A QPTAS for Gapless MEC*

Shilpa Garg[1] and Tobias Mömke[2]

[1]Max Planck Institute for Informatics, Saarland Informatics Campus, Germany.
sgarg@mpi-inf.mpg.de
[2]Saarland University, Saarland Informatics Campus, Germany.
moemke@cs.uni-saarland.de

October 18, 2017

Abstract

We consider the problem Minimum Error Correction (MEC). A MEC instance is an $n \times m$ matrix M with entries from $\{0, 1, -\}$. Feasible solutions are composed of two binary m-bit strings, together with an assignment of each row of M to one of the two strings. The objective is to minimize the number of mismatches (errors) where the row has a value that differs from the assigned solution string. The symbol "$-$" is a wildcard that matches both 0 and 1. A MEC instance is gapless, if in each row of M all binary entries are consecutive.

GAPLESS-MEC is a relevant problem in computational biology, and it is closely related to segmentation problems that were introduced by [Kleinberg–Papadimitriou–Raghavan STOC'98] in the context of data mining.

Without restrictions, it is known to be UG-hard to compute an $O(1)$-approximate solution to MEC. For both MEC and GAPLESS-MEC, the best polynomial time approximation algorithm has a logarithmic performance guarantee. We partially settle the approximation status of GAPLESS-MEC by providing a quasi-polynomial time approximation scheme (QPTAS). Additionally, for the relevant case where the binary part of a row is not contained in the binary part of another row, we provide a polynomial time approximation scheme (PTAS).

*Research partially funded by Deutsche Forschungsgemeinschaft grant MO2889/1-1.

1 Introduction.

The minimum error correction problem (MEC) is a segmentation problem where we have to partition a set of length m strings into two classes. A MEC instance is given by a set of n strings over $\{0, 1, -\}$ of length m, where the symbol "$-$" is a wildcard symbol. The strings are represented by an $n \times m$ matrix M, where the ith string determines the ith row $M_{i,*}$ of M.

The distance dist of two symbols a, a' from $\{0, 1, -\}$ is

$$\mathrm{dist}(a, a') := \begin{cases} 1: & a = 0, a' = 1 \text{ or } a = 1, a' = 0 \\ 0: & \text{otherwise.} \end{cases}$$

For two strings s, s' from $\{0, 1, -\}^m$ where s_j, s'_j denotes the j-th symbol of the respective string, $\mathrm{dist}(s, s') := \sum_{j=1}^{m} \mathrm{dist}(s_j, s'_j)$. A feasible solution to the MEC is a pair of two strings σ, σ' from $\{0, 1\}^m$. The optimization goal is to find a feasible solution (σ, σ') that minimizes

$$\mathrm{cost}(\sigma, \sigma') := \sum_{i=1}^{n} \min\{\mathrm{dist}(M_{i,*}, \sigma), \mathrm{dist}(M_{i,*}, \sigma')\}\,.$$

A MEC instance is called *gapless* if in each of the n rows of M, all entries from $\{0, 1\}$ are consecutive. (As regular expression, a valid row is a word of length m from the language $-^*\{0, 1\}^*-^*$). The MEC problem restricted to gapless instances is GAPLESS-MEC.

Our motivation to study GAPLESS-MEC stems from its applications in computational biology. Humans are diploid, and hence there exist two versions of each chromosome. Determining the DNA sequences of these two chromosomal copies – called haplotypes – is important for many applications ranging from population history to clinical questions [**?**, **?**]. Many important biological phenomena such as compound heterozygosity, allele-specific events like DNA methylation or gene expression can only be studied when haplotype-resolved genomes are available [**?**].

Existing sequencing technologies cannot read a chromosome from start to end, but instead deliver small pieces of the sequences (called reads). Like in a jigsaw puzzle, the underlying genome sequences are reconstructed from the reads by finding the overlaps between them.

The upcoming next-generation sequencing technologies (e.g., Pacific Biosciences) have made the production of relatively long continuous sequences with sequencing errors feasible, where the sequences come from both copies of chromosome. These sequences are aligned to a reference genome or to a structure called contig. We can formulate the result of this process as a GAPLESS-MEC instance.

We can therefore formulate haplotyping as the problem of reconstructing the genomes from sequences represented in the form of GAPLESS-MEC instances by correcting the sequencing errors. The use of sequencing datasets for complete haplotype phasing is quickly becoming reality [**?**, **?**].

GAPLESS-MEC is a generalization of a problem called BINARY-MEC, the version of MEC with only instances M where all entries of M are in $\{0, 1\}$. Finding an optimal solution to BINARY-MEC is equivalent to solving the hypercube 2-segmentation problem (H2S) which was introduced by Kleinberg, Papadimitriou, and Raghavan [**?**, **?**] and which is known to be NP-hard [**?**, **?**]. The optimization version of BINARY-MEC differs from H2S in that we minimize the number of mismatches instead of maximizing the number of matches. BINARY-MEC allows for good approximations. Ostravsky and Rabiny [**?**] obtained a PTAS for BINARY-MEC based on random embeddings. Building on the work of Li et al. [**?**], Jiao et al. [**?**] presented a deterministic PTAS for BINARY-MEC.

GAPLESS-MEC was shown to be NP-hard by Cilibrasi et al. [**?**].[1] Additionally, they showed that allowing a single gap in each strings renders the problem APX-hard. More recently, Bonizzoni et al. [**?**] showed that it

[1] Their result predates the hardness result of Feige [**?**] for H2S. The proof of the claimed NP-hardness of H2S by Kleinberg, Papadimitriou, and Raghavan [**?**] was never published.

is unique games hard to approximate MEC with constant performance guarantee, whereas it is approximable within a logarithmic factor in the size of the input. To our knowledge, previous to our result their logarithmic factor approximation was also the best known approximation algorithm for GAPLESS-MEC.

1.1 Our results.

We provide a quasi-polynomial time approximation scheme (QPTAS) for GAPLESS-MEC and thus partially settle the approximability for this problem: GAPLESS-MEC is not APX-hard unless NP $\subseteq$ QP (cf. [**?**]). Thus our result reveals a separation of the hardness of the gapless case and the case where we allow a single gap. Furthermore, already BINARY-MEC is strongly NP-hard since the input does not contain numerical values. Therefore we can exclude the existence of an FPTAS for both BINARY-MEC and GAPLESS-MEC unless P $=$ NP.

Furthermore, we address the class of *subinterval-free* GAPLESS-MEC instances where no string is contained in another string. More precisely, for each pair of rows from M we exclude that the set columns with binary entries from one row is a strict subset of the set of columns with binary entries from the other row. We provide a PTAS for subinterval-free instances.

1.2 Overview of our approach.

Our algorithm is a dynamic program (DP) that is composed of several levels. Given a general GAPLESS-MEC instance, we decompose the rows of the instance into length classes according to the length of the continuous binary parts of the rows. For each length class we consider a well-selected set of columns such that each row crosses at least one the columns and at most two. (We say that a row crosses a column if the binary part of the row contains that column.)

We further decompose each length class into two sub-classes, one that crosses exactly one column and one that crosses exactly two columns. For the second class, it is sufficient to consider every other column, which leaves us with many *rooted* instances. Thus for each sub-instance there is a single column (the root) which is crossed by all rows of the instance.

We further decompose rooted sub-instances into the left hand side and the right hand side of the root. We can arrange the rows and columns of these instances in such a way that all rows cross the first column. We order the rows from top to bottom by increasing length.

The first level of our DP solves these highly structured instances. The basic idea that we want to apply is that we want to select a constant number of rows from the instance that represents the solution. This strategy fails because of differing densities within the instance: the selected rows have to represent both the entries of columns crossed by many short rows and entries of arbitrarily small numbers of rows crossing many columns. To resolve this issue, we observe that computing the solution strings σ and σ' is equivalent to finding a partition of M into two row sets, one assigned to σ and the other assigned to σ'. If we assume to have the guarantee that for both solution strings σ and σ' an ε fraction of rows of the matrix M forms a BINARY-MEC sub-instance, we show that the idea above works.

This insight motivates to separate the instance from left to right into sub-problems with the required property and to assemble them from left to right using a DP. There are, however, several complications. In order to choose the right sub-instances, we have to take into account that the choice depends on which rows are assigned to σ and which are assigned to σ'. Therefore the DP has to take special care when identifying the sub-instances.

Furthermore, in order to stitch sub-instances together to form a common solution, the solution computed in the left sub-instance has to compute its solution oblivious of the choices of the right sub-instance. This means that we have to compute a solution to the left sub-instance without looking at a fraction of rows.

We build up the DP by considering the smallest units and assambling them to bigger units, i.e., we proceed in reverse order of the separation into sub-instances described above. In order to have a clear presentation of the result, we first show, in Section **??**, that it is possible to compute a $(1+O(\varepsilon))$-approximate BINARY-MEC solution without looking at an ε fraction of rows. We then generalize these results in order to be able to handle the sub-instances that we will solve repeatedly in our DP, in Section **??**.

In order to combine the sub-instances, we face technical complications due to having distinct sub-problems for those rows assigned to σ and those rows assigned to σ'. In order to separate these complications from the actual combination of sub-problems, we first consider instances for which we only want to have one solution string, in Section **??**. We note that these instance would be easily solvable by simply selecting the majority value in each column, but our aim is something different. As needed for the actual instances, in the DP we restrict our view and only considering constantly many rows at a time. Afterwards, in Section **??**, we show how to combine our insights with taking care of both σ and σ'.

Before we consider rooted instances, we first develop our techniques by considering subinterval-free instances which are easier to handle (Section **??**). Observe that the instances considered until now are special rooted sub-interval-free instances. We show how to solve arbitrary rooted sub-interval-free instances by combining the DP with additional information about the sub-problems that contain the root. We then introduce the notion of domination in order to combine rooted sub-interval-free instances with a DP proceeding from left to right.

Until this point, all of our algorithms are running in polynomial time. When considering arbitrary rooted instances, in Section **??**, the left hand side of the root may have a completely different structure than the right hand side of the root. If we allow, however, quasipolynomial running time, we can solve the problem. We use that each of the two sub-instances (the one on the left and the one on the right) is composed of at most logarithmically many sub-problems. Considering all of the sub-problems simultaneously allows us to take care of dependencies between the left hand side and the right hand side and still solve them as if they were separate instances.

We next consider GAPLESS-MEC instances where all strings are in the same length class, i.e., the longest binary part is at most twice as long as the shortest binary part of a row (Section **??**). As described earlier, we consider two sets of rows such that for each of these sets there is a sequence of columns such that each row crosses exactly one of the columns. Combining such rooted instances from left to right then can be done in the same spirit as combining rooted sub-interval-free instances. To solve the entire length-class, we combine both solutions by running a new DP that considers quadruples of DP cells.

Finally, in Section **??**, we are able to solve all length classes simultaneously by noting that the total number of different length classes is logarithmic in the input. We solve general instances in the same spirit as the combined sub-instances of a single length class. Instead of considering quadruples of cells, however, we use $\log(n)$-vectors of quadruples. This adds another power of $\log(n)$ to the running time, which is still quasi-polynomial.

1.3 Further related work.

Binary-MEC is a variant of the Hamming k-Median Clustering Problem when $k = 2$ and there are PTASs known [**?**, **?**]. Li, Ma, and Wang [**?**] provided a PTAS for the general consensus pattern problem which is closely related to MEC. Additionally, they provided a PTAS for a restricted version of the star alignment problem aligning with at most a constant number of gaps in each sequence.

Alon [**?**] provided a PTAS for H2S, the maximization version of BINARY-MEC and Wulff, Urner and Ben-David [**?**] showed that there is also a PTAS for the maximization version of MEC. For GAPLESS-MEC, He et al. [**?**] studied the fixed-parameter tractability in the parameter of fragment length with some restrictions. These restrictions allow their dynamic programming algorithm to focus on the reconstruction of a single haplotype and, hence, to limit the possible combinations for each column. There is an FPT algorithm

parameterized by the coverage [**?**, **?**] (and some additional parameters for pedigrees). Bonizzoni et al. [**?**] provided FPT algorithms parameterized by the fragment length and the total number of corrections for Gapless-MEC. There are some tools which can be used in practice to solve Gapless-MEC instances [**?**, **?**].

Most research in haplotype phasing deals with exact and heuristic approaches to solve BINARY-MEC. Exact approaches, which solve the problem optimally, include integer linear programming [**?**] and fixed-parameter tractable algorithms [**?**, **?**]. There is a greedy heuristic approach proposed to solve Binary-MEC [**?**].

Lancia et al. [**?**] obtained a network-flow based polynomial time algorithm for Minimum Fragment Removal (MFR) for gapless fragments. Additionally, they found the relation of Minimum SNPs Removal (MSR) to finding the largest independent set in a weakly triangulated graph.

1.4 Preliminaries and notation.

We consider a GAPLESS-MEC instance, which is a matrix $M \in \{0,1,-\}^{n\times m}$. The ith row of M is the vector $M_{i,*} \in \{0,1,-\}^{1\times m}$ and the jth column is the vector $M_{*,j} \in \{0,1,-\}^{n\times 1}$. The length of the binary part in $M_{i,*}$ is $|M_{i,*}|$. We say that the ith row of M *crosses* the jth column if $M_{i,j} \in \{0,1\}$.

For each feasible solution (σ,σ') for M, we specify an assignment of rows $M_{i,*}$ to solution strings. The default assignment is specified as follows. For a row $M_{i,*}$, we assign $M_{i,*}$ to σ if $\mathrm{dist}(\sigma, M_{i,*}) \leq \mathrm{dist}(\sigma', M_{i,*})$. Otherwise we assign $M_{i,*}$ to σ'. For the rows of M assigned to σ we write $\sigma(M)$ and for the rows assigned to σ' we write $\sigma'(M)$. For a given instance, $\mathsf{Opt} = (\tau,\tau')$ denotes an optimal solution. Observe that knowing Opt allows us to obtain an optimal assignments $\tau(M)$ and $\tau'(M)$ by assigning each row to the solution string with fewest errors and knowing $\tau(M)$ and $\tau'(M)$ allows us to obtain an optimal solution by selecting the column-wise majority values.

2 All rows cross column one.

Suppose we are given an instance of GAPLESS-MEC where all entries of column one in M are zero or one. Formally, we consider GAPLESS-MEC with the restriction that in M, $M_{i,1} \in \{0,1\}$ for each index i. We order the rows such that $|M_{i,*}| \leq |M_{i+1,*}|$ for each i, i.e., we order them from top to bottom in increasing length of the binary part.

We first build the tools that we need for the simplified setting of BINARY-MEC instances. We sometimes consider the majority string of a matrix. The function MAJORITY computes a string of length m by setting the jth entry to the most frequent entry of $M_{*,j}$, for each column j.

In order to compute BINARY-MEC solutions, we consider the algorithm BINARY_δ which is based on the PTAS for BINARY-MEC obtained by Jiao et al. [**?**]. We state BINARY_δ as a randomized algorithm in order to reveal the core ideas in a simple form. Due to the manner in which we want to use the algorithm later on, we introduce parameters R, r, R', r'. R and R' are sets of rows from which the algorithm is allowed to sample. The other two parameters r and r' are integers. We ensure in the algorithm that the computed solution has $|\sigma(M)| = r$ and $|\sigma'(M)| = r'$. The intuition is that we guess $|\tau(M)|$ and $|\tau'(M)|$ and set r, r' to these values. We note that the randomized algorithm BINARY_δ requires the knowledge of an optimal solution (τ,τ').

A simple derandomization helps us to find a solution without knowledge of (τ,τ'): We replace the random selection of $1/\delta$ rows by trying all of the $O\big(n^{2/\delta}\big)$ possible selections of two row sets $\tilde{R}, \tilde{R}'$ of size $1/\delta$ each. For each selection, we compute $\sigma = \text{MAJORITY}(\tilde{R})$ and $\sigma' = \text{MAJORITY}(\tilde{R}')$. We keep the solution (σ,σ') with minimal value $\mathrm{cost}(\sigma,\sigma')$. Since there are only polynomial many possibilities, the algorithm runs in polynomial time. We observe that BINARY_δ with input matrix M and $R = R' = M$, $r = |\tau(M)|$, $r' = |\tau'(M)|$ computes a BINARY-MEC solution to M; here we can guess the values $|\tau(M)|, |\tau'(M)|$ by trying all possibilities.

Algorithm 1: BINARY$_\delta$

Input : Two sets of selected rows R, R' from a BINARY-MEC instance M, two numbers r, r'.
Output : A pair of strings (σ, σ'), an assignment of the rows to these strings.
Select uniformly at random (with repetition) a multi-set $\tilde{R}$ of $1/\delta$ strings from $R \cap \tau(M)$ and a multi-set $\tilde{R}'$ of $1/\delta$ strings from $R' \cap \tau'(M)$;
Set $\sigma = \text{MAJORITY}(\tilde{R})$ and $\sigma' = \text{MAJORITY}(\tilde{R}')$;
For each row i, determine the value $d_i := \text{dist}(\sigma, M_{i,*}) - \text{dist}(\sigma', M_{i,*})$;
Assign the r rows with maximal values d_i to σ and the remaining r' rows to σ'.

Lemma 1. *For $r = |\tau(M)|$ and $r' = |\tau'(M)|$ let R, R' be sets of rows such that $|\tau(R)| \geq (1-\varepsilon)r$ and $|\tau'(R')| \geq (1-\varepsilon)r'$. Then the derandomized version of* BINARY$_\varepsilon$ *is a $(1 + O(\varepsilon))$-approximation algorithm for* BINARY-MEC.

Lemma **??** is a special case of Lemma **??**. We therefore postpone the proof.

We are now ready to continue with a first (simplified) version of a technical key lemma which we will refer to as "partial input lemma." Intuitively, the lemma states that we can find a $(1 + O(\varepsilon))$ approximation for BINARY-MEC, even if a fraction of rows (depending on ε) is hidden.

Lemma 2. *For a* BINARY-MEC *instance M and sufficiently small $\varepsilon > 0$, let R be an arbitrary set of rows from M with at least $(1-\varepsilon)|\tau(M)|$ rows from $\tau(M)$ and R' an arbitrary set of rows from M with at least $(1-\varepsilon)|\tau'(M)|$ rows from $\tau'(M)$. Let $r = |\tau(R)|$ and $r' = |\tau(R')|$. Let (σ, σ') be the solution computed by* BINARY$_\varepsilon$ *for the matrix $P = R \cup R'$. Then (σ, σ') is a $(1 + O(\varepsilon))$-approximation for the instance M.*

Let $Q := M \setminus P$ where we identify M with its set of rows. The claim still holds if within Q, we assign $\tau(Q)$ to σ and $\tau'(Q)$ to σ'.

Proof. For ease of presentation, we assume that all appearing numbers are integers. It is easy to adapt the proof by rounding fractional numbers appropriately.

We first analyze the computed solution string σ. The matrix M has n rows and m columns. Let η be the total number of errors of (τ, τ') within M and let η' be the total number of errors of (σ, σ') within P. Clearly $\eta' \leq (1 + O(\varepsilon))\eta$. We may assume $r \geq r'$ since otherwise we can simply rename the two strings τ, τ'. Additionally, by renaming of σ and σ', we may assume that $|\sigma(P) \cap \tau(P)| \geq |\sigma'(P) \cap \tau(P)|$. Therefore $|\tau(P)| \geq n/3$ and $|\sigma(P) \cap \tau(P)| \geq n/6$.

Claim 2.1. There is a set I of $m - 25\eta/n$ indices i such that $\sigma_i = \tau_i$ for all $i \in I$.

Proof of Claim. We concentrate on the columns of M where both strings τ and σ have at most $n/12$ errors within P. By counting the errors, there are at most $12\eta/n$ columns where τ has at least $n/12$ errors. Similarly, there are at most $12(1 + O(\varepsilon))\eta'/n < 13\eta/n$ many columns where σ has at least $n/12$ errors. Therefore there is a set I of at least $m - 25\eta/n$ columns where simultaneously both τ and σ have less than $n/12$ errors each.

Now suppose that the claim was not true and there was an index $i \in I$ with $\tau_i \neq \sigma_i$. Then, since $|\tau(P) \cap \sigma(P)| \geq n/6$, either σ_i or τ_i is erroneous in at least $n/12$ rows of $\tau(P) \cap \sigma(P)$, a contradiction. $\diamondsuit$

Next we analyze σ' for the columns I. Let j be a column (i.e., an index) from I. By symmetry, we may assume $\sigma_j = \tau_j = 0$. We aim to show that an optimal solution has always sufficiently many errors to pay for wrong decisions.

Let η_j be the number of error of (τ, τ') in column j of M and let η'_j be the number of errors of (σ, σ') in column j of P. Let $\eta''_j = \eta_j + \eta'_j$.

Claim 2.2. For each column j of I, either $\sigma'_j = \tau'_j$ or $\eta''_j \geq (1-\varepsilon)|\tau'(M)|/2$.

Proof of Claim. We distinguish two cases. We first assume $\tau'_j = 0$ and therefore $\sigma'_j = 1$. If there are more than $r'/2$ ones in column j of R', (τ, τ') has more than $r'/2$ errors in column j and thus $\eta_j \geq |\tau'(R')|/2$. Otherwise $\sigma'(R)$ has at least $r'/2$ zeros in column j and therefore $\eta'_j \geq |\tau'(R')|/2$. We obtain $\eta''_j \geq |\tau'(R')|/2 \geq (1-\varepsilon)|\tau'(M)|/2$ as claimed.

In the second case, $\tau'_j = 1$ and we assume that $\sigma'_j = 0$. If there are more than $r'/2$ ones in column j of R', (σ, σ') has more than $r'/2$ errors in column j and thus $\eta'_j \geq |\tau'(R')|/2$. Otherwise $\tau'(R')$ has at least $r'/2$ zeros in column j and therefore $\eta_j \geq |\tau'(R')|/2$. Again, we obtain $\eta''_j \geq |\tau'(R')|/2 \geq (1-\varepsilon)|\tau'(M)|/2$ as claimed. ◊

Since by our assumption $|\tau'(Q)| < \varepsilon|\tau'(M)|$, Claim **??** implies that within I we have a $(1 + O(\varepsilon))$-approximation.

To finish the proof, we argue that η is large enough to pay for all errors in Q outside of I. Due to our discussion regarding σ' in I, it is save to set $\sigma(Q) := \tau(Q)$ and $\sigma'(Q) := \tau'(Q)$. Then (σ, σ') has at most $O(\varepsilon)\eta$ more errors than (τ, τ') within Q for the columns of I. Let this number of errors be η_I.

Then, using the size of I stated in Claim **??**, the total number of errors of (σ, σ') in M is at most $(1 + O(\varepsilon))\eta + \eta_I + \varepsilon n \cdot 25\eta/n$, i.e., the errors of $\text{BINARY}_\varepsilon$ within P, the errors within Q in the columns of I, and all other entries of Q. The obtained approximation ratio is

$$\frac{(1 + O(\varepsilon))\eta + \eta_I + \varepsilon n \cdot 25\eta/n}{\eta} \leq \frac{\eta + O(\varepsilon)\eta + 25\varepsilon\eta}{\eta} = 1 + O(\varepsilon)\,.$$

The first inequality uses that for some constant k, $(1 + k\varepsilon)\eta \geq \eta + \eta_I$. □

2.1 Simple instances with wildcards.

Next we show that we can extend the PTAS for BINARY-MEC to a specific class of GAPLESS-MEC instances $\mathcal{G}$. Recall that we consider GAPLESS-MEC instances M where the first column of M has only binary entries and the rows are ordered in increasing length of the binary part. An instance is in $\mathcal{G}$ if it satisfies these conditions and additionally, there are at least $\varepsilon|\tau(M)|$ rows of $\tau(M)$ and at least $\varepsilon|\tau'(M)|$ rows of $\tau'(M)$ that have only entries from $\{0, 1\}$. For an instance $G \in \mathcal{G}$, we consider several sets of consecutive rows. In the following, values r and r' have the same meaning as in BINARY_δ.

Let R_A be the set of rows such that (i) all rows with wildcards are contained in R_A, (ii) there are exactly[2] $(1-\varepsilon)r$ rows from $\tau(G)$ in R_A, (iii) R_A contains the first row, and (iv) R_A is the minimal set with these properties. We added point (iv) merely to avoid ambiguity. The set R_B contains the rows after R_A such that (v) there are $(\varepsilon - \varepsilon^2)r$ rows from $\tau(G)$ in R_B and (vi) R_B is the minimal set with this property. Then the set R_C contains all remaining rows of G. The row sets R_A, R_B, R_C for one solution string are shown in Figure **??**. We define R'_A, R'_B, and R'_C analogously, but replace r and τ by r' and τ'. We assume that all considered terms are integers.

We introduce a new algorithm SWC_δ (Simple Wildcards) for our setting. The algorithm is similar to BINARY_δ: we consider two row sets R and R' and sample rows from these sets. But additionally, we partition them into $2/\varepsilon^2$ disjoint subsets. Let $\mathcal{R} := \{R_1, R_2, \ldots, R_{2/\varepsilon^2}\}$ and $\mathcal{R}' := \{R'_1, R'_2, \ldots, R'_{2/\varepsilon^2}\}$. We want that the first $1/\varepsilon^2$ sets of $\mathcal{R}$ and $\mathcal{R}'$ are partitions of R_A and R'_A, and the second $1/\varepsilon^2$ sets are partitions of R_B and R'_B. For a multiset of rows S let $\text{MAJORITY}_j(S)$ be the *weighted* majority restricted to column j defined as follows. We only consider sets of rows R_i such that in column j, all entries of R_i are in $\{0, 1\}$; otherwise we remove all copies of rows in $R_i \cap S$ from S. For $\ell := 1/\varepsilon^2$ let $U_j := S \cap \bigcup_{i \leq \ell} R_i$ and $L_j = S \cap \bigcup_{i > \ell} R_i$. In U_j and L_j, we replace all zeros by -1 and compute the number $\nu := \sum_{i \in U_j} (1-\varepsilon)i/(\varepsilon - \varepsilon^2) + \sum_{i \in L_j} i$. Then $\text{MAJORITY}_j(S) = 0$ if $\nu < 0$ and $\text{MAJORITY}_j(S) = 1$ if $\nu \geq 0$. The intuitive meaning is that we

[2]To be precise, this assumption introduces a small imprecision which needs careful but straightforward handling.

sample rows from the upper part with a much lower density then the rows of the lower part. We therefore introduce a bias such that all rows are sampled with the same marginal probability. Let (τ, τ') be an optimal solution.

Algorithm 2: SWC_δ

Input : An instance $G \in \mathcal{G}$, partitions $\mathcal{R}$ and $\mathcal{R}'$ of row sets R and R' into $2/\varepsilon^2$ classes, numbers r, r'.
Output: A pair of strings (σ, σ').
For each i, we consider $R_i \in \mathcal{R}, R'_i \in \mathcal{R}'$. Select uniformly at random (with repetition) a multi-set $\tilde{R}_i$ of $1/\delta$ strings from each $R_i \cap \tau(G)$ and a multi-set $\tilde{R}'_i$ of $1/\delta$ strings from $R'_i \cap \tau'(G)$;
Set $\tilde{R} := \bigcup_i \tilde{R}_i$ and $\tilde{R}' := \bigcup_i \tilde{R}'_i$; `// Note that` $\tilde{R} \cap \tilde{R}' = \emptyset$.
For each column j, set $\sigma_j := \textsc{Majority}_j(\tilde{R})$ and $\sigma'_j := \textsc{Majority}_j(\tilde{R}')$;
For each row i, determine the value $d_i := \mathrm{dist}(\sigma, G_{i,*}) - \mathrm{dist}(\sigma', G_{i,*})$;
Assign the r rows with maximal values d_i to σ and the remaining r' rows to σ'.

We derandomize SWC_δ analogous to $\textsc{Binary}_\delta$. Again, the knowledge of (τ, τ') is required in the randomized algorithm but not in the derandomized one.

Observe that for small (i.e., constant) values of r or r', the algorithm SWC_δ can be replaced by an exact algorithm since we know $\tau(G)$ if and only if we know $\tau'(G)$, and we are able to guess constantly many rows.

Lemma 3. *For sufficiently large $r = |\tau(G)|$ and $r' = |\tau'(G)|$, let $R = R_A \cup R_B$ and $R' = R'_A \cup R'_B$ be sets of rows as described above. For $\ell = \varepsilon^{-2}$, let $\mathcal{R} = \{R_1, R_2, \ldots, R_{2\ell}\}$ and $\mathcal{R}' = \{R'_1, R'_2, \ldots, R'_{2\ell}\}$ be partitions of R, R' such that (i) for each set R_i with $i \leq \ell$, $|\tau(R_i)| = \varepsilon^2 \cdot (1-\varepsilon)r$; (ii) for each set R_i with $i > \ell$, $|\tau(R_i)| = \varepsilon^2 \cdot (\varepsilon - \varepsilon^2)r$; (iii) for each set R'_i with $i \leq \ell$, $|\tau'(R'_i)| = \varepsilon^2 \cdot (1-\varepsilon)r'$; and (iv) for each set R'_i with $i > \ell$, $|\tau'(R'_i)| = \varepsilon^2 \cdot (\varepsilon - \varepsilon^2)r'$.*

Then the derandomized version of $\mathrm{SWC}_{\varepsilon^3}$ is a $(1 + O(\varepsilon))$-approximation algorithm for instances from $\mathcal{G}$.

The proof is based on using Chernoff bounds and can be found in Appendix **??**. With this preparation we are able to generalize Lemma **??** to work with SWC_δ instead of $\textsc{Binary}_\delta$. We obtain a generalized version of the partial input lemma.

Lemma 4. *For a* GAPLESS-MEC *instance $M \in \mathcal{G}$ and sufficiently small $\varepsilon > 0$, let R be an arbitrary set of rows from M with at least $(1-\varepsilon^2)|\tau(M)|$ rows from $\tau(M)$ and R' an arbitrary set of rows from M with at least $(1-\varepsilon^2)|\tau'(M)|$ rows from $\tau'(M)$. Let $r = |\tau(M)|$, $r' = |\tau(M')|$, and $\mathcal{R}, \mathcal{R}'$ as required in Lemma **??**. Let (σ, σ') be the solution computed by $\mathrm{SWC}_{\varepsilon^3}$ for the matrix $P = R \cup R'$. Then (σ, σ') is a $(1 + O(\varepsilon))$-approximation for the instance M.*

Let $Q := M \setminus P$ where we identify M with its set of rows. The claim still holds if within Q, we assign $\tau(Q)$ to σ and $\tau'(Q)$ to σ'.

The new setup only requires small changes in the proof of Lemma **??**. Instead of the entire instance M, for the analysis we have to focus on the sub-matrix without $R_A \cap R'_A$. All remaining changes are straightforward.

For the dynamic program, we need that the selection of rows withing R_B provides an interface between two sub-instances. The following lemma follows directly from our previous analyses.

Lemma 5. *Let us consider a run of $\mathrm{SWC}_{\varepsilon^3}$ with the parameters of Lemma **??**, but instead of a random sampling in R_B, R'_B, we consider an arbitrary feasible selection of rows $\tilde{R} \cap R_B, \tilde{R}' \cup R_B$. Suppose that with this selection we obtain an α-approximation for P, then we obtain an $\alpha + O(\varepsilon)$-approximation for M.*

2.2 A simplified DP for a single solution string.

To show how to use Lemma **??** in our DP, we first focus on how to determine one of the two solution strings. To this end we assume that we already know $\tau(M)$ and $\tau'(M)$ and that in $\mathrm{SWC}_{\varepsilon^3}$ we have $\sigma' = \tau'$. We restrict our access to these sets, however, to only considering a constant number of rows at a time. Let $r = |\tau(M)|$ and $r' = |\tau(M)|$.

We design a dynamic program with DP cells for each combination of the following parameters. (a) Three consecutive ranges of rows determined by numbers $1 \leq a < b < c < d \leq n$. These numbers determine an upper range R_A from row a to row $b-1$, a lower range R_B from row b to row $c-1$ and a bottom range R_C from c to d. (b) A multiset T_U of $1/\varepsilon^5$ rows from R_A. (c) A multiset T_L of $1/\varepsilon^5$ rows from R_B. (d) A partition of T_U into $1/\varepsilon^2$ ranges of rows, determined by numbers $a = a_1 < a_2 < \cdots < a_{1/\varepsilon^2} < b$. (e) A partition of T_L into $1/\varepsilon^2$ ranges of rows, determined by numbers $b = b_1 < b_2 < \cdots < b_{1/\varepsilon^2} < c$. Among these choices, we only consider cells where each range in (d) and (e) contains $1/\varepsilon^3$ rows from T_U resp. T_L.

For each DP cell ζ we store a string σ^ζ from column 1 to the end of the first string of R_B, i.e., row b of M. (Recall that the rows are ordered in increasing length from top to bottom.)

The intuitive meaning of a DP cell is that the first class R_A contains a $1-\varepsilon$ fraction of rows assigned to τ within the considered range and R_B contains an $\varepsilon - \varepsilon^2$ fraction. The selected rows of T_U and T_L should be thought of as a random selection of rows from $\tau(M)$ (with repetition) such that in each range from (c) and (d) there are $1/\varepsilon^3$ sampled rows.

A DP cell corresponds to an instance $D_\zeta = R_A, R_B, R_C$ for which we only consider the columns before the end of the bth row.

The algorithm works in two phases. The first phase is an initialization. We assign an initial DP value to *each* DP cell. The value specifies the approximate number of errors if the DP cell is the first one that contributes to the solution (with smallest value a). Afterwards, we will update the initial values in a second phase of the algorithm.

We initialize each cell ζ by applying Lemma **??** on D_ζ, but instead of sampling rows in $\mathrm{SWC}_{\varepsilon^3}$, we use T_U and T_L. The selection of rows takes the role of the derandomization step. The partition $\mathcal{R}$ is determined by (d) and (e). Since we only consider one solution string, we do not have to fix r', R' and $\mathcal{R}'$. The result is the string σ^ζ that does not depend on R_C.

Let B be the sub-matrix of M that contains the rows R_B and all columns where the first row of R_B has entries in $\{0,1\}$. In order to update the DP values and to analyze the quality of the computed solution we count the number of errors within $\tau(M)$ restricted to the columns of B. For each computed string σ^ζ, the value of its DP cell is the number of errors within $\tau(M)$ between rows a and c.

For all DP cells with $a = 1$ in (a), we already have assigned the final value in the initialization step. We now iteratively update the cells for $a \geq 2$. Let ζ be the cell that we want to update and $\tilde{\zeta}$ another cell with the parameters marked by the symbol prime. We say that $\tilde{\zeta}$ is a *predecessor* of ζ if $a = \tilde{b}$; and $b = \tilde{c}$; and $c = \tilde{d}$; and $T_U = \tilde{T}_L$, where the variables marked with tilde are the parameters of $\tilde{\zeta}$.

To compute ζ, we assume that all of its predecessors are updated already. The DP selects the predecessor $\tilde{\zeta}$ with the minimal DP value (i.e., the minimum number of errors within the prefix). To compute σ^ζ, we consider the columns of D_ζ after $\sigma^{\tilde{\zeta}}$, i.e., exactly those columns for which we have not computed a solution yet. We apply Lemma **??** on the obtained instance analogous to the initialization phase. By appending the obtained solution after $\sigma^{\tilde{\zeta}}$ we obtain a new candidate string for σ^ζ. We compare the old value of ζ with the number of errors in D_ζ between a and c. If the candidate string incurs fewer errors than σ^ζ, we update ζ: its value is the new number of errors and we store the candidate string. After considering all predecessors of ζ, we have computed its final value. In particular, now we can also use it as a predecessor of other cells. Eventually all possible update operations are done. It might happen, however, that we were not able to compute the entire solution yet. The reason is that valid DP cells as specified select a large number of rows, which may not be possible in the end. In order to finish the DP, we additionally consider special cells that are

defined as before, but with $c = d = n$. Intuitively, we use these cells when only $1/\varepsilon^3$ rows of $\tau(M)$ are left. For these cells, our computation considers the optimal solution for the suffix of σ.

Lemma 6. *If we can determine the number of errors within $M(\tau)$ with respect to each prefix of σ, then the algorithm above is a PTAS.*

Proof. Since all binary strings are feasible solutions, our algorithm vacuously produces a valid solution.

To analyze the quality of the computed solution, we partition $\tau(M)$ into ranges. Starting from the top-most row of $\tau(M)$, for each $i \geq 0$, the ith range R_i contains the next $(\varepsilon^i - \varepsilon^{(i+1)})r$ rows of $\tau(M)$. To be consistent with properties needed in later proofs, we ensure that the first row of each R_i is contained in $\tau(M)$ and thus we add the rows between R_i and R_{i+1} to R_i. We note that if only a constant number of rows of $\sigma(M)$ are left, we can compute the partial solutions optimally and there are DP cells for exactly this purpose: there is a DP cell ζ_i such that the last $1/\varepsilon^3$ rows of $\tau(M)$ are located between a and b and R_i contains exactly these rows. To keep a clean notation, in the following we implicitly assume that cells with constantly many rows of $\sigma(M)$ are handled separately.

For each range we obtain blocks corresponding to those of Figure **??**. The block A_0 contains the rows of R_0 and the columns one to the end of the first row of $\tau(R_0)$. For each $i > 0$, block A_i contains the rows of R_i and the columns after those of A_{i-1} to the end of the first row of R_i. Then B_i is composed of the columns of A_i and A_{i+1} and of the rows of A_{i+1}. The block C_i has the same columns as B_i but the rows of A_{i+2}.

We consider the DP cells ζ_i for each i with the following parameters. (a) $\hat{a}_i$ is the first row of A_i, $\hat{b}_i$ the first row of B_i, $\hat{c}_i$ the first row of C_i, and $\hat{d}_i$ the last row of C_i. (b),(c) T_U and T_L are sets of $1/\varepsilon^5$ rows from $\tau(A_i)$ and $\tau(B_i)$, respectively, satisfying the requirements of the DP cells. These rows are selected in such a way that applying $\mathrm{SWC}_{\varepsilon^3}$ they are at most the expected solution value obtained by picking uniformly at random with repetition. (d), (e) the parameters $\hat{a}_{i,j}$ and $\hat{b}_{i,j}$ for $j = 1, 2, \ldots, 1/\varepsilon^3$ partition $\tau(A_i)$ and $\tau(B_i)$ into sets of equal sizes.

We obtain sub-instances D_i for each $i \geq 0$ such that D_i contains all rows from $\hat{a}_i$ on and the columns from the end of row $\hat{a}_i$ to the end of row $\hat{b}_i$ The remaining parameters $r_i, R_i, \mathcal{R}_i$ follow directly from (a) to (e). For each D_i there is exactly one DP cell ζ_i.

We inductively show that the expected value of each cell the value of ζ_i is at most a factor $(1 + \varepsilon)$ larger than the number of errors of an optimal solution restricted to the considered prefix. For $i = 0$ we only consider D_0 and the invariant follows directly from Lemma **??**.

Suppose now that $i \geq 0$ and for all $\tilde{i} < i$ the invariant is true. Then we consider the subinstance D_i.

We apply Lemma **??** to compute the string σ^{ζ_i} for D_i. We obtain a $(1 + \varepsilon)$ for the prefix covered by σ^{ζ_i} for the following reason. The part before D_i was fixed, and independent of the rows from $\hat{b}_i$ on already gave a $(1 + \varepsilon)$ approximation. The part of σ^{ζ_i} within D_i gives a $(1 + \varepsilon)$ approximation by the claim of Lemma **??**.

Observe that these results hold without considering any row from $\hat{c}_i$ on, and by Lemma **??** we only have to consider the rows of D_i to ensure this property. We can therefore continue the induction until the entire string σ is determined. □

2.3 Adding the second string.

Next we show how to embed a dynamic program for σ' into the dynamic program for σ. To distinguish the two parts, we call one dynamic program the σ-DP and the other one the σ'-DP.

A σ-DP cell ζ and a σ'-DP cell ζ' have the same parameters as our previous DP. (Again, we mark the parameters of ζ' with the symbol prime). The cell ζ' is a *child* of ζ if $b' > b$ and $T_U \cup T_L$ is disjoint to $T'_U \cup T'_L$. The child relationship is shown in Figure **??**. The intuition is that the string $\sigma^{\zeta'}$ is shorter than σ^{ζ}. We define a child of ζ' analogously by changing the roles of σ and σ'. Observe that a σ-DP cell cannot be a child of another σ-DP cell and no σ'-DP cell a child of another σ'-DP cell. The child relationship is orthogonal to to the predecessor relationship defined in Section **??**: a cell ζ can be a child of a cell ζ' even

though a predecessor of ζ' is the child of ζ. In contrast to the child-relationship, the predecessor relationship always refers to the same type of cell: either both are cells from the σ-DP or both are cells from the σ'-DP.

The new dynamic program has a DP-cell for each pair (ζ, ζ') of a σ cell ζ and each child ζ' of ζ and, conversely, each pair (ζ', ζ) of a σ' cell ζ' and each child ζ of ζ'. Additionally, we store two numbers r and r' that are used to guess $|\tau(M)|$ and $|\tau'(M)|$.

By renaming the strings we may assume that the DP starts with a σ-DP. For each σ-DP cell we run an initialization similar to Section **??**. At this point we cannot compute the values of the σ-DP cell since they depend on σ'. Nevertheless, the content of σ^ζ in $\mathrm{SWC}_{\varepsilon^3}$ only depends on ζ and not on ζ'. We can therefore assume σ^ζ to be known within the initialization phase. For each cell (ζ, ζ'), we store a pair of solution strings $\rho(\zeta, \zeta')$. These two strings have the same length as $(\sigma^\zeta, \sigma^{\zeta'})$, but the content is determined by the DP. The value of a cell (ζ, ζ') is the minimum number of errors when considering $\rho(\zeta, \zeta')$ in all rows of M above b'.

We compute the value and pair of strings of the DP cell (ζ, ζ') as follows. Let us first assume that neither ζ nor ζ' has a predecessor. Then the first string of $\rho(\zeta, \zeta')$ is σ^ζ and the second string is $\sigma^{\zeta'}$, as we do not rely on previous DP entries at this point. Let j be the index of the last entry of $\sigma^{\zeta'}$. Then the value of (σ, σ') is the minimum number of errors in the submatrix of M formed by the rows above b' when we assign the strings of $\rho(\zeta, \zeta')$ to these rows optimally. The intuition is that these are the only rows for which we have complete information. At the same time, these are the rows that we want to forget about in the subsequent DP.

We inductively assume that all DP cells for predecessors of ζ' have been computed already. There are two cases how the DP can proceed. For the second iteration of the DP, these cases are shown in Figure **??**. We continue with the same ζ. The first case is that we consider a σ'-cell ζ'' that is a child of ζ and has predecessors that are also children of ζ. We assume that for all cells including these predecessors we have computed the final DP value. We determine the infix of σ and σ' after the end of the first row of T''_U until the last column of the last row of T''_U as we did before. Then we consider the predecessors $\tilde{\zeta}$ of ζ'' that are children of ζ. We take the $\tilde{\zeta}$ with minimal DP value of $(\zeta, \tilde{\zeta})$. Our new prefix of σ' is composed of the prefix of $(\zeta, \tilde{\zeta})$ and the new infix. The value of (ζ, ζ'') is the total number of errors above b'', the version of b' for ζ''.

The crux of the DP is the "switch" between σ and σ' that happens in the second case, if the subsequent σ'-cell is not a child of σ. Let ζ'' be a σ'-cell that is not a child of σ. Clearly, every meaningful ζ'' here has the property that $T_U \cup T_L$ is disjoint to $T''_U \cup T''_L$. Thus we require that ζ is a child of ζ''. For each predecessor ζ' of ζ'' such that ζ' is a child of ζ, we compute the infix of σ' after the prefix of σ' determined by (ζ, ζ') until the end of the last row of T''_U. The difference, however, is that we do not know all values of σ: the computed infix of σ' reaches beyond the prefix determined for σ. The solution to the new situation is to switch the role of σ and σ': we now compute the cell (ζ'', ζ). The value of this cell is the minimum number of errors above b over all choices of predecessors ζ'.

Now the case where also ζ has a predecessor follows analogous to the case where ζ' has a predecessor by switching the roles of σ and σ'.

As in the previous section, we consider DP cells with $c = d = n$ in order to terminate the computation correctly.

Theorem 1. *The combined DP is a PTAS.*

Proof. To see that the DP works in polynomial time, we observe that instead of simple DP cells in Lemma **??** here we consider pairs of DP cells. Therefore the number of cells is squared and thus stays polynomial. During the recursive construction of the solution, we compare each cell to be computed with one compatible cell at a time. Therefore the construction of the solution also takes only polynomial time. As in Lemma **??**, the computed solution is vacuously feasible.

We continue with analyzing the quality of the computed solution. Let (τ, τ') be an optimal solution. We set $r = |\tau(M)|$ and $r' := |\tau'(M)|$. By renaming the two strings we may assume that the last row of the first $(1 - \varepsilon)r$ rows of $\tau(M)$ is below the first row of the last $\varepsilon r'$ rows of $\tau'(M)$. Figuratively this means that in Fig. **??** the bottom of A and the top of B' overlap.

We consider DP cells similar to the proof of Lemma **??**. Starting from the top-most row of $\tau(M)$, for each $i \geq 0$, the ith range R_i contains the next $(\varepsilon^i - \varepsilon^{i+1})r$ rows of $\tau(M)$. The rows not in $\tau(M)$ can be assigned to the ranges arbitrarily. To make the selection compatible with the guessing of ranges, we ensure that the first row of each R_i is contained in $\tau(M)$. Then we choose R_i such that all rows of M until R_{i+1} are contained in R_i. Again, if only a constant number of rows of $\sigma(M)$ are left, we can compute the partial solutions optimally and there are DP cells for exactly this purpose: there is a DP cell ζ_i such that the last $2/\varepsilon^5$ rows of $\tau(M)$ are located between a and b and R_i contains exactly these rows. As before, to keep a clean notation, in the following we implicitly assume that cells with constantly many rows of $\sigma(M)$ are handled separately.

For each range we obtain blocks corresponding to those of Figure **??**. The block A_0 contains the rows of R_0 and the columns one to the end of the first row of $\tau(A)$. For each $i > 0$, block A_i contains the rows of R_i and the columns after those of A_{i-1} to the end of the first row of R_i. Then B_i is composed of the columns of A_i and A_{i+1} and of the rows of A_{i+1}. The block C_i has the same columns as B_i but the rows of A_{i+2}.

We consider the DP cells ζ_i for each i with the following parameters. (a) a_i is the first row of A_i, b_i the first row of B_i, c_i the first row of C_i, and d_i the last row of C_i. (b),(c) T_U and T_L are a sets of $1/\varepsilon^5$ rows from $\tau(A_i)$ and $\tau(B_i)$, respectively, chosen in such a way that they match the expected value of a selection chosen uniformly at random with repetition that respects $\mathcal{R}$. (d) the parameters $a_{i,j}$ for $j = 1, 2, \ldots, 1/\varepsilon^3$ partition τi into sets of equal sizes. Analogously we define $a'_i, A'_i, b'_i, B'_i, c'_i, C'_i, d'_i$ for ζ'_i.

We construct a solution SOL and inductively show that the expected value of each considered cell (ζ_i, ζ'_j) and (ζ'_i, ζ_j) is at most a factor $(1 + O(\varepsilon))$ larger than the number of errors of an optimal solution restricted to the considered prefix. Afterwards we show that our algorithm computes a solution at least as good as SOL.

We first consider the DP cell (ζ_0, ζ'_0). Recall that we assumed w.l.o.g. that ζ'_0 is a child of ζ_0. We apply Lemma **??** with the parameters of the pair of cells to obtain the prefixes σ^{ζ_0} and $\sigma'^{\zeta'_0}$. The total number of errors within the columns of the prefixes is therefore at most a factor $(1 + \varepsilon)$ larger than in (τ, τ').

Next suppose that ζ'_1 is a child of ζ_0. Then, similar to the proof of Lemma **??**, we apply Lemma **??** to obtain $\sigma'^{\zeta'_1}$. Now we argue that the choice of the new DP cell ensures that the new (longer) solution prefixes $\rho(\sigma_0, \sigma'_1)$ again give a $(1 + \varepsilon)$-approximation. Since σ^{ζ_0} does not depend on the choices of (ζ_0, ζ'_1), we only have to consider the second string. By Lemma **??**, the selected rows in B'_1 are sufficient to ensure that we can afford not to look at C'_1. Since the DP cell dictates the choice of rows within A'_1 and B'_1, and σ'_0 is a predecessor of σ'_1 (with the same selection within A'_1), the choice of the predecessor cell can only influence the solution by a factor $(1 + \varepsilon)$. Observe that we consider each column in only a constant number of cells. This ensures that the overall error adds up to at most a factor $(1 + O(\varepsilon))$.

The argument does not depend on σ_0 to be the first cell of the instance and can therefore be iterated. The case with $(\zeta'_i, \zeta_{i'})$ is analogous.

Finally suppose that we have computed the value of $(\zeta_i, \zeta'_{i'})$ and $\zeta'_{i'+1}$ is *not* a child of ζ_i. Observe that now ζ_i is a child of $\zeta'_{i'+1}$. As before, we use Lemma **??** to bound the error of $\sigma'^{B'\zeta'_{i'+1}}$. The only difference is that now $\sigma'^{\zeta'_{i'+1}}$ is fixed and we argue about ζ_i as a child. □

3 Subinterval-free instances.

We show how to generalize the results of the previous section in order to handle instances where no interval of a string s is a proper subinterval of a string s'. To this end, we first show how to handle the rooted version of sub-interval free instances, where there is one column j such that each string of the instance crosses j.

Lemma 7. *Let M be a* GAPLESS-MEC *instance such that no string is the substring of another string. Furthermore we assume that there is a column j of M such that each string of the instance crosses j. Then there is a PTAS for M.*

Proof. We order the rows of M from top to bottom such that for each pair i, i' of rows with the binary part of i starting on the left of the binary part of i', i is above i'. In other words, the binary strings are ordered from top to bottom with increasing starting position (i.e., column). Observe that the sub-string freeness property ensures that the last binary entry of i' is not on the left of the last binary entry of i.

Let s and t be the first and the last row of M. The column j determines a block b of M that spans all rows and the columns from the first binary entry of t, j_t, to the last binary entry of s, j_s. In particular, b has only binary entries.

Observe that the right hand side of j_t (the submatrix of M composed of all columns with index at least j_t) forms a GAPLESS-MEC instance as required in Theorem **??**. Similarly, the submatrix of M that contains all rows of M and columns 1 to j_s forms a GAPLESS-MEC instance as required in Theorem **??** if we invert both the order of the rows and the columns. Instead of changing the ordering of the matrix, we can run the algorithm from right to left and from bottom to top.

We would like to apply Theorem **??** independently to the two specified sub-problems. To this end we define a special set of DP cells γ with cells $(\zeta_W, \zeta'_W) \in \gamma$. The content of these cells is similar to the regular cells, but it contains the information for both sides simultaneously. More precisely, a cell ζ_W has the following entrees.

(a) Three consecutive ranges of rows determined by numbers $1 \leq \overleftarrow{d} < \overleftarrow{c} < \overleftarrow{b} < \overrightarrow{b} < \overrightarrow{c} < \overrightarrow{d} \leq n$. These numbers determine an upper range R_A from row $\overleftarrow{b}+1$ to row $\overrightarrow{b}-1$ and the following further ranges. A left lower range $\overleftarrow{R}_L$ from row $\overleftarrow{b}$ to row $\overleftarrow{c}+1$ and a left bottom range $\overleftarrow{R}_B$ from $\overleftarrow{c}$ to $\overleftarrow{d}$, as well as a right lower range $\overrightarrow{R}_L$ from row $\overrightarrow{b}$ to row $\overrightarrow{c}-1$ and a right bottom range $\overrightarrow{R}_B$ from $\overrightarrow{c}$ to $\overrightarrow{d}$. (b) A multiset T_U of $1/\varepsilon^5$ rows from R_A. (c) A multiset $\overleftarrow{T}_L$ of $1/\varepsilon^5$ rows from $\overleftarrow{R}_L$. (c') A multiset $\overrightarrow{T}_L$ of $1/\varepsilon^5$ rows from $\overrightarrow{R}_L$. (d) A partition of T_U into $1/\varepsilon^2$ classes, determined by numbers $\overleftarrow{b} = a_1 < a_2 < \cdots < a_{1/\varepsilon^2} < \overrightarrow{b}$. (e) A partition of T_L into $1/\varepsilon^2$ classes, determined by numbers $\overleftarrow{b} = \overleftarrow{b}_1 < \overleftarrow{b}_2 < \cdots < \overleftarrow{b}_{1/\varepsilon^2} < \overleftarrow{c}$. (e') A partition of $\overrightarrow{T}_L$ into $1/\varepsilon^2$ classes, determined by numbers $\overrightarrow{b} = \overrightarrow{b}_1 < \overrightarrow{b}_2 < \cdots < \overrightarrow{b}_{1/\varepsilon^2} < \overrightarrow{c}$.

We analogously obtain ζ'_W with the same variables but marked with the symbol prime. The rows selected in ζ'_W are required to be disjoint from those in ζ_W.

The cells $(\zeta_W, \zeta'_W) \in \gamma$ takes a special role as common "center" of two separate DPs . Observe that for each feasible entry of (ζ_W, ζ'_W), we can apply Theorem **??** independently to the left and to the right, since the DP cells (ζ_W, ζ'_W) takes the role of the left-most cell in Theorem **??**. The strings only overlap between the columns j_t, j_s where we obtain an instance of BINARY-MEC. The two solution strings within this block are determined by the rows from $\overleftarrow{c}$ to $\overrightarrow{c}$ and from $\overleftarrow{c}'$ to $\overrightarrow{c}'$, depending only on (ζ_W, ζ'_W). None of the remaining steps from Section **??** interfere with each other. We therefore run the following DP. We first compute all cells (ζ_W, ζ'_W). For each cell, we store an infix of σ and an infix of σ'. The infix of σ starts at t_j and ends at s_j. The entries of the two strings are those that we obtain from $\mathrm{SWC}_{\varepsilon^3}$. (In this case, it is even sufficient to apply $\mathrm{BINARY}_{\varepsilon}$.)

To see that the DP yields a good enough approximation, again we compare against an optimal solution (τ, τ'). Clearly we get a $(1 + O(\varepsilon))$-approximation for the infix between column j_t and j_s if for a DP cell (ζ_W, ζ'_W) that simulates a choice of rows that is uniformly at random and ranges satisfying the conditions of Lemma **??**. By Lemma **??**, we can ensure that the computed solution does not consider the rows above $\overleftarrow{c}$ or below $\overrightarrow{c}$. Since the further processing respects our choice between $\overleftarrow{c}$ and $\overrightarrow{c}$, the claim follows from Theorem **??**. □

We now generalize Lemma **??** to general sub-interval free instances. Instead of a single column j crossed by all strings, we determine a sequence $q = (q_1, q_2, \dots)$ of columns with the property that each string crosses exactly one of them. Let s_1 be the first string in M. Then we choose column q_1 to be the column of the last entry of s_1.

We recursively specify the remaining columns. For a given j such that we know q_j, let s_i be the last (i.e., bottom-most) string that crosses q_j. Then we choose q_{j+1} to be the last (i.e., rightmost) column of string s_{i+1}. For each q_i in the sequence q, we determine a block W_i analogous to W in Lemma **??**.

A simple induction shows that by the no-substring property and the chosen order of strings, each string crosses at least one column of q and none of them crosses more than one. In particular, for each j, the solution on the left hand side of q_j depends on rows of M disjoint from the rows that determine the solution on the right hand side of q_{j+1}.

In order to combine the solution on the right hand side of q_j with the solution on the left hand side of q_{j+1}, we introduce a notion of dominance. Consider a block $\overrightarrow{T}$ of M that only contains rows that cross q_i and a block $\overleftarrow{T}$ of M that only contains rows that cross q_{i+1}. We say that $\overrightarrow{T}$ τ-dominates $\overleftarrow{T}$ if for each column c of $\overrightarrow{T}$ with a non-zero number of binary entries of c in $\tau(\overrightarrow{T})$, the number of binary entries in $\tau(\overrightarrow{T})$ is at least $1/\varepsilon$ the number in $\tau(\overleftarrow{T})$. We analogously define the τ-dominance of $\overleftarrow{T}$ and τ'-dominance.

We observe that if $\overrightarrow{T}$ is τ-dominant over $\overleftarrow{T}$ for some column c, it is also τ-dominant for all columns on the left hand side of c: until q_i is reached, when moving to the left the number of binary entries of $\tau(\overrightarrow{T})$ increases and the number of binary entries of $\tau(\overleftarrow{T})$ decreases. Analogously, if $\overleftarrow{T}$ is τ-dominant over $\overrightarrow{T}$ for some column c, it is also τ-dominant for all columns on the right hand side of c.

We therefore have a possibly empty interval I without τ-dominance such that the columns of $\overrightarrow{T}$ on the left hand side of I are τ-dominant and the columns of $\overleftarrow{T}$ on the right hand side of I are τ-dominant. The different cases for I are shown in Figures **??** and **??**.

The *dominance region* of $\overrightarrow{T}$ is the set of columns where $\overrightarrow{T}$ is dominant over $\overleftarrow{T}$, and vice versa. Within the dominance region, our old DP can simply compute solutions without considering interferences: the dominated block is small enough to be ignored by Lemma **??**.

Within the interval I, both cells have to "cooperate." Effectively we obtain a BINARY-MEC block in the middle with a slope on top and a slope on the bottom. This sub-instance can be solved directly. We handle it similar to the center in Lemma **??**.

We use DP cells similar to Lemma **??**, but with the following differences. For some j, let us consider column $q_j \in q$. Then we consider such a collection κ_j of special DP cells $(\zeta_{q_j}, \zeta'_{q_j}) \in \kappa_j$, with the same properties as (ζ_W, ζ'_W). We refer to them as the jth center cells. Additionally, for each center cell we also store the dominance regions on the left and right of q_j (i.e., we guess the boundaries of the left and the right interval I). Formally this means to extend the cells by for numbers that encode two intervals $\overleftarrow{I}$ and $\overrightarrow{I}$. For each of the regions, the cells also store the intervals in the same manner as before: numbers $\overleftarrow{c}_1, \overleftarrow{c}_2, \ldots, \overleftarrow{c}_{1/\varepsilon^2}$ that partition those strings crossing q_j that reach into I but do not necessarily span over I, numbers $\overleftarrow{d}_1, \overleftarrow{d}_2, \ldots, \overleftarrow{d}_{1/\varepsilon^2}$ of the remaining rows that entirely span over I, and the same type of partition $\overrightarrow{c}_1, \overrightarrow{c}_2, \ldots, \overrightarrow{c}_{1/\varepsilon^2}$ and $\overrightarrow{d}_1, \overrightarrow{d}_2, \ldots, \overrightarrow{d}_{1/\varepsilon^2}$ for $\overrightarrow{I}$. For each of these row ranges, the cell contains a selection of $1/\varepsilon^3$ rows. These values all belong to ζ_{q_j} and we add the same type of values for ζ'_{q_j}.

We now determine an infix of σ and σ' at $\overrightarrow{I}$ and $\overleftarrow{I}$ by applying $\mathrm{SWC}_{\varepsilon^3}$, again with the random selection replaced by the DP selection.

We embed our old DP into an outer DP that processes the entries of q from left to right. For each cell $(\zeta_{q_1}, \zeta' q_1)$, we compute the prefixes of σ, σ' until q_1 exactly as in Lemma **??**. We start the DP to the right hand side also the same way as before, but with the difference that as soon as we reach the row ranges for $\overrightarrow{I}$, we use the already fixed choice instead of guessing new rows.

For all $j > 1$ continue in the same manner starting from $(\zeta_{q_j}, \zeta'_{q_j})$ and handle the processing of $\overleftarrow{I}$ as we did before with $\overrightarrow{I}$. Observe that we can see the processing of $(\zeta_{q_1}, \zeta'_{q_1})$ as a special case with empty interval $\overleftarrow{I}$, and to obtain the suffix of σ, σ', the last interval $\overrightarrow{I}$ can be handled as empty interval.

Theorem 2. *Let M be a* GAPLESS-MEC *instance such that no string is the substring of another string. Then the DP above is a PTAS for M.*

Proof. Let (τ, τ') be an optimal solution. For each separate $q_j \in q$, we run the same DP as in Lemma **??** and thus we obtain a $1+\varepsilon$-approximation. For the intervals $\overleftarrow{I}$ and $\overrightarrow{I}$, there is a choice of parameters that matches the choices analyzed in Lemma **??**. We therefore only have to argue that the transition between sub-instances works correctly. We consider the dominant regions determined by (τ, τ') and consider the DP cells that guess these regions correctly from left to right. Let I be one of the guessed regions without dominance. Then the number of errors that the computed solution (σ, σ') has within this region is at most the number of errors in the solution of $\mathrm{SWC}_{\varepsilon^3}$, and we apply Lemma **??** with an empty ignored region to show that also the in-fixes of (σ, σ') at I give a $(1+\varepsilon)$-approximation, analogous to the centers in Lemma **??**. □

4 A QPTAS for general instances.

In the previous section, we crucially used that there are specific columns with the property that we could identify a sub-problem belonging to the left hand side and distinct sub-problem belonging to the right hand side. The intersection of both sides was taken care of using a single DP cell which was enough to separate both sides. Without the non-inclusion property, we lose this nice structure.

The issue already appears in the special case of *rooted* instances (shown in Figure **??**(a)) where there is a single column c (the root) such that each string of the instance crosses c. We can solve rooted GAPLESS-MEC, however, if we allow quasi-polynomial running time.

We apply two distinct orderings of rows to obtain our sub-problems. We first order the rows with increasing starting positions as in Lemma **??**. An optimal solution (τ, τ') then determines a sequence of sub-matrices $\overleftarrow{A}_1, \overleftarrow{A}_2, \ldots, \overleftarrow{A}_k$. Instead of moving from $\overleftarrow{A}_1$ to $\overleftarrow{A}_k$ using a DP, however, we now guess the strings for all k sub-matrices *simultaneously*. Additionally, we guess the matrices $\overleftarrow{A}'_1, \overleftarrow{A}'_2, \ldots, \overleftarrow{A}'_{k'}$ simultaneously. We obtain a combined DP cell $\overleftarrow{\zeta}$ for $k+k'$ sub-matrices. Afterwards we reorder the rows in order to handle the right hand side. More precisely, we order the strings in increasing position of the *last* binary entry. The obtained structure corresponds to the right hand side of the instance handled in Lemma **??**. Again we form the sub-matrices $\overrightarrow{A}_1, \overrightarrow{A}_2, \ldots, \overrightarrow{A}_{k'}$ analogous to the left hand side and guess the selected strings of all matrices simultaneously, giving a combined DP cell for all matrices on the right hand side.

A DP cell for the left hand side is compatible to a DP cell on the right hand side, if there is no string assigned to σ on the one side and to σ' on the other side. The algorithm keeps the solution with fewest errors over all combinations of a left DP cell with a compatible right DP cell.

Lemma 8. *The algorithm above is a QPTAS for the rooted case of* GAPLESS-MEC.

Proof. To analyze the running time, we observe that k and k' are at most $\log_{1/\varepsilon}(n)$ since for each i we assume that $|\tau(\overleftarrow{A}_{i+1})| = \varepsilon|\tau(\overleftarrow{A}_i)|$ and $|\tau'(\overleftarrow{A}'_{i+1})| = \varepsilon|\tau'(\overleftarrow{A}'_i)|$. The number of instances $\overrightarrow{A}_i$ and $\overrightarrow{A}'_i$ are also at most $\log_{1/\varepsilon}(n)$ each, for the same reason.

We thus obtain super-cells that are combined of logarithmically many sub-cells with polynomial complexity. We obtain an overall super-cell which is a quadruple $(\overleftarrow{\zeta}, \overleftarrow{\zeta}', \overrightarrow{\zeta}, \overrightarrow{\zeta}')$, and we have to distinguish $\left(n^{O(1)}\right)^{4\log_{1/\varepsilon}(n)} = n^{O(\log n)}$ different cells, which is quasi-polynomial[3].

We now analyze the performance guarantee. For each column j, we obtain the values σ_j and σ'_j in almost the same way as we do in Lemma **??**, but with the difference that we require consistency with all other rows sampled. For an optimal solution (τ, τ'), it is sufficient to only consider choices of rows such that all

[3]We assume that n and m are polynomially related. This is justified because there are $n \cdot m$ entries of M and therefore measuring in m instead of n would also give a quasi-polynomial complexity.

rows selected for σ are in $\tau(M)$ and all rows selected for σ' are in $\tau'(M)$. Such a selection of rows ensures consistency. Note that we could apply the proof of Lemma **??** from the root to the left hand side and to the right hand side independently, if we knew $\tau(M)$ and $\tau'(M)$, just by avoiding wrong assignments. The simultaneous selection of all relevant rows ensures that we consider at least one selection of rows that satisfies these strong conditions. This solution is a $(1+\varepsilon)$ approximation by the proof of Lemma **??**, and our DP computes a solution of at least the same quality since we consider the overall number of errors with respect to all sampled rows. □

4.1 Length classes.

We next show how to use Lemma **??** to handle length classes of strings. To this end, let us assume w.l.o.g. that m (i.e., the number of columns in M) is a power of 2. Then for each $i \geq 0$, the ith length class L_i is the set of all strings of length ℓ with $\ell \in (m/2^{i+1}, m/2^i]$. We observe the following known property of length classes.

Lemma 9. *For each $i \geq 0$ there is a set $q_i = \{q_{i,1}, q_{i,2}, \dots\}$ of columns such that (a) each string in L_i crosses at least one column from q_i and (b) no string from L_i crosses more than two columns from q_i. Furthermore, we can choose the sets such that $q_i \subseteq q_{i+1}$.*

Proof. At level i, for each k with $1 \leq k \leq 2^{i+1}$ we select the column with index $k \cdot m/2^{i+1}$. Observe that the distance between two consecutive columns from q_i is $m/2^{i+1}$, which matches the shortest length of strings in L_i: if a minimal string starts right after a column of q_i, its last entry will cross the next column of q_i. Since strings do not start before column 1 and column m is contained in each q_i, claim (a) follows. To see (b), observe that a maximum length string of L_i is at most $m/2^i$. Let j be an index. The distances from $q_{i,j}$ to the column right before $q_{i,j+1}$ and from $q_{i,j+1}$ to right before q_{i+2} are exactly $m/2^{i+1}$. If the string starts directly at a column $q_{i,j}$ from q_i, it would cross column $q_{i,j+1}$ and end right before column $q_{i,j+2}$ as shown in Figure **??**(b).

The last claimed property follows directly from the construction of the sets q_i as shown in Figure **??**(b). □

For each i, we now separate L_i into two sub-instances. One sub-instance L'_i is formed by those rows from L_i that only cross one column of q_i and the second sub-instance L''_i is formed by those rows that cross exactly two columns of L_i.

We first show that we can handle each class separately.

Lemma 10. *There is a QPTAS for* GAPLESS-MEC *if all strings are in the same class L'_i or L''_i.*

Proof. We first note that by skipping all $q_{i,j}$ with even j, the strings in L''_i cross exactly one column of the set. It is therefore sufficient to handle L'_i.

For each column $q_{i,j}$, we create a set of DP cells that stores information about a center region and about non-domination intervals, exactly as in the proof of Theorem **??**. We can do this by using different row orderings on the left hand side and on the right hand side of $q_{i,j}$, as we did in the proof of Lemma **??**. For each $q_{i,j}$, let ξ_j be the set of super-cells from Lemma **??** for $q_{i,j}$, but with the additional center and non-domination information. We then design a DP that moves from left to right through the columns in q_i. The DP and its analysis now follow from the proof of Theorem **??**, but we consider the left hand side and right hand side of each cell from ξ_j simultaneously. The analysis is the same as in Lemma **??**. □

Combining the two sub-classes gives a QPTAS for an entire length class.

Lemma 11. *There is a QPTAS for* GAPLESS-MEC *if all strings are in the same class L_i.*

Proof. To combine the PTAS for L'_i and L''_i, we proceed from left to right. For each index j let ξ'_j be the sets of DP cells for L'_i and for the odd indices j let ξ''_j be the set of cells for L''_i. For a column c before $q_{i,1}$, each pair of cells $(C, C') \in \xi'_1 \times \xi''_1$) determines two subproblems (sets of boxes) that are used for computing the two separate solutions. Let $\mathcal{B}(C, C', z)$ be the set of all boxes (sets of rows) for σ considered in the sub-cells of (C, C') at column z and let $\mathcal{B}'(C, C', z)$ be the set of all boxes (sets of rows) for σ' considered in the sub-cells of (C, C') at column z. We can interpret Lemma **??** such that there is a (C, C') for which the chosen rows determine the solutions of the subproblems for c entirely.

We now extend the DP as follows. We compose solution from left to right, starting with the prefix of (σ, σ') before $q_{i,1}$ and then step by step filling the intervals between $q_{i,j}$ and $q_{i,j+1}$ for $j \geq 1$. The starting interval can be seen as the interval between a dummy-column q_0 and q_1. For some j, let us analyze its interval. If j is odd, we simultaneously consider the cells $\xi'_j, \xi'_{j+1}, \xi''_j, \xi''_{j+2}$. Otherwise, we simultaneously consider the cells $\xi'_j, \xi'_{j+1}, \xi''_{j-1}, \xi''_{j+1}$.

For each column c with index ℓ in the interval, in both cases the values of the DP cells reveal all $\mathrm{SWC}_{\varepsilon^3}$ instances at c that we would have to solve in order to obtain solutions for L'_i and L''_i separately. Instead of solving these instances separately, we solve them simultaneously.

Let $\hat{c}_1, \hat{c}'_1$ and $\hat{c}_2, \hat{c}'_2$ be the vectors from the proof of Lemma **??** that contain those rows from $\tau(M)$ that belong to L'_i resp. L''_i and are contained in sets from $\mathcal{B}(C, C', \ell)$ resp. $\mathcal{B}'(C, C', \ell)$ with only binary entries. Then the entry σ_ℓ is the weighted majority of entries. The weighted majority is defined as in the algorithm $\mathrm{SWC}_{\varepsilon^3}$, but for both $\hat{c}_1$ and $\hat{c}_2$, i.e., we scale the weight of each entry from $\hat{c}_1$ by $|\hat{c}_1|/(|\hat{c}_1| + |\hat{c}_2|)$ and each entry from $\hat{c}_2$ by $|\hat{c}_2|/(|\hat{c}_1| + |\hat{c}_2|)$. Recall that in the proof of Lemma **??** we argued that for each single column the expected number of errors is at most a factor $(1 + \varepsilon)$ larger than for (τ, τ'). We can apply exactly the same argument here, with the only difference that we consider four different density classes instead of two. Again, for τ' the arguments are analogous.

Since we consider all cells for the entire interval simultaneously, one of the choices is at least as good as sampling uniformly at random with knowledge of $\tau(M)$ and $\tau'(M)$. We therefore obtain a solution for the interval with at most a $(1 + O(\varepsilon))$ factor of errors compared to (τ, τ')

Finally we have to join the results that we obtain for the intervals. Observe that for each pair of cells $(C''_j, C''_{j+2}) \in \xi''_j \times \xi''_{j+2}$ there are two consecutive pairs of cells $(C'_j, C'_{j+1}) \in \xi'_j \times \xi'_{j+1}$ and $(C'_{j+1}, C'_{j+2}) \in \xi'_{j+1} \times \xi'_{j+2}$. For a quadruple of cells $(C'_j, C''_j, C'_{j+1}, C''_{j+2})$ we consider each quadruple on the left hand side ending with the matching cells C'_j, C''_j. Among these, we take the one with fewest errors. To obtain the value of the new quadruple, we add the errors in the interval $(q_{i,j}q_{i,j+1}]$ to the value of the selected predecessor quadruple. To compute the value $(C'_{j+1}, C''_j, C'_{j+2}, C''_{j+2})$, we consider all cells $(C'_j, C''_j, C'_{j+1}, C''_{j+2})$, i.e., the cells that have the same $C''_j, C''_{j+1}, C'_{j+1}$ for all choices of C'_j. We add the errors between $q_{i,j+1}$ and $q_{i,j+2}$ to the smallest value found among the predecessors.

The approximation ratio follows from along the lines of the analysis in Lemma **??** and the quasi-polynomial running time from that we only consider constantly many super-cells simultaneously.

□

4.2 The general QPTAS.

Finally we combine our insights to an algorithm for general instances shown in Figure **??**.

We partition the rows into their at most $\log_2(m)$ length classes L_i. The main idea is that for each column j, we only have to consider those quadruple of super-cells from Lemma **??** that cross j.

We therefore consider at most $O(\log(n))$ quadruples of super-cells simultaneously. Then the overall complexity of a joint cell is quasi-polynomial: the number of different cells is $\left(n^{O(\log n)}\right)^{O(\log n)} = n^{O(\log^2 n)}$. Let $Q_{i,j}$ be the set of quadruples of length class i crossing column j.

For each length class i, a quadruple $q \in Q_{i,j}$ can starting at j, cross j, or end in j. If j is the index of $q_{i,\ell}$, The quadruple q starts in j if it is formed by cells $(C'_\ell, C''_\ell, C'_{\ell+1}, C''_{\ell+2})$ (see proof of Lemma **??**). It ends

in j if it is formed by $(C'_{\ell-1}, C''_{\ell-2}, C'_\ell, C''_\ell)$. If j lies between $q_{i,\ell}$ and q_i, ℓ, j crosses those quadruples that contain C'_ℓ and $C'_{\ell+1}$. If non of the cases are true, we do not consider q in the cells for column j.

Let us consider a $\log(n)$ vector of quadruples v. We require that if for some i, the quadruple $q \in Q_{i,j}$ ends at j, then the same also holds for all quadruples of shorter length classes (with index larger than i). This also implies that if for some i, the quadruple of length class i starts at j, then the same also holds for all quadruples of shorter length classes. In particular, in order to be able to combine neighboring vectors of quadruples, we do not allow to mix starting and ending quadruples. Let ϕ be the set of all $\log(n)$ vectors of tuples as described above (with one tuple of each length class).

The DP for general instances follows the ideas of Lemma **??**. We move from left to right column by column. For column j, let us consider a vector $v \in \phi$. We distinguish whether v has starting or ending quadruples. (One of the two cases must apply due to the shortest length class.) For a $v \in \phi$ with starting quadruples, let d be the smallest number such that there is a quadruple of length class d starting at j. To compute v we consider all $v' \in \phi$ with the following properties. (a) v' has the same quadruples for all length classes $d' < d$ and (b) for $d' \geq d$, the right hand sides of the quadruples of length class d' in v' match the left hand sides of the quadruples of v (see proof of Lemma **??**). Then the value of v is the minimum value over all v'.

For a $v \in \phi$ with ending quadruples, let d be the smallest number such that there is a quadruple of length class d ending at $j-1$. (In the very first column of the instance, we do not need this value.) To compute v we consider all $v' \in \phi$ with the following properties. (a) v' has the same quadruples for all length classes $d' < d$ and (b) for $d' \geq d$, the right hand sides of the quadruples of length class d' in v' match the left hand sides of the quadruples of v in column $j-1$. Then the value of v is the sum of the minimum value over all such v' and the number of errors in column j obtained by applying $\mathrm{SWC}_{\varepsilon^3}$ exactly as in the proof of Lemma **??**. Observe that in Lemma **??** we combined two pairs of super-cells that we can see as two length classes L' and L''. The only difference is that here we combine $\log(n)$ quadruples of super-cells.

Theorem 3. *The above algorithm is a QPTAS for* GAPLESS-MEC.

Acknowledgment

We would like to thank Tobias Marschall for helpful discussions.

Appendix

A Proof of Lemma ??

Proof. We argue that for each column, the expected number of errors is at most a factor $(1 + O(\varepsilon))$ larger than in an optimal solution. Then the claim follows from linearity of expectation and our discussion about derandomization.

We consider the jth column of G. Let $c := \tau(G)_{*,j}$, but without rows that have an entry "$-$" in column j. Let $p := |\{i : c_i = 0\}|/|c|$ be the fraction of zeros in c. By swapping the zeros and ones we can assume w.l.o.g. that $p \geq 1 - p$, i.e., $p \geq 1/2$. Our assumption implies $\tau_j = 0$ and the optimal solution has $(1 - p)|c|$ errors within c.

The general idea of the proof is as follows. Suppose we would select exactly one row from $\tau(G)$ uniformly at random. Then with probability p, the algorithm has $(1 - p)|c|$ errors in c and with probability $(1 - p)$ the number of errors is $p|c|$. Therefore the expected number of errors is $(p(1 - p) + (1 - p)p)|c| = 2p(1 - p)|c|$. We obtain the approximation ratio $2p(1 - p)|c|/((1 - p)|c|) = 2p$.

We will see that the approximation ratio improves with choosing several rows instead of a single one. Additionally, we have to handle the circumstance that we only sample from R and ignore R_C.

There is a further issue regarding $\mathcal{R}$. Let s be the smallest index such that R_s and c intersect, i.e., R_s is the first set with binary entries in column j. Then rows sampled for R_s may be located outside of c at positions with wildcards in column j. We avoid the complications caused by the wildcards by only considering classes R_i for $i > s$.

To summarize, c has at least εr entries and we ignore at most $2\varepsilon^2$ of these due to R_C and R_s. For each $i > s$, we sample $1/\varepsilon^3$ rows from R_i. Let c' be c restricted to $\bigcup_{s<i\leq\ell} R_i$ and let c'' be c restricted to $\bigcup_{i>\ell} R_i$. For each i, let c_i be the fraction of zeros of c in R_i. Let $\hat{c}$ be c without R_s and R_C and let $\bar{c}$ be the part of c in $R_C \cup R_s$. For each i, we define p_i to be the fraction of zeros c_i.

We define a random variable $X_{i,k}$ for each $s < i \leq 2\ell$ and $1 \leq k \leq 1/\varepsilon^3$. For each i, k, we pick an entry from c_i uniformly at random. Then $X_{i,k}$ is the value of the picked entry. For all i, k, $E[X_{i,k}] = 1 - p_i$. Observe that the $X_{i,k}$ are independent Poisson trials. Let $X' := \sum_{s<i\leq\ell, 1\leq k\leq 1/\varepsilon^3} X_{i,k}$ and $X'' := \sum_{i'>\ell, 1\leq k\leq 1/\varepsilon^3} X_{i,k}$. We want to use Chernoff bounds to control the probability to take the wrong decision. It is sufficient to consider X' with $s = \ell - 1$, since in all other cases the probabilities are amplified more. Let $\mu' := E[X']$. We analyze the ranges of μ' separately.

Case 1: Let us assume that $\mu' \in [0, 1/(2\mathrm{e}\varepsilon^3)]$. We define $\delta' := 1/(2\mu'\varepsilon^3) - 1$. Using a multiplicative Chernoff bound (cf. [**?**]), we obtain

$$\Pr(X' \geq 1/(2\varepsilon^3)) < \Big(\frac{\mathrm{e}^{\delta'}}{(1+\delta')^{(1+\delta')}}\Big)^{\mu'} = \Big(\frac{1}{1+\delta'}\Big)^{\mu'}\Big(\frac{\mathrm{e}}{1+\delta'}\Big)^{\mu'\delta'} \tag{1}$$

$$= (2\mu'\varepsilon^3)^{\mu'}(\mathrm{e}\cdot 2\mu'\varepsilon^3)^{(1/(2\varepsilon^3)-\mu')} \tag{2}$$

Note that both terms of (**??**) are numbers between zero and one. If $\mu' < 1/\varepsilon$, the right term is smaller than $\varepsilon^4\mu'$. Otherwise the left term is smaller than $\varepsilon^4\mu'$

The range of μ' implies that the majority of entries in $\hat{c}'$ is zero. Recall that $\hat{c}'$ has an $\varepsilon^3\mu'$ fraction of zeros. The expected number of errors done by the algorithm is therefore at most $(1 - \varepsilon^4 \cdot \mu') \cdot (\varepsilon^3\mu') + \varepsilon^4 \cdot \mu' \cdot (1 - \varepsilon^3\mu') = (1 + \varepsilon)\varepsilon^3\mu'$.

Case 2: Let us assume that $\mu' \in (1/(2\mathrm{e}\varepsilon^3), 1/(2\varepsilon^3) - 1/\varepsilon^2]$. We use Hoeffding's inequality [**?**] to analyze the range. To this end, we scale X' and obtain $\bar{X}' := \varepsilon^3 X'$, which has values between zero and one. Then

$$\Pr(\bar{X}' - E[\bar{X}'] \geq \varepsilon) \leq \mathrm{e}^{-2\varepsilon^2/\varepsilon^3} = \mathrm{e}^{-2/\varepsilon}\,.$$

Since for sufficiently small ε, $\mathrm{e}^{-2/\varepsilon} < \varepsilon/(2\mathrm{e}) \leq \varepsilon^4\mu'$, again we obtain a $(1+\varepsilon)$-approximation in expectation.

All other ranges now follow immediately: For $\mu' \in (1/(2\varepsilon^3) - 1/\varepsilon^2, 1/(2\varepsilon^3)]$ every solution is a $(1+O(\varepsilon))$-approximation and for larger μ' the majority of entries in $\hat{c}'$ is one. The analysis is analogous.

In order to combine X' and X'', we introduce a bias for X' such that we count rows i for $s < i \leq \ell$ with a factor $(1-\varepsilon)/(\varepsilon-\varepsilon^2)$. Then

$$\bar{X} := \frac{\bar{X}' \cdot (\ell - s)(1-\varepsilon)/(\varepsilon-\varepsilon^2) + \bar{X}'' \cdot \ell}{(\ell - s)(1-\varepsilon)/(\varepsilon-\varepsilon^2) + \ell}\,.$$

Then, using the union bound, setting $\sigma_j = 0$ for $\bar{X} < 1/2$ and $\sigma_j = 1$ otherwise gives an expected $1 + O(\varepsilon)$ approximation within $\hat{c}$. Errors in $\bar{c}$ are either also errors in an optimal solution, or they contribute at most a factor $O(\varepsilon)$ to the total number of errors. Thus overall we obtain an approximation ratio $1 + O(\varepsilon)$ within c. The algorithm $\mathrm{SWC}_{\varepsilon^3}$ has at most the same approximation ratio, since the only difference is that we do not fix the $X_{i,k}$ to be zero or one. Thus the random process used by the algorithm can only have a lower variance.

This finishes our analysis for $\tau(G)_{*,j}$. For $\tau'(G)_{*,j}$, the proof is analogous.

To finish the proof we still have to argue that we may assign the r rows with maximal values d_i to σ and the remaining r' rows to σ'. The reason is that at all times we compare against the assignment of (τ, τ'), which is a stronger result then the mere approximation ratio. Since $|\tau(G)| = r$ and the choice of d_i gives the assignment with fewest errors, the claim follows. □

Again, we introduced a small but easy to handle imprecision due to the assumption that we can choose exactly the same number of strings from each range.

B Figures

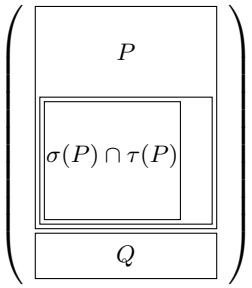

Figure 1: Separation of M into blocks P and Q, where the rows of M are reordered for ease of presentation. The Block on the left hand side shows the rows $\sigma(P) \cap \tau(P)$ within columns I (moved to the left).

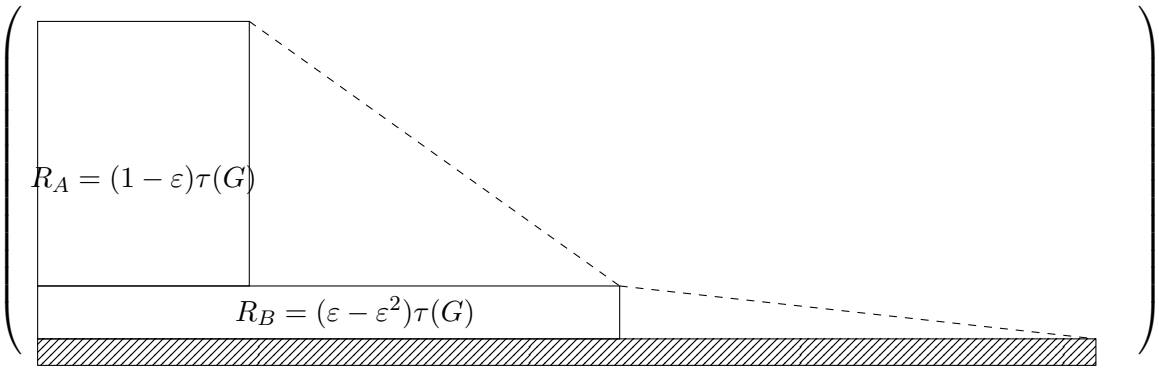

Figure 2: Row sets on an instance G . The bottom stripe depicts a block R_C of G. All the blocks have binary values without wildcards.

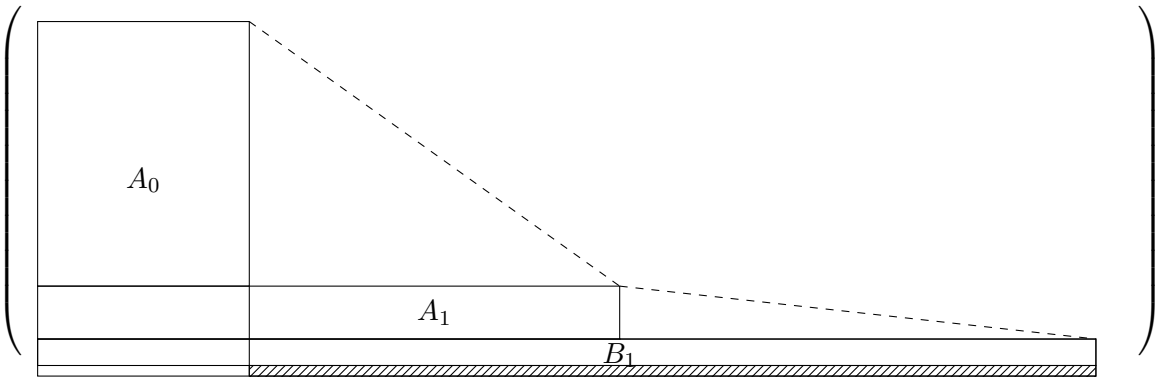

Figure 3: Blocks A_i, B_i on an instance G for the i^{th} iteration of DP in Lemma **??**. The bottom stripe depicts a block C_1 of G.

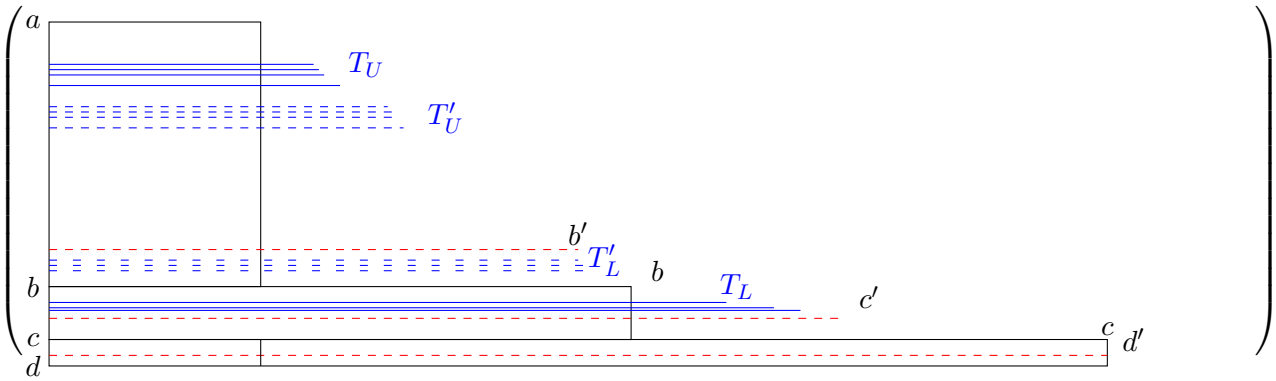

Figure 4: Example of a child cell in the DP for the second string. The cell starting at b' (drawn in red) is a child of the cell staring at a (drawn in blue), because $b' > b$ and $T_U \cup T_L$ is disjoint from $T'_U \cup T'_L$.

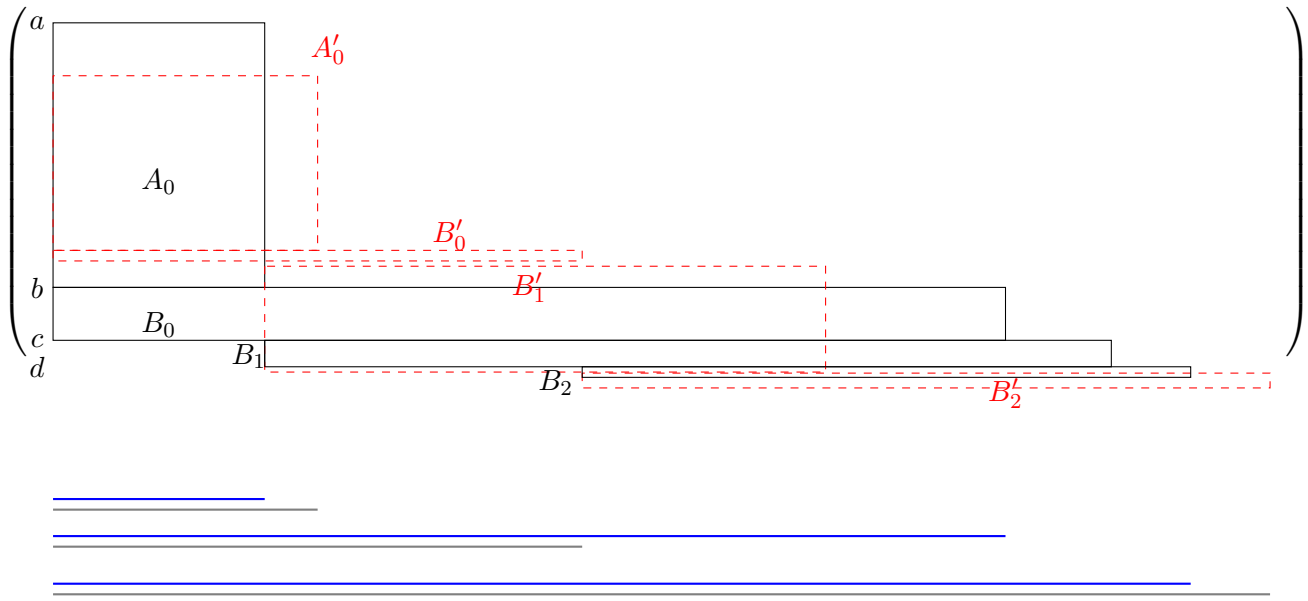

Figure 5: Blocks of an instance M in the DP for the second string. B'_0 is a child of A_0, B'_1 is a child of A_0, B'_1 is a child of A_0. The blue and gray lines represents σ and σ' respectively from first two iterations of DP. The sketch shows the switch example in the second iteration.

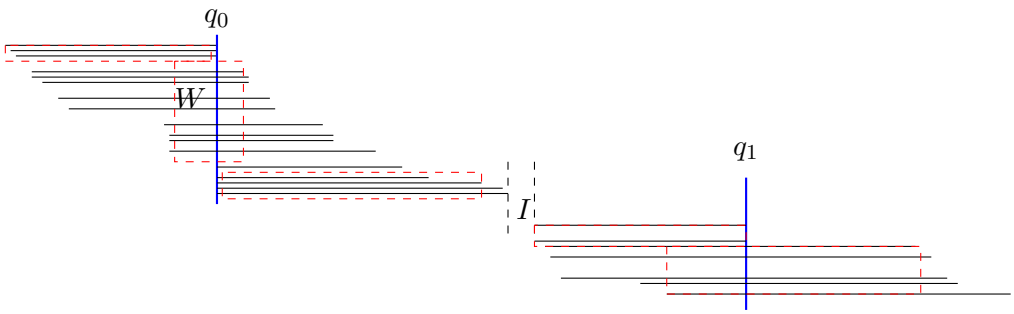

Figure 6: Blocks represented by ranges shown in red on an instance M and the blue lines are the columns, I and C shows the empty interval and central region respectively.

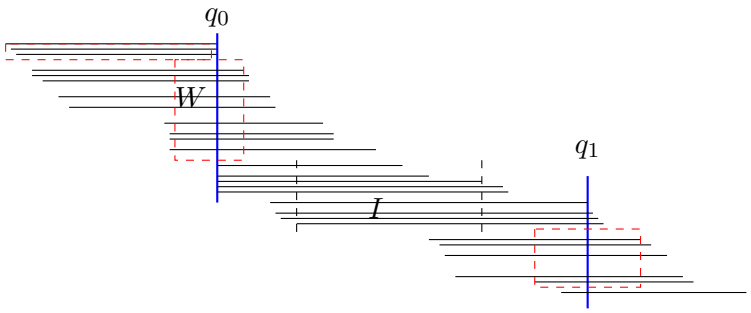

Figure 7: This sketch shows non-dominance example in region I.

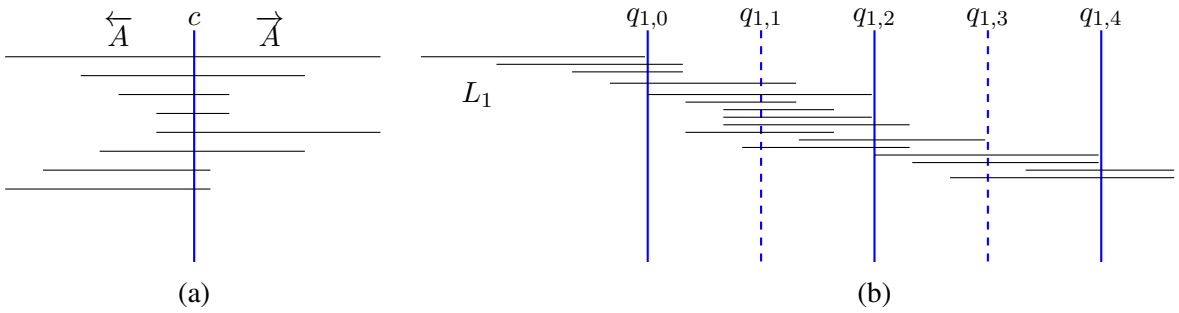

Figure 8: (a) Sub-matrices for rooted GAPLESS-MEC. (b) For a single-length-class instance, the sketch shows the strings crossing each column either exactly once or exactly twice.

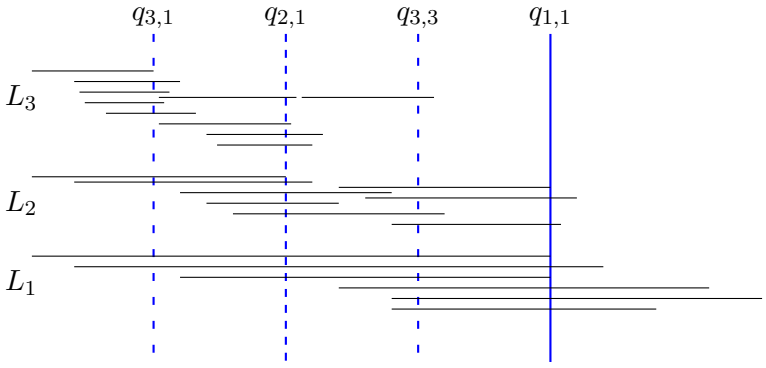

Figure 9: Different length classes L_1 with corresponding column $q_{1,1}$, L_2 with corresponding columns $q_{2,1}, q_{2,2} = q_{1,1}$, and L_3 with corresponding columns $q_{3,1}, q_{3,2} = q_{2,1}, q_{3,3}, q_{3,4} = q_{1,1}$.