following theorem.

▶ **Theorem 9.** *For $d = \max_{i \in [n]} d_i$, CONRMC can be solved in $O(n\ell + (d+1)^{d+k+1} n)$ time.*

**Proof.** First, we prove that Algorithm 3 is correct by induction on $d_1 + |P_*(\mathbf{S}[1])|$. More specifically, we show that the algorithm returns **Yes** if and only if a vector $v \in \Sigma^\ell$ satisfying $\delta(\mathbf{S}[i], v) \leq d_i$ for all $i \in [n]$ exists.

Consider the base case $d_1 + |P_*(\mathbf{S}[1])| = 0$. Since $d_1 = 0$, the algorithm terminates in Line 3. When $d_1 = 0$, any solution vector must agree with $\mathbf{S}[1]$ on each entry unless the entry is missing in $\mathbf{S}[1]$. Hence, the output in Line 3 is correct by Theorem 5. Consider the case $d_1 + |P_*(\mathbf{S}[1])| > 0$. Let $R_i = (P_*(\mathbf{S}[1]) \setminus P_*(\mathbf{S}[i])) \cup Q(\mathbf{S}[1], \mathbf{S}[i])$ for each $i \in [2, n]$. If $|R_i| \leq d_i$ holds for all $i \in [2, n]$, then the vector $\mathbf{S}[1] \oplus \sigma^\ell$ (the vector obtained by filling each missing entry in $\mathbf{S}[1]$ with $\sigma$) is a solution for an arbitrary character $\sigma \in \Sigma$. Hence, Line 5 is correct. Suppose that there exists a solution vector $v \in \Sigma^\ell$ with $\delta(v, \mathbf{S}[i]) \leq d_i$ for all $i \in [n]$. We show that the branching in Line 8 is correct. Let $R$ be as specified in Line 7. We claim that there exists a $j \in R$ with $v[j] = \mathbf{S}[i, j]$ for every choice of $R$. Otherwise, $v[j] \neq \mathbf{S}[i, j]$ and $\mathbf{S}[i, j] \neq *$ holds for all $j \in R$ and we have $\delta(v, \mathbf{S}[i]) > d_i$ (a contradiction). Note that $\mathbf{S}[:, [\ell] \setminus \{j\}]$ has exactly one less missing entry if $j \in P_*(\mathbf{S}[1])$ and that $d_1' = d_1 - 1$ in case of $j \in Q(\mathbf{S}[1], \mathbf{S}[i])$. It follows that $d_1 + |P_*(\mathbf{S}[1])|$ is strictly smaller in the recursive call (Line 10). Hence, the induction hypothesis ensures that the algorithm returns **Yes** when $v[j] = \mathbf{S}[i, j]$ holds. On the contrary, it is not hard to see that the algorithm returns **No** if there is no solution vector. Thus, Algorithm 3 is correct.

We examine the time complexity. Assume without loss of generality that $k = |P_*(\mathbf{S}[1])|$ and $d = d_1$ hold initially. Consider the search tree where each node corresponds to a call on either Algorithm 1 or Algorithm 3. If $d_1 > 0$, then $d_1 + P_*(\mathbf{S}[1])$ decreases by 1 in each recursion and there are at most $d + 1$ recursive calls. Let $u$ be some node in the search tree that invokes Algorithm 1 for the first time. We have seen in the proof of Theorem 5 that the subtree rooted at $u$ is a tree of depth at most $|P_*(\mathbf{S}[1])|$, in which each node has at most

---

■ **Algorithm 4** Algorithm for NEIGHBORING STRING by Ma and Sun [18].

---

**Input:** A matrix $\mathbf{T} \in (\Sigma \cup \{*\})^{n \times \ell}$ and $d_1, \ldots, d_n \in \mathbb{N}$.
**Task:** Decide whether there exists a row vector $v \in \Sigma^\ell$ with $d(v, \mathbf{T}[i]) \leq d_i$ for all $i \in [n]$.
 1: **if** $\delta(\mathbf{T}[1], \mathbf{T}[i]) \leq d_i$ for all $i \in [n]$ **then return Yes** .
 2: Choose any $i \in [n]$ such that $\delta(\mathbf{T}[1], \mathbf{T}[i]) > d_i$.
 3: Let $Q = Q(\mathbf{T}[1], \mathbf{T}[i])$.
 4: **for all** $v \in \Sigma^{|Q|}$ such that $\delta(v, \mathbf{T}[1]) \leq d_1$ and $\delta(v, \mathbf{T}[i]) \leq d_i$ **do**
 5:     Let $\mathbf{T}' = \mathbf{T}[:, [\ell] \setminus Q]$ and $d_1' = \min\{d_1 - \delta(v, \mathbf{T}[1, Q]), \lceil d_1/2 \rceil - 1\}$.
 6:     Let $d_{i'}' = d_{i'} - d(v, \mathbf{T}[i', Q])$ for each $i' \in [2, n]$.
 7:     **if** recursion on $(\mathbf{T}, d_1', \ldots, d_n')$ returns **Yes then return Yes**.
 8: **return No**.

---

$d_i - \delta(\mathbf{S}[1], \mathbf{S}[i]) + 1 \leq d + 1$ children. Note also that $u$ lies at depth $d + k - |P_*(\mathbf{S}[1])|$. Thus, the depth of the search tree is at most $d + k$ and the search tree has size $O((d+1)^{d+k})$. We can assume that $\ell \leq nd$ by Lemma 8 and hence each node requires $O(nd)$ time. This results in the overall running time of $O(n\ell + (d+1)^{d+k+1}n)$.   ◄

Now, we provide a more efficient fixed-parameter algorithm when the alphabet size is small, based on an algorithm by Ma and Sun [18] (see Algorithm 4). Whereas Algorithm 2 considers each position of (a subset of) $Q(\mathbf{T}[1], \mathbf{T}[i])$ one by one, their algorithm considers all vectors on $Q(\mathbf{T}[1], \mathbf{T}[i])$ in a single recursion. The following lemma justifies that $d_1$ can actually be halved in each iteration (the vectors $u$ and $w$ correspond to $\mathbf{T}[1]$ and $\mathbf{T}[i]$, respectively).

▶ **Lemma 10.** *[18, Lemma 3.1]  Let $u, v, w \in \Sigma^\ell$ be row vectors satisfying $\delta(u, w) > \delta(v, w)$. Then, it holds that $\delta(u[Q'], v[Q']) < \delta(u, v)/2$ for $Q' = [\ell] \setminus Q(u, w)$.*

**Proof.** Assume that $\delta(u[Q'], v[Q']) \geq \delta(u, v)/2$. We can rewrite the value of $\delta(u, v) + \delta(v, w)$ as follows:

$$\delta(u, v) + \delta(v, w) = \delta(u[Q'], v[Q']) + \delta(v[Q'], w[Q']) + \delta(u[Q], v[Q]) + \delta(v[Q], w[Q]),$$

where $Q$ is a shorthand for $Q = Q(v, w)$. It follows from the definition of $Q'$ that $u[Q'] = w[Q']$ and hence

$$\delta(u[Q'], v[Q']) = \delta(v[Q'], w[Q']). \tag{1}$$

We also note that $\delta(u[Q] + v[Q]) + \delta(v[Q] + w[Q]) \geq |Q| = \delta(v, w)$ because it must hold that $u[j] \neq v[j]$ or $v[j] \neq w[j]$ for each $j \in Q$. Now, we obtain the following contradiction:

$$\delta(u, v) + \delta(v, w) \geq 2\delta(u[Q'], v[Q']) + \delta(u, w) > \delta(u, v) + \delta(v, w),$$

concluding the proof.   ◄

Lemma 10 plays a crucial role in obtaining the running time $O(n\ell + (16|\Sigma|)^d nd)$ of Ma and Sun [18]. However, Lemma 10 may not hold in the presence of missing entries. To work around this issue, let us introduce a new variant of CLOSEST STRING which will be useful to derive a fixed-parameter algorithm for CONRMC (Theorem 12). We will use a special character "◇" to denote a "dummy" character.

NEIGHBORING STRING WITH DUMMIES (NSD)

**Input:**     A matrix $\mathbf{T} \in (\Sigma \cup \{\diamond\})^{n \times \ell}$ and $d_1, \ldots, d_n \in \mathbb{N}$.
**Question:** Is there a row vector $v \in \Sigma^\ell$ such that $\delta(v, \mathbf{T}[i]) \leq d_i$ for each $i \in [n]$?

Note that the definition of NSD forbids dummy characters in the solution vector $v$. Observe that Lemma 10 holds even if row vectors $u$ and $w$ contain dummy characters. We show (based on Lemma 10) that NSD can be solved using the algorithm of Ma and Sun [18] as a subroutine.

▶ **Lemma 11.** *NSD can be solved in $O(n\ell + |\Sigma|^k \cdot nk + 2^{4d-3k} \cdot |\Sigma|^d \cdot nd))$ time, where $d := \max_{i \in [n]} d_i$ and $k$ is the minimum number of dummy characters in any row vector of $\mathbf{T}$.*

**Proof.** With Lemma 10, assuming $d = d_1$ one can prove by induction on $d_1$ that Algorithm 4 solves the NSD problem if the first row vector $\mathbf{T}[1]$ contains no dummy characters by induction on $d_1$. Refer to [18, Theorem 3.2] for details. We describe how we use Algorithm 4 of Ma and Sun [18] to solve NSD. Let $I = (\mathbf{T}, d_1, \ldots, d_n)$ be an instance of NSD. We assume that $|P_\diamond(\mathbf{T}[1])| = k$. For each row vector $u$ of $\Sigma^k$, we invoke Algorithm 4 with the input matrix $\mathbf{T}' = \mathbf{T}[[\ell] \setminus P_\diamond(\mathbf{T}[1])]$ and the distance bounds $d_1 - k, d_2 - \delta(u, \mathbf{T}[2, P_\diamond(\mathbf{T}[1])]), \ldots, d_n - \delta(u, \mathbf{T}[n, P_\diamond(\mathbf{T}[1])])$. Note that $\mathbf{T}'[1]$ contains no dummy character and thus the output of Algorithm 4 is correct. We return **Yes** if and only if Algorithm 4 returns **Yes** at least once. Let us prove that this solves NSD. If $I$ is a **Yes**-instance with solution vector $v \in \Sigma^\ell$, then it is easy to verify that Algorithm 4 returns **Yes** when $u = v[P_\diamond(\mathbf{T}[1])]$. On the contrary, the distance bounds in the above procedure ensure that $I$ is a **Yes**-instance if Algorithm 4 returns **Yes**.

Now we show that this procedure runs in the claimed time. Ma and Sun [18] proved that Algorithm 4 runs in

$$O\left(n\ell + \binom{d_{\max} + d_{\min}}{d_{\min}} \cdot (4|\Sigma|)^{d_{\min}} \cdot nd_{\max}\right)$$

time, where $d_{\max} = \max_{i \in [n]} d_i$ and $d_{\min} = \min_{i \in [n]} d_i$. In fact, they showed that each node in the search tree requires $O(nd_{\max})$ time by remembering previous distances, as it only concerns $O(d_{\max})$ columns. In the same spirit, one can compute distances from the first row vector for each NSD-instance under consideration in $O(nk)$ time, given the corresponding distances in the input matrix. Since we have $d_{\max} \leq d$ and $d_{\min} \leq d - k$ for each call of Algorithm 4, it remains to show that $\binom{2d-k}{d} \in O(2^{2d-k})$. Using Stirling's approximation $\sqrt{2\pi} n^{n+1/2} e^{-n} \leq n! \leq e n^{n+1/2} e^{-n}$ which holds for all positive integers $n$, we obtain

$$\binom{2d-k}{d} = \frac{(2d-k)!}{d! \cdot (d-k)!} \leq c \cdot \frac{(2d-k)^{2d-k}}{d^d \cdot (d-k)^{d-k}}$$

for some constant $c$. We claim that the last term is upper-bounded by $c \cdot 2^{2d-k}$. We use the fact that the function $x \mapsto x \log x$ is convex over its domain $x > 0$ (note that the second derivative is given by $x \mapsto 1/x$). Since a convex function $f \colon D \to \mathbb{R}$ satisfies $(f(x) + f(y))/2 \geq f((x+y)/2)$ for any $x, y \in D$, we obtain

$$d \log d + (d-k) \log(d-k) \geq 2\left(\frac{2d-k}{2}\right) \cdot \log\left(\frac{2d-k}{2}\right).$$

It follows that $d^d \cdot (d-k)^{d-k} = 2^{d \log d + (d-k) \log(d-k)} \geq 2^{(2d-k) \log(d-k/2)} = (d-k/2)^{2d-k}$. This shows that $\binom{2d-k}{d} \in O(2^{2d-k})$. ◀

Finally, to show our second main result in this section, we provide a polynomial-time reduction from CONRMC to NSD.

▶ **Theorem 12.** *CONRMC can be solved in $O(n\ell + 2^{4d+k} \cdot |\Sigma|^{d+k} \cdot n(d+k))$ time.*

$$\mathbf{S} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & * \\ * & * & 2 \end{bmatrix} \qquad\qquad \mathbf{T} = \begin{bmatrix} 0 & 0 & 0 & \diamond & \diamond \\ 1 & 1 & \diamond & \sigma & \diamond \\ \diamond & \diamond & 2 & \sigma & \sigma \end{bmatrix}$$

▪ **Figure 2** An illustration of the reduction in Theorem 12. Given the matrix $\mathbf{S}$ with $k = 2$ (left), our reduction constructs the matrix $\mathbf{T}$ with $k$ additional columns (right). Note that every row vector in $\mathbf{T}$ contains exactly two dummy characters. The MINRMC instance $(\mathbf{S}, d = 1)$ is a **Yes**-instance with a solution vector $v = 100$. The corresponding NSD-instance $(\mathbf{T}, d + k = 3)$ is also a **Yes**-instance with a solution vector $v' = 100\sigma\sigma$.

**Proof.** Let $I = (\mathbf{S}, d_1, \ldots, d_n)$ be an instance of CONRMC. We construct an instance $I' = (\mathbf{T}, d_1 + k, \ldots, d_n + k)$ of NSD where $\mathbf{T} \in (\Sigma \cup \{\diamond\})^{n \times (\ell + k)}$ and each row vector of $\mathbf{T}$ contains exactly $k$ dummy characters. Note that such a construction yields an algorithm running in $O(n(\ell + k) + |\Sigma|^k \cdot nk + 2^{4d+k} \cdot |\Sigma|^{d+k} \cdot n(d+k)) = O(n\ell + 2^{4d+k} \cdot |\Sigma|^{d+k} \cdot n(d+k))$ time using Lemma 11. Let $\sigma \in \Sigma$ be an arbitrary character. We define the row vector $\mathbf{T}[i]$ for each $i \in [n]$ as follows: Let $\mathbf{T}[i, [\ell]] = \mathbf{S}[i] \oplus \diamond^\ell$ (in other words, the row vector $\mathbf{T}[i, [\ell]]$ is obtained from $\mathbf{S}[i]$ by replacing $*$ by $\diamond$) for the leading $\ell$ entries. For the remainder, let

$$\mathbf{T}[i, \ell + j] = \begin{cases} \sigma & \text{if } j \leq |P_*(\mathbf{S}[i])|, \\ \diamond & \text{otherwise,} \end{cases}$$

for each $j \in [k]$ (Figure 2 shows an illustration). We claim that $I$ is a **Yes** instance if and only if $I'$ is a **Yes** instance.

($\Rightarrow$) Let $v \in \Sigma^\ell$ be a solution of $I$. We claim that the vector $v' \in \Sigma^{\ell+k}$ with $v'[[\ell]] = v$ and $v[[\ell + 1, \ell + k]] = \sigma^k$ is a solution of $I'$. For each $i \in [n]$, we have

$$\delta(v', \mathbf{T}[i]) = \delta(v'[[\ell]], \mathbf{T}[i, [\ell]]) + \delta(\sigma^k, \mathbf{T}[i, [\ell + 1, \ell + k]]).$$

It is easy to see that the first term is at most $d_i + |P_*(\mathbf{S}[i])|$ and that the second term equals $k - |P_*(\mathbf{S}[i])|$. Thus we have $\delta(v', \mathbf{T}) \leq d_i + k$.

($\Leftarrow$) Let $v' \in \Sigma^\ell$ be a solution of $I'$. Since the row vector $\mathbf{T}[i, [\ell + 1, \ell + k]]$ contains $k - |P_*(\mathbf{S}[i])|$ dummy characters, we have $\delta(v'[[\ell]], \mathbf{T}[i, [\ell]]) \leq (d_i + k) - (k - |P_*(\mathbf{S}[i])|) = d_i + |P_*(\mathbf{S}[i])|$ for each $i \in [n]$. It follows that $\delta(v'[[\ell]], \mathbf{S}[i]) \leq d_i$ holds for each $i \in [n]$. ◄

Note that the algorithm of Theorem 12 is faster than the algorithm of Theorem 9 for $|\Sigma| < d/16$ and faster than the $O^*(|\Sigma|^k \cdot d^d)$-time algorithm by Hermelin and Rozenberg [12] for $|\Sigma| < d/2^{4+d/k}$.

## 6    Conclusion

We studied problems appearing both in stringology and in the context of matrix completion. The goal in both settings is to find a consensus string (matrix row) that is close to all given input strings (rows). The special feature here now is the existence of wildcard letters (missing entries) appearing in the strings (rows). Thus, these problems naturally generalize the well-studied CLOSEST STRING and related string problems. Although with applications in the context of data mining, machine learning, and computational biology at least as well motivated as CLOSEST STRING, so far there is comparatively little work on these "wildcard problems". This work is also meant to initiate further research in this direction.

We conclude with a list of challenges for future research:

- Can the running time of Theorem 12 be improved? Since Ma and Sun [18] proved that Closest String can be solved in $O((16|\Sigma|)^d \cdot n\ell)$ time, a plethora of efforts have been made to reduce the base in the exponential dependence in the running time [6, 19, 4, 5, 22]. A natural question is whether these results can be translated to MinRMC and ConRMC as well.

- Another direction would be to consider the generalization of MinRMC with *outliers*. The task is to determine whether there is a set $I \subseteq [n]$ of row indices and a vector $v \in \Sigma^\ell$ such that $|I| \leq t$ and $\delta(v, \mathbf{S}[[n] \setminus I]) \leq d$. For complete input matrices, this problem is known as Closest String with Outliers and fixed-parameter tractability with respect to $d + t$ is known [2]. Hence, it is interesting to study whether the outlier variant of MinRMC (or ConRMC) is fixed-parameter tractable with respect to $d + k + t$.

- Finally, let us mention a maximization variant MaxRMC where the goal is to have a radius at least $d$. The case of complete input matrices is referred to as Farthest String [21] and fixed-tractability with respect to $|\Sigma| + d$ is known [10, 21]. Is MaxRMC also fixed-parameter tractable with respect to $|\Sigma| + d$?

---- **References** ----

**1** Amihood Amir, Jessica Ficler, Liam Roditty, and Oren Sar Shalom. On the efficiency of the Hamming c-centerstring problems. In *25th Symposium on Combinatorial Pattern Matching (CPM '14)*, pages 1–10. Springer, 2014.

**2** Christina Boucher and Bin Ma. Closest string with outliers. *BMC Bioinformatics*, 12(S-1):S55, 2011.

**3** Laurent Bulteau, Falk Hüffner, Christian Komusiewicz, and Rolf Niedermeier. Multivariate algorithmics for NP-hard string problems. *Bulletin of the EATCS*, 114, 2014.

**4** Zhi-Zhong Chen, Bin Ma, and Lusheng Wang. A three-string approach to the closest string problem. *Journal of Computer and System Sciences*, 78(1):164–178, 2012.

**5** Zhi-Zhong Chen, Bin Ma, and Lusheng Wang. Randomized fixed-parameter algorithms for the closest string problem. *Algorithmica*, 74(1):466–484, 2016.

**6** Zhi-Zhong Chen and Lusheng Wang. Fast exact algorithms for the closest string and substring problems with application to the planted $(\ell, d)$-motif model. *IEEE/ACM Transactions Computational Biology and Bioinformatics*, 8(5):1400–1410, 2011.

**7** Eduard Eiben, Robert Ganian, Iyad Kanj, Sebastian Ordyniak, and Stefan Szeider. On clustering incomplete data. *CoRR*, abs/1911.01465, 2019. URL: http://arxiv.org/abs/1911.01465.

**8** Moti Frances and Ami Litman. On covering problems of codes. *Theory of Computing Systems*, 30(2):113–119, 1997.

**9** Robert Ganian, Iyad A. Kanj, Sebastian Ordyniak, and Stefan Szeider. Parameterized algorithms for the matrix completion problem. In *35th International Conference on Machine Learning (ICML '18)*, volume 80 of *Proceedings of Machine Learning Research*, pages 1642–1651. PMLR, 2018.

**10** Jens Gramm, Jiong Guo, and Rolf Niedermeier. Parameterized intractability of distinguishing substring selection. *Theory of Computing Systems*, 39(4):545–560, 2006.

**11** Jens Gramm, Rolf Niedermeier, and Peter Rossmanith. Fixed-parameter algorithms for closest string and related problems. *Algorithmica*, 37(1):25–42, 2003.

**12** Danny Hermelin and Liat Rozenberg. Parameterized complexity analysis for the closest string with wildcards problem. *Theoretical Computer Science*, 600:11–18, 2015. Preliminary version appeared at *CPM '14*.

**13** Dušan Knop, Martin Koutecký, and Matthias Mnich. Combinatorial $n$-fold integer programming and applications. *Mathematical Programming*, 2019.

**14**     Tomohiro Koana, Vincent Froese, and Rolf Niedermeier. Complexity of combinatorial matrix completion with diameter constraints. *CoRR*, abs/2002.05068, 2020. `arXiv:2002.05068`.

**15**     Ming Li, Bin Ma, and Lusheng Wang. Finding similar regions in many sequences. *Journal of Computer and System Sciences*, 65(1):73–96, 2002.

**16**     Ming Li, Bin Ma, and Lusheng Wang. On the closest string and substring problems. *Journal of the ACM*, 49(2):157–171, 2002.

**17**     Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. *SIAM Journal on Computing*, 47(3):675–702, 2018.

**18**     Bin Ma and Xiaoming Sun. More efficient algorithms for closest string and substring problems. *SIAM Journal on Computing*, 39(4):1432–1443, 2009.

**19**     Naomi Nishimura and Narges Simjour. Enumerating neighbour and closest strings. In *7th International Symposium on Parameterized and Exact Computation (IPEC '12)*, pages 252–263. Springer, 2012.

**20**     Carsten Sinz. Towards an optimal CNF encoding of boolean cardinality constraints. In *11th International Conference on Principles and Practice of Constraint Programming (CP '05)*, pages 827–831. Springer, 2005.

**21**     Lusheng Wang and Binhai Zhu. Efficient algorithms for the closest string and distinguishing string selection problems. In *3rd International Frontiers in Algorithmics Workshop (FAW '09)*, pages 261–270. Springer, 2009.

**22**     Shota Yuasa, Zhi-Zhong Chen, Bin Ma, and Lusheng Wang. Designing and implementing algorithms for the closest string problem. *Theoretical Computer Science*, 786:32–43, 2019.