Order-Preserving Pattern Matching Indeterminate Strings

Rui Henriques
INESC-ID and Instituto Superior Técnico, Universidade de Lisboa, Portugal
rmch@tecnico.ulisboa.pt (correspondence)

Alexandre P. Francisco
INESC-ID and Instituto Superior Técnico, Universidade de Lisboa, Portugal

Luís M. S. Russo
INESC-ID and Instituto Superior Técnico, Universidade de Lisboa, Portugal

Hideo Bannai
Department of Computer Science, Kyushu University, Japan

Abstract

Given an indeterminate string pattern p and an indeterminate string text t, the problem of order-preserving pattern matching with character uncertainties (μOPPM) is to find all substrings of t that satisfy one of the possible orderings defined by p. When the text and pattern are determinate strings, we are in the presence of the well-studied exact order-preserving pattern matching (OPPM) problem with diverse applications on time series analysis. Despite its relevance, the exact OPPM problem suffers from two major drawbacks: 1) the inability to deal with indetermination in the text, thus preventing the analysis of noisy time series; and 2) the inability to deal with indetermination in the pattern, thus imposing the strict satisfaction of the orders among all pattern positions.

In this paper, we provide the first polynomial algorithms to answer the μOPPM problem when: 1) indetermination is observed on the pattern or text; and 2) indetermination is observed on both the pattern and the text and given by uncertainties between pairs of characters. First, given two strings with the same length m and $O(r)$ uncertain characters per string position, we show that the μOPPM problem can be solved in $O(mr \lg r)$ time when one string is indeterminate and $r \in \mathbb{N}^+$ and in $O(m^2)$ time when both strings are indeterminate and r=2. Second, given an indeterminate text string of length n, we show that μOPPM can be efficiently solved in polynomial time and linear space.

2012 ACM Subject Classification Theory of computation → Pattern matching

Keywords and phrases Order-preserving pattern matching, Indeterminate string analysis, Generic pattern matching, Satisfiability

Digital Object Identifier 10.4230/LIPIcs.CPM.2018.2

Funding This work was developed in the context of a secondment granted by the BIRDS MASC RISE project funded in part by EU H2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement no.690941. This work was further supported by national funds through Fundação para a Ciência e Tecnologia (FCT) with reference UID/CEC/50021/2013.

1 Introduction

Given a pattern string p and a text string t, the exact order preserving pattern matching (OPPM) problem is to find all substrings of t with the same relative orders as p. The problem is applicable to strings with characters drawn from numeric or ordinal alphabets.

Illustrating, given p=(1,5,3,3) and t=(5,1,4,2,2,5,2,4), substring $t[1..4]$=(1,4,2,2) is reported since it satisfies the character orders in p, $p[0] \leq p[2] = p[3] \leq p[1]$. Despite its relevance, the OPPM problem has limited potential since it prevents the specification of errors, uncertainties or don't care characters within the text.

Indeterminate strings allow uncertainties between two or more characters per position. Given indeterminate strings p and t, the problem of order preserving pattern matching uncertain text (μOPPM) is to find all substrings of t with an assignment of values that satisfy the orders defined by p. For instance, let p=(1,2|5,3,3) and t=(5,0,1,2|1,2,5,2|3,3|4). The substrings $t[1..4]$ and $t[4..7]$ are reported since there is an assignment of values that preserve either $p[0]<p[1]<p[2]=p[3]$ or $p[0]<p[2]=p[3]<p[1]$ orderings: respectively $t[1..4]$=(0,1,2,2) and $t[4..7]$=(2,5,3,3).

Order-preserving pattern matching captures the structural isomorphism of strings, therefore having a wide-range of relevant applications in the analysis of financial times series, musical sheets, physiological signals and biological sequences [32, 39, 36]. Uncertainties often occur across these domains. In this context, although the OPPM problem is already a relaxation of the traditional pattern matching problem, the need to further handle localized errors is essential to deal with noisy strings [33]. For instance, given the stochasticity of gene regulation (or markets), the discovery of order-preserving patterns in gene expression (or financial) time series needs to account for uncertainties [35, 34]. Numerical indexes of amino-acids (representing physiochemical and biochemical properties) are subjected to errors difficulting the analysis of protein sequences [38]. Another example are ordinal strings obtained from the discretization of numerical strings, often having two uncertain characters in positions where the original values are near a discretization boundary [33].

Let m and n be the length of the pattern p and text t, respectively. The exact OPPM problem has a linear solution on the text length $O(n+m\lg m)$ based on the Knuth-Morris-Pratt algorithm [41, 39, 22]. Alternative algorithms for the OPPM problem have also been proposed [21, 12, 20]. Contrasting with the large attention given to the resolution of the OPPM problem, to our knowledge there are no polynomial-time algorithms to solve the μOPPM problem. Naive algorithms for μOPPM assess all possible pattern and text assignments, bounded by $O(nr^m)$ when considering up to r uncertain characters per position.

This work proposes the first polynomial algorithms able to answer the μOPPM problem. Accordingly, the contributions are organized as follows. First, we show that an indeterminate string of length m order-preserving matches a determinate string with the same length in $O(mr \lg r)$ time based on their monotonic properties. Second, we show that μOPPM two indeterminate strings with the same length and r=2 can be solved in $O(m^2)$ time by reducing μOPPM to a 2-satisfiability task. Third, given a text string of length n, we show that the μOPPM problem is in polynomial time and linear space, and efficiently solved using effective filtration procedures.

2 Background

Let Σ be a totally ordered alphabet and an element of Σ^* be a string. The length of a string w is denoted by $|w|$. The empty string ε is a string of length 0. For a string w=xyz, x, y and z are called a prefix, substring, and suffix of w, respectively. The i-th character of a string w is denoted by $w[i]$ for each $1 \leq i \leq |w|$. For a string w and integers $1 \leq i \leq j \leq |w|$, $w[i..j]$ denotes the substring of w from position i to position j. For convenience, let $w[i..j]=\varepsilon$ when $i>j$.

Given strings x and y with equal length m, y is said to order-preserving against x [41], denoted by $x \approx y$, if the orders between the characters of x and y are the same, i.e.

$x[i] \leq x[j] \Leftrightarrow y[i] \leq y[j]$ for any $1 \leq i,j \leq m$. A non-empty pattern string p is said to order-preserving match (*op-match* in short) a non-empty text string t iff there is a position i in t such that $p \approx t[i-|p|+1..i]$. The *order-preserving pattern matching* (OPPM) problem is to find all such text positions.

2.1 The Problem

Given a totally ordered alphabet Σ, an indeterminate string is a sequence of disjunctive sets of characters $x[1]x[2]..x[n]$ where $x[i] \subseteq \Sigma$. Each position is given by $x[i]{=}\sigma_1..\sigma_r$ where $r{\geq}1 \wedge \sigma_i{\in}\Sigma$.

Given an indeterminate string x, a *valid assignment* $\$x$ is a (determinate) string with a single character at position i, denoted $\$x[i]$, contained in the $x[i]$ set of characters, i.e. $\$x[1]{\in}x[1],..,\ \$x[m]{\in}x[m]$. For instance, the indeterminate string $(1|3,3|4,2|3,1|2)$ has 2^4 valid assignments. Given an indeterminate position $x[i] \subseteq \Sigma$, $\$x_j[i]$ is the j^{th} ordered value of $x[i]$ (e.g. $\$x_0[i]{=}1$ for $x[i]{=}1|2$). Given an indeterminate string x, let a *partially assigned* string $\S x$ be an indeterminate string with an arbitrary number of uncertain characters removed, i.e. $\S x[1]{\subseteq}x[1],..,\S x[m]{\subseteq}x[m]$.

Given a determinate string x of length m, an indeterminate string y of equal length is said to be *order-preserving* against x, identically denoted by $x \approx y$, if there is a valid assignment $\$y$ such that the relative orders of the characters in x and $\$y$ are the same, i.e. $x[i] \leq x[j] \Leftrightarrow \$y[i] \leq \$y[j]$ for any $1 \leq i,j \leq m$. Given two indeterminate strings x and y with length m, y preserves the orders of x, $x \approx y$, if exists $\$y$ in y that respects the orders of a valid assignment $\$x$ in x.

A non-empty indeterminate pattern string p is said to order-preserving match (*op-match* in short) a non-empty indeterminate text string t iff there is a position i in t such that $p \approx t[i\text{-}|p|{+}1..i]$. The problem of *order-preserving pattern matching with character uncertainties* (μOPPM) problem is to find all such text positions.

To understand the complexity of the μOPPM problem, let us look to its solution from a naive stance yet considering state-of-the-art OPPM principles. The algorithmic proposal by Kubica et al. [41] is still up to this date the one providing a lowest bound, $O(n{+}q)$, where $q{=}m$ for alphabets of size $m^{O(1)}$ ($q{=}m \lg m$ otherwise). Given a determinate string x of length m, an integer i ($1{\leq}i{<}m$) is said in the context of this work to be an *order-preserving border* of x if $x[1..i] \approx x[m\text{-}i{+}1..m]$. In this context, given a pattern string p, the orders between the characters of p are used to linearly infer the order borders. The order borders can then be used within the Knuth-Morris-Pratt algorithm to find op-matches against a text string t in linear time [41].

Given a determinate string p of length m and an indeterminate string t of length n, the previous approach is a direct candidate to the μOPPM problem by decomposing t in all its possible assignments, $O(r^n)$. Since determinate assignments to t are only relevant in the context of m-length windows, this approach can be improved to guarantee a maximum of $O(r^m)$ assignments at each text position. Despite its simplicity, this solution is bounded by $O(nr^m)$. This complexity is further increased when indetermination is also considered in the pattern, stressing the need for more efficient alternatives.

2.2 Related work

The exact OPPM problem is well-studied in literature. Kubica et al. [41], Kim et al. [39] and Cho et al. [22] presented linear time solutions on the text length by respectively combining order-borders, rank-based prefixes and grammars with the Knuth–Morris–Pratt (KMP) algorithm [40]. Cho et al. [21], Belazzougui et al. [12], and Chhabra et al. [20] presented $O(nm)$ algorithms that show a sublinear average complexity by either combining bad character heuristics with the Boyer–Moore algorithm [13] or applying filtration strategies. Recently, Chhabra et al. [18] proposed further principles to solve OPPM using word-size packed string matching instructions to enhance efficiency.

In the context of numeric strings, multiple relaxations to the exact pattern matching problem have been pursued to guarantee that approximate matches are retrieved. In norm matching [7, 44, 1, 47], matches between numeric strings occur if a given distance threshold $f(x,y) \leq \theta$ is satisfied. In (δ,γ)-matching [14, 26, 24, 23, 42, 43, 45], strings are matched if the maximum difference of the corresponding characters is at most δ and the sum of differences is at most γ.

In the context of nominal strings, variants of the pattern matching task have also been extensively studied to allow for don't care symbols in the pattern [37, 25, 9], transposition-invariant [42], parameterized matching [11, 6], less than matching [5], swapped matching [2, 46], gaps [15, 16, 31], overlap matching [4], and function matching [3, 8].

Despite the relevance of the aforementioned contributions to answer the exact order-preserving pattern matching and generic pattern matching, they cannot be straightforwardly extended to efficiently answer the μOPPM problem.

3 Polynomial time μOPPM for equal length pattern and text

Section 3.1 introduces the first efficient algorithm to solve the μOPPM problem when one string is indeterminate ($r \in \mathbb{N}^+$). *Section 3.2* shows the existence of a polynomial solution when both strings are indeterminate and uncertainties are observed between pairs of characters (r=2). Based on the reducibility of the graph coloring problem to the formulations proposed in *Section 3.2*, we hypothesize that op-matching indeterminate strings with an arbitrary number of uncertain characters per position ($r \in \mathbb{N}^+$) is in class **NPC**. The proof of this intuition is, nevertheless, considered out of the scope, being regarded as future work.

3.1 $O(mr \lg r)$ time μOPPM when one string is indeterminate

Given a determinate string x of length m, there is a well-defined permutation of positions, π, that specifies a non-monotonic ascending order of characters in x. For instance, given x=(1,4,3,1), then $x[0]$=$x[3]$<$x[2]$<$x[1]$ and π=(0,3,2,1). Given a determinate string y with the same length, y op-matches x if it y satisfies the same m-1 orders. For instance, given x=(1,4,3,1) and y=(2,5,4,3), x orders are not preserved in y since $y[0] \neq y[3] < y[2] < y[1]$.

The monotonic properties can be used to answer μOPPM when one string is indeterminate. Given an indeterminate string y, let x_π and y_π be the permuted strings in accordance with π orders in x. To handle equality constraints, positions in y_π with identical characters in x_π can be intersected, producing a new string y'_π with s length ($s \leq m$). Illustrating, given x=(4,1,4,2) and y=$(2|7, 2, 7|8, 1|4|8)$, then π=(1,3,0,2), x_π=(1,2,4,4), y_π=$(2, 8|4|1, 7|2, 8|7)$ and y'_π=$(y_\pi[0], y_\pi[1], y_\pi[2] \cap y_\pi[3])$=$(2, 8|4|1, 7)$. To handle monotonic inequalities, $y'_\pi[i]$ characters can be concatenated in descending order to compose z=$y'_\pi[0]y'_\pi[1]..y'_\pi[s]$ and the orders between x and y verified by testing if the longest increasing subsequence (LIS) [29] of z has s length. In the given example, z=$(2, 8, 4, 1, 7)$, and the LIS of z=$(\mathbf{2}, 8, \mathbf{4}, 1, \mathbf{7})$ is w=(2,4,7). Since $|w|$=$|y'_\pi|$=3, y op-matches x.

▶ **Theorem 1.** *Given a determinate string x and an indeterminate string y, let x_π and y_π be the sorted strings in accordance with π order of characters in x. Let the positions with equal*

Algorithm 1: $O(mr \lg r)$ μOPPM algorithm with one indeterminate string

```
Input: determinate $x$, indeterminate $y$ ($|x|$=$|y|$=$m$)
$\pi \leftarrow$ sortedIndexes($x$);                    // $O(m)$ if $|\Sigma| = m^{O(1)}$ ($O(m \lg m)$ otherwise)
$x_\pi \leftarrow$ permute($x$,$\pi$), $y_\pi \leftarrow$ permute($y$,$\pi$);      // $O(m + mr)$
$j \leftarrow 0$; $y'_\pi[0] \leftarrow \{y_\pi[0]\}$;
foreach $i \in$ 1..m-1 do                                // $O(mr \lg r)$
    if $x_\pi[i] = x_\pi[i$-1$]$ then $y'_\pi[j] \leftarrow y'_\pi[j] \cap \{y_\pi[i]\}$; // $O(r \lg r)$
    else $j \leftarrow j$+1; $y'_\pi[j] \leftarrow \{y_\pi[i]\}$;
$s \leftarrow |y'_\pi|$, nextMin $\leftarrow -\infty$;
foreach $i \in$ 0..$s$-1 do                              // $O(mr)$
    nextMin $\leftarrow$ min$\{a \mid a \in y'_\pi[i], a$>nextMin$\}$;    // $O(r)$
    if $\nexists$ nextMin then return false;
return true;
```

characters in x_π be intersected in y_π to produce a new indeterminate string y'_π. Consider z_i to be a string with $y'_\pi[i]$ characters in descending order and z=$z_1z_2..z_m$, then:

$$|w| = |y'_\pi| \Leftrightarrow y \approx x \quad \text{where} \;\; w = \text{longest increasing subsequence in } z \tag{1}$$

Proof. ($\Rightarrow$) If the length of the longest increasing subsequence (LIS), $|w|$, equals the number of monotonic relations in x, $|y'_\pi|$, then $y \approx x$. By sorting characters in descending order per position, we guarantee that at most one character per position in y'_π appears in the LIS (respecting monotonic orders in x given y'_π properties). By intersecting characters in positions of y with identical characters in x, we guarantee the eligibility of characters satisfying equality orders in x, otherwise empty positions in y'_π are observed and the LIS length is less than $|y'_\pi|$. ($\Leftarrow$) If $|w|$<$|y'_\pi|$, there is no assignment in y that op-matches x due to one of two reasons: 1) there are empty positions in y'_π due to the inability to satisfy equalities in x, or 2) it is not possible to find a monotonically increasing assignment to y'_π and, given the properties of y'_π, y_π cannot preserve the orders of x_π. ◀

Solving the LIS task on a string of size n is $O(n \lg n)$ [29] where n=$|z|$=$O(rm)$. In addition, set intersection operations are performed $O(m)$ times on sets with $O(r)$ size, which can be accomplished in $O(rm \lg r)$ time. As a result, the μOPPM problem with one indeterminate string can be solved in $O(rm \lg(rm))$.

Given the fact that the candidate string for the LIS task has properties of interest, we can improve the complexity of this calculus (*Theorem* 2) in accordance with Algorithm 1.

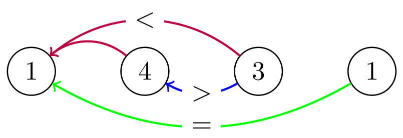
▶ **Theorem 2.** *μOPPM two strings of length m, one being indeterminate, is in $O(mr \lg r)$ time, where $r \in \mathbb{N}+$.*

Proof. In accordance with Algorithm 1, μOPPM is bounded by the verification of equalities, $O(mr \lg r)$ [27]. Testing inequalities after set intersections can be linearly performed on the size of y, $O(mr)$ time, improving the $O(mr \lg(mr))$ bound given by the LIS calculus. ◀

The analysis of Algorthim 1 further reveals that the μOPPM problem with one indeterminate string requires linear space in the text length, $O(mr)$.

3.2 $O(m^2)$ time μOPPM (r=2) with indeterminate pattern and text

As indetermination in real-world strings is typically observed between pairs of characters [33], a key question is whether μOPPM on two indeterminate strings is in class **P** when r=2. To explore this possibility, new concepts need to be introduced. In OPPM research, character orders in a string of length m can be decomposed in 3 sequences with m unit sets:



Pattern	1	4	3	1
Leq$[i]$	$\emptyset$	$\emptyset$	$\emptyset$	$\{0\}$
Ordered indexes (asc)	0	3	2	1
Lmax$[i]$ (nearest asc smaller not in *Leq*$[i]$)	$\emptyset$	$\{0\}$	$\{0\}$	$\emptyset$
Ordered indexes (desc)	2	0	1	3
Lmin$[i]$ (nearest desc smaller not in *Leq*$[i]$)	$\emptyset$	$\emptyset$	$\{1\}$	$\emptyset$

Figure 1 Orders identified for p=(1,4,3,1) in accordance with Kubica et al. [41].

▶ **Definition 3.** For i=0,...,$m-1$:

- $Leq_x[i]$=$\{\max\{\ k : k<i,\ x[i]=x[k]\ \}\}$ ($\emptyset$ if there is no eligible k)
- $Lmax_x[i]$=$\{\max\{\text{argmax}_k\{\ x[k] : k<i,\ x[i]>x[k]\ \}\}\}$ ($\emptyset$ if there is no eligible k)
- $Lmin_x[i]$=$\{\max\{\text{argmin}_k\{\ x[k] : k<i,\ x[i]<x[k]\ \}\}\}$ ($\emptyset$ if there is no eligible k)

Leq, *Lmax* and *Lmin* capture =, > and < relationships between each character $x[i]$ in x and the closest preceding character $x[k]$. These orders can be inferred in linear time for alphabets of size $m^{O(1)}$ and in $O(m \lg m)$ time for other alphabets by answering the "all nearest smaller values" task on the sorted indexes [41]. Figure 1 depicts *Leq*, *Lmax* and *Lmin* for x=(1,4,3,1). Given determinate strings x and y, A=$Leq_x[t$+1], B=$Lmax_x[t$+1] and C=$Lmin_x[t$+1], if $x[1..t] \approx y[1..t]$ then:

$$x[1..t+1] \approx y[1..t+1] \Leftrightarrow \forall_{a\in A}(y[t+1] = y[a]) \wedge \forall_{b\in B}(y[t+1] > y[b]) \wedge \forall_{c\in C}(y[t+1] < y[c]). \quad (2)$$

When allowing uncertainties between pairs of characters, previous research on the OPPM problem cannot be straightforwardly extended due to the need to trace $O(2^m)$ assignments on indeterminate strings.

▶ **Lemma 4.** *Given a determinate string x, an indeterminate string y, and the singleton sets A=$Leq_x[t+1]$, B=$Lmax_x[t+1]$ and C=$Lmin_x[t+1]$ containing a position in 1..t. If $x[1..t] \approx y[1..t]$ is verified on a specific assignment of y characters, denoted $\S y$, then:*

$$x[1..t+1] \approx y[1..t+1] \Leftrightarrow \exists_{\$y[t+1]\in\S y[t+1]}\forall_{a\in A}\exists_{\$y[a]\in\S y[a]}\forall_{b\in B}\exists_{\$y[b]\in\S y[b]}\forall_{c\in C}\exists_{\$y[c]\in\S y[c]}$$
$$\$y[t+1] = \$y[a] \ \bigwedge\ \$y[t+1] > \$y[b] \ \bigwedge\ \$y[t+1] < \$y[c]$$

Proof. ($\Rightarrow$) In accordance with *Leq*, *Lmax* and *Lmin* definition, for any $a\in A$, $b\in B$ and $c\in C$ we have $x[t+1]$=$x[a]$, $x[t+1]$>$x[b]$ and $x[t+1]$<$x[c]$. If there is an assignment to $y[1..t+1]$ in $\S y$ that preserves the orders of $x[1..t+1]$, then for each $a\in A$, $b\in B$ and $c\in C$ $\$y[t+1]$=$\$y[a]$, $\$y[t+1]$>$\$y[b]$ and $\$y[t+1]$<$\$y[c]$ (where $\$y[t+1] \in \S y[t+1]$, $\$y[a] \in \S y[a]$, $\$y[b] \in \S y[b]$, $\$y[c] \in \S y[c]$). ($\Leftarrow$) We need to show that $x[1..t+1] \approx y[1..t+1]$. Since $x[1..t] \approx y[1..t]$, for $i < t$, $\exists_{\$y[i]\in\S y[i],\$y[t+1]\in\S y[t+1]}$: $x[t+1]$>$x[i] \Leftrightarrow \$y[t+1]$>$\$y[i]$. Assuming $x[t+1]$>$x[i]$ for some $i \in \{1..t\}$: by the definition of *Lmax*, $\forall_{b\in B}x[b]$>$x[i]$; by the order-isomorphism of $x[1..t]$ and $\$y[1..t]$ in $\S y[1..t]$, there is $\$y[i] \in \S y[i]$ and $\$y[b] \in \S y[b]$ that $\forall_{b\in B}\$y[b]$>$\$y[i]$; and by the assumption of the lemma, $\forall_{b\in B}\$y[t+1]$>$\$y[b]$; hence $\$y[t+1]$>$\$y[i]$. Similarly, $x[t+1]$<$x[i]$ (and $x[t+1]$=$x[i]$) implies $\$y[t+1]$<$\$y[i]$ (and $\$y[t+1]$=$\$y[i]$), yielding the stated equivalence. ◀

Given two strings of equal length, the μOPPM problem can be schematically represented according to the identified order restrictions. Figure 2 represents restrictions on the indeterminate string y=$(2, 4|5, 3|5, 1|2)$ in accordance with the observed orders in x=(1,4,3,1).

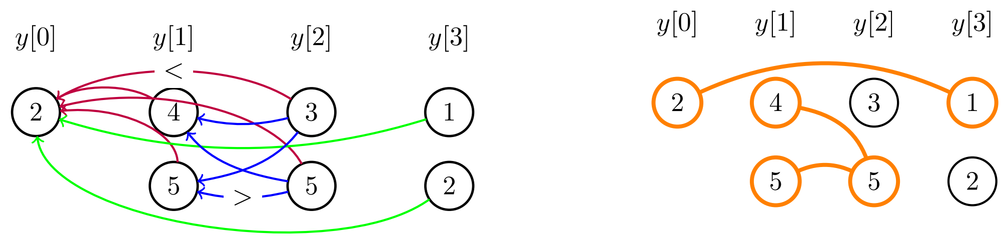


Figure 2 Schematic representation of the pairwise ordering restrictions for text y=$(2,4|5,3|5,1|2)$ and pattern x=(1,4,3,1). In the left side, all order verifications are represented, while in the right side only the order conflicts are signaled (e.g. $y[1]$=4 cannot be selected together with $y[2]$=5).

The left side edges are placed in accordance with *Lemma 4* and capture assessments on the orders between pairs of characters. The right side edges capture incompatibilities detected after the assessments, i.e. pairs of characters that cannot be selected simultaneously (for instance, $y[0]$=2 and $y[3]$=1, or $y[1]$=4 and $y[2]$=5). For the given example, there are two valid assignments, $\$y_1$=(2,4,3,2) and $\$y_2$=(2,5,4,2), that satisfy $x[0]$=$x[3]$<$x[2]$<$x[1]$, thus y op-matches x.

To verify whether there is an assignment that satisfies the identified ordering restrictions, we propose the reduction of μOPPM problem to a Boolean satisfiability problem.

Given a set of Boolean variables, a formula in conjunctive normal form is a conjunction of clauses, where each clause is a disjunction of literals, and a literal corresponds to a variable or its negation. Let a 2CNF formula be a formula in the conjunctive normal form with at most two literals per clause. Given a CNF formula, the *satisfiability* (SAT) problem is to verify if there is an assigning of values to the Boolean variables such that the CNF formula is satisfied.

▶ **Theorem 5.** *The μOPPM problem over two strings of equal length, one being indeterminate, can be reduced to a satisfiability problem with the following CNF formula:*

$$\begin{aligned}\phi = &\bigwedge_{i=0}^{m-1}\Big(\bigvee_{\$y[i]\in y[i]} z_{i,\$y[i]}\Big)\\ &\wedge \bigwedge_{i=0}^{m-1}\Big(\bigwedge_{\$y[i]\in y[i]}\ \bigwedge_{j\in Leq[i],\$y[j]\in y[j]} (\neg z_{i,\$y[i]}\vee\neg z_{j,\$y[j]}\vee \$y[i]=\$y[j])\\ &\wedge \bigwedge_{\$y[i]\in y[i]}\ \bigwedge_{\substack{j\in Lmax[i]\\ \$y[j]\in y[j]}} (\neg z_{i,\$y[i]}\vee\neg z_{j,\$y[j]}\vee \$y[i]>\$y[j])\\ &\wedge \bigwedge_{\$y[i]\in y[i]}\ \bigwedge_{\substack{j\in Lmin[i]\\ \$y[j]\in y[j]}} (\neg z_{i,\$y[i]}\vee\neg z_{j,\$y[j]}\vee \$y[i]<\$y[j])\Big)\end{aligned} \tag{3}$$

Proof. Let us show that if x op-matches y then ϕ is satisfiable, and if x does not op-match y then ϕ is not satisfiable. ($\Rightarrow$) When $x\approx y$, there is an assignment of values to y, $\$y$, that satisfy the orderings of x. ϕ is satisfiable if there is at least one variable assigned to true per clause $\vee_{\$y[i]\in y[i]}\ z_{i,\$y[i]}$ given conflicts $\neg z_{i,\$y[i]}\vee\neg z_{j,\$y[j]}$. As conflicts do not prevent the existence of a valid assignment (by assumption), then $\exists_{\$y}\wedge_{i\in\{0..m-1\}} z_{i,\$y[i]}$ and ϕ is satisfiable. ($\Leftarrow$) When x does not op-match y, there is no assignment of values $\$y$∈$y$ that can satisfy the orders of x. Per formulation, the conflicts $\neg z_{i,\$y[i]} \vee \neg z_{j,\$y[j]}$ prevent the satisfiability of one or more clauses $\vee_{\$y[i] \in y[i]}\ z_{i,\$y[i]}$, leading to a non-satisfiable formula. ◀

If the established ϕ formula is satisfiable, there is a Boolean assignment to the variables that specify an assignment of characters in y, $\$y$, preserving the orders of x (as defined by *Leq*, *Lmax* and *Lmin*). Otherwise, it is not possible to select an assignment $\$y$ op-matching x. ϕ has at most $r{\times}m$ variables, $\{z_{i,\sigma} \mid i \in \{0..m\text{-}1\}, \sigma \in \Sigma\}$. The Boolean value assigned to a variable $z_{i,\sigma}$ simply defines that the associated character σ from $y[i]$ can be either considered (when true) or not (when false) to compose a valid assignment $\$y$ that op-matches the given determinate string x. The reduced (3) formula is composed of two major types of clauses: $\vee_{\$y[i] \in y[i]} z_{i,\$y[i]}$, and $(\neg z_{i,\$y[i]} \vee \neg z_{j,\$y[j]} \vee \mathsf{bool})$ where bool is either given by $\$y[i] = \$y[j]$, $\$y[i] < \$y[j]$or $\$y[i] > \$y[j]$. Clauses of the first type specify the need to select at least one character per position in y to guarantee the presence of valid assignments. The remaining clauses specify ordering constraints between characters. If an inequality, such as $\$y[i] > \$y[j]$, is assessed as true, the associated clause is removed. Otherwise, $(\neg z_{i,\sigma_1} \vee \neg z_{j,\sigma_2})$ is derived, meaning that these σ_1 and σ_2 characters should not be selected simultaneously since they do not satisfy the orders defined by a given pattern. For instance, the pairs of characters in orange from Figure 2 should not be simultaneously selected due to order conflicts. To this end, $(\neg z_{0,2} \vee \neg z_{3,1})$ and $(\neg z_{1,4} \vee \neg z_{2,5})$ clauses need to be included to verify if $y \approx x$. Considering $y{=}(2, 4|5, 4|5, 1|2)$ and $x{=}(1,4,3,1)$, schematically represented in Figure 2, the associated CNF formula is:

$$\phi = z_{0,2} \wedge (z_{1,4} \vee z_{1,5}) \wedge (z_{2,4} \vee z_{2,5}) \wedge (z_{3,1} \vee z_{3,2}) \wedge (\neg z_{0,2} \vee \neg z_{3,1}) \wedge (\neg z_{1,4} \vee \neg z_{2,5})$$

▶ **Theorem 6.** *Given two strings of length m, one being indeterminate with $r{=}2$, the μOPPM problem can be reduced to a 2SAT problem with a CNF formula with $O(m)$ size.*

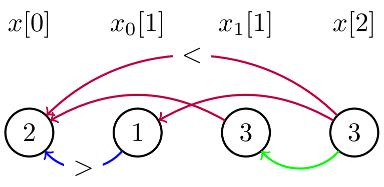
Proof. Given *Theorem 5* and the fact that the reduced CNF formula has at most two literals per clause – ϕ is a composition of $\vee_{\$y[i] \in y[i]} z_{i,\$y[i]}$ clauses with $|y[i]| \in \{1, 2\}$ and $(\neg z_{i,\$y[i]} \vee \neg z_{j,\$y[j]} \vee \mathsf{bool})$ clauses – μOPPM with $r{=}2$ and one indeterminate string is reducible to 2SAT. The reduced formula has at most $10m$ clauses with 2 literals each, being linear in m:

- [*clauses that impose the selection of at least one character per position in y*] Since y has m positions, and each position is either determinate (unitary clause) or defines an uncertainty between a pair of characters, there are m clauses and at most $2m$ literals;
- [*clauses that define the ordering restrictions between two variables*] A position in the indeterminate string $y[i]$ needs to satisfy at most two order relations. Considering that i, $Leq[i]$, $Lmax[i]$ and $Lmin[i]$ specify uncertainties between pairs of characters, there are up to 12 restrictions per position: 4 ordering restrictions between characters in $y[i]$ and $y[Leq[i]]$, $y[Lmax[i]]$ and $y[Lmin[i]]$. Whenever the order between two characters is not satisfied, a clause is added per position, leading to at most $12m$ clauses. ◀

▶ **Theorem 7.** *The μOPPM between determinate and indeterminate strings of equal length can be solved in linear time when $r{=}2$.*

Proof. Given the fact that a 2SAT problem can be solved in linear time [10][1], this proof directly derives from *Theorem 6* as it guarantees the soundness of reducing μOPPM ($r{=}2$) to a 2SAT problem with a CNF formula with $O(m)$ size. ◀

[1] 2SAT problems have linear time and space solutions on the size of the input formula. Consider for instance the original proposal [10], the formula ϕ is modeled by a directed graph $G{=}(V, E)$, with two nodes per variable z_i in ϕ (z_i and $\neg z_i$) and two directed edges for each clause $z_i \vee z_j$ (the equivalent



Pattern	2	1	3	3
i	0	1	1	2
j	0	0	1	0
Leq$[i\|j]$	$\emptyset$	$\emptyset$	$\emptyset$	{1}
Ordered indexes (asc)	1	0	2	3
Lmax$[i\|j]$	$\emptyset$	$\emptyset$	{0}	{0,1}
Ordered indexes (desc)	2	3	0	1
Lmin$[i\|j]$	$\emptyset$	{0}	$\emptyset$	$\emptyset$

Figure 3 Order relationships of $x=(2,1|3,3)$ and the corresponding *Lmax* and *Lmin* vectors.

As the size of the mapped CNF formula ϕ is $O(m)$ and the a valid algorithm to verify its satisfiability would require the construction of a graph with $O(m)$ nodes and edges, the required memory for the target μOPPM problem is $\Theta(m)$.

When moving from one to two indeterminate strings, previous contributions are insufficient to answer the μOPPM problem. In this context, the *Leq*, *Lmax* and *Lmin* vectors need to be redefined to be inferred from an indeterminate string:

▶ **Definition 8.**
- $Leq_x[i|j]=\{k : k<i,\ \exists_p\ \$x_j[i]=\$x_p[k]\}$ ($\emptyset$ if there is no eligible k), for $i=0,\ldots,m-1$
- $Lmax_x[i|j]=\{k : k<i,\ \exists_p\ \$x_j[i]>\$x_p[k]\}$ ($\emptyset$ if there is no eligible k), for $i=0,\ldots,m-1$
- $Lmin_x[i|j]=\{k : k<i,\ \exists_p\ \$x_j[i]<\$x_p[k]\}$ ($\emptyset$ if there is no eligible k), for $i=0,\ldots,m-1$

Figure 3 schematically represents the order relationships of $x=(2,1|3,3)$ and the associated *Leq*, *Lmax* and *Lmin* vectors. In this scenario, $x[2]$ needs to be verified not only against $x_0[1]$ but also against $x_1[1]$ in case $x_0[1]$ is disregarded. Understandably, due to character uncertainties, $O(m^2)$ ordering verifications are required (*Def.8*).

▶ **Lemma 9.** *Given indeterminate strings x and y, let $A_j=Leq_x[t+1|j]$, $B_j=Lmax_x[t+1|j]$ and $C_j=Lmin_x[t+1|j]$ (Def.8) be the orders associated with $\$x_j[t+1]$. If $x[1..t]\approx y[1..t]$ is verified on a partial assignment of y characters, denoted by $\S y$, then:*

$$x[1..t+1]\approx y[1..t+1] \Leftrightarrow \exists_{j\in\{0,1\}}\exists_{\$y[t+1]\in\S y[t+1]}\forall_{a\in A_j,b\in B_j,c\in C_j}$$

$$\exists_{\$y[a]\in\S y[a],\$y[b]\in\S y[b],\$y[c]\in\S y[c]}\Big(\$y[t{+}1]=\$y[a]\ \bigwedge\ \$y[t{+}1]>\$y[b]\ \bigwedge\ \$y[t{+}1]<\$y[c]\Big)$$

Proof. ($\Rightarrow$) Similar to the proof of *Lemma 4*, yet A, B and C conditional to $x[t+1]$ (Def.3) are now given by A_j, B_j and C_j conditional to $x_j[t+1]$ (Def.8). If there is an assignment to $y[1..t+1]$ in $\S y$ that preserves one of the possible orders in $x[1..t+1]$, then for any $a\in A_j$, $b\in B_j$ and $c\in C_j$: $\$y[t+1]=\$y[a]$, $\$y[t+1]>\$y[b]$ and $\$y[t+1]<\$y[c]$ (where $\$y[t+1]\in\S y[t+1]$, $\$y[a]\in\S y[a]$, $\$y[b]\in\S y[b]$, $\$y[c]\in\S y[c]$).

($\Leftarrow$) We need to show that $x[1..t+1]\approx y[1..t+1]$. Since $x[1..t]\approx y[1..t]$, it is sufficient to prove that for $i\leq t$: exists $\$x[i]\in\S x[i], \$x[t+1]\in\S x[t+1], \$y[i]\in\S y[i], \$y[t+1]\in$

implicative forms $\neg z_i\Rightarrow z_j$ and $\neg z_j\Rightarrow z_i$). Given G, the strongly connected components (SCCs) of G can be discovered in $O(|V|+|E|)$. During the traversal if a variable and its complement belong to the same SCC, then the procedure stops as ϕ is determined to be unsatisfiable. Given the fact that both $V=O(m)$ and $E=O(m)$ by *Lemma 6*, this procedure is $O(m)$ time and space.

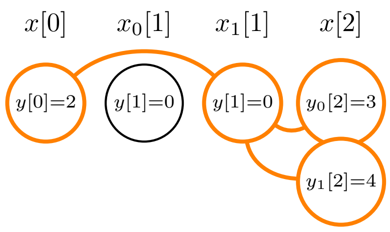


Figure 4 Conflicts when op-matching $y{=}(2,0,3|4)$ against $x{=}(2,1|3,3)$.

$\S y[t+1]$ such that $\$x[t+1]{=}\$x[i] \Leftrightarrow \$y[t+1]{=}\$y[i]$, $\$x[t+1]{>}\$x[i] \Leftrightarrow \$y[t+1]{>}\$y[i]$ and $\$x[t+1]{<}\$x[i] \Leftrightarrow \$y[t+1]{<}\$y[i]$. This results from Def.8, the order-isomorphism property and *Lemma 4*. ◀

Figure 4 represents encountered restrictions when op-matching $x{=}(2,1|3,3)$ against $y{=}(2,0,3|4)$. The right side edges capture the detected incompatibilities, i.e. pairs of characters that cannot be selected simultaneously. For the given example, there are 2 valid assignments – $\$y_1{=}(2{,}0{,}3)$ and $\$y_2{=}(2{,}0{,}4)$ – satisfying $\$x_0[1]{<}\$x_0[0]{<}\$x_0[2]$, thus $x \approx y$.

To verify whether there is an assignment that satisfies the identified ordering restrictions, Theorem 10 extends the previously introduced SAT mapping given by (3).

▶ **Theorem 10.** *Given Lmax and Lmin (Def.8), μOPPM problem over two indeterminate strings of equal length can be reduced to a satisfiability problem with the following CNF formula:*

$$\phi = \bigwedge_{x[i]\in x \wedge |x[i]|=1} w_{i,x_0[i]} \wedge \bigwedge_{x[i]\in x \wedge |x[i]|>1} \Bigg(\Big(\bigvee_{\$x[i]\in x[i]} w_{i,\$x[i]}\Big) \wedge \Big(\bigvee_{\$x[i]\in x[i]} \neg w_{i,\$x[i]}\Big)\Bigg) \tag{4.1}$$

$$\wedge \bigwedge_{i=0}^{m-1} \Bigg(\bigwedge_{\$x[i]\in x[i]} \Big(\bigvee_{\$y[i]\in y[i]} z_{i,\$x[i],\$y[i]}\Big) \wedge \Big(\bigvee_{\$y[i]\in y[i]} \neg z_{i,\$x[i],\$y[i]}\Big)\Bigg) \tag{4.2}$$

$$\wedge \bigwedge_{i=0}^{m-1} \Bigg(\Big(\bigwedge_{\$x[i]\in x[i]} \bigwedge_{\$y[i]\in y[i]} \Big(\neg z_{i,\$x[i],\$y[i]} \bigvee w_{i,\$x[i]}\Big)\Bigg) \tag{4.3}$$

$$\begin{aligned} \wedge \bigwedge_{i=0}^{m-1} \bigwedge_{\$y[i]\in y[i],\$x[i]\in x[i]} \Bigg(& \\ & \bigwedge_{j\in Leq[i]} \bigwedge_{\$y[j]\in y[j],\$x[j]\in x[j]} \big(\neg z_{i,\$x[i],\$y[i]} \vee \neg z_{j,\$x[j],\$y[j]} \vee \$y[i] = \$y[j]\big) \\ & \wedge \bigwedge_{j\in Lmax[i]} \bigwedge_{\substack{\$y[j]\in y[j]\\ \$x[j]\in x[j]}} \begin{pmatrix} \neg z_{i,\$x[i],\$y[i]} \vee \neg z_{j,\$x[j],\$y[j]} \\ \vee \$y[i] > \$y[j] \end{pmatrix} \\ & \wedge \bigwedge_{j\in Lmin[i]} \bigwedge_{\substack{\$y[j]\in y[j]\\ \$x[j]\in x[j]}} \begin{pmatrix} \neg z_{i,\$x[i],\$y[i]} \vee \neg z_{j,\$x[j],\$y[j]} \\ \vee \$y[i] < \$y[j] \end{pmatrix}\Bigg) \end{aligned} \tag{4.4}$$

Proof. If $x \approx y$ then ϕ is satisfiable, and if x does not op-match y then ϕ is not satisfiable.

($\Rightarrow$) When x op-matches y, there is an assignment of values in x and y such that $\$x \approx \$y$. ϕ is satisfiable if there is one and only one variable $w_{i,\$x[i]}$ per i^{th} position (4.1). This occurs iff one of the variables $z_{i,\$x[i],\$y[i]}$ is true for a given i^{th} position in accordance with (4.2) and

(4.3). As conflicts (4.4) do not prevent the existence of a valid assignment (by assumption), one or more variables $z_{i,\$x[i],\$y[i]}$ can be selected per position. ϕ can then be satisfied by fixing a single variable $z_{i,\$x[i],\$y[i]}$ per i^{th} position as true and the remaining variables as false. Given (4.3) equivalences between $w_{i,\$x[i]}$ and $z_{i,\$x[i],\$y[i]}$ variables, ϕ is consequently satisfiable.

($\Leftarrow$) When x does not op-match y, there is no assignment of values $\$x{\in}x$ and $\$y{\in}y$ such that $\$x \approx \$y$. In this context, the conflicts (4.4) can prevent the satisfiability of clauses (4.2) or (4.1), thus leading to an unsat formula. Per formulation, in the absence of an order-preserving match, conflicts will prevent the assignment of at least one variable $z_{i,\$x[i],\$y[i]}$ on compatible $(i, \$x[i])$ pairs, which are necessarily shown as conflicts on (4.1) clauses as a consequence of the assignment constraints placed by (4.2) and (4.3) clauses. ◀

If the formula is satisfiable, there is a Boolean assignment to the variables such that there is an assignment of characters in y, $\$y$, and in x, $\$x$, such that both strings op-match. Otherwise, it is not possible to select assignments such that $x \approx y$. Given $r{=}2$, the established ϕ formula has at most $6m$ Boolean variables: a) at most $2m$ variables of the type $\{w_{i,\sigma} \mid i \in\{0..m\text{-}1\},\ \sigma \in \Sigma\}$ corresponding to characters in x; and b) at most $4m$ variables of the type $\{z_{i,\sigma_1,\sigma_2} \mid i \in\{0..m\text{-}1\},\ \sigma_1, \sigma_2 \in \Sigma\}$ defining combinations of characters in the i^{th} position of x and y. Boolean values assigned to variables $w_{i,\sigma}$ are used to find a valid assignment of characters in x. Boolean values assigned to variables z_{i,σ_1,σ_2} define whether characters $\sigma_1 \in x[i]$ and $\sigma_2 \in y[i]$ belong to an op-match. The reduced formula is composed of four major types of clauses:

- (4.1) a single character per x position should be selected if exists $\$x$ such that $\$x \approx y$;
- (4.2) a single character per indeterminate y position should be selected if there is a valid assignment $\$y$ such that $\$x \approx \$y$, where $\$x$ is given by assignments to (4.1) clauses;
- (4.3) clauses that guarantee an association between x and y: $z_{i,\$x[i],\$y[i]} \Rightarrow w_{i,\$x[i]}$;
- (4.4) clauses specify ordering constraints between pairs of characters $\sigma_1 \in y[i]$ and $y[Leq[i]]$, $y[Lmax[i]]$ and $y[Lmin[i]]$. If the inequalities $\$y[i]{=}\$y[j]$, $\$y[i]{>}\$y[j]$ and $\$y[i]{<}\$y[j]$ are assessed as false, these leads to clauses of the form $(\neg z_{i,\sigma_1} \vee \neg z_{j,\sigma_2})$, meaning that these characters should not be selected simultaneously in the given positions (see Figure 4).

To instantiate the proposed mapping, consider $x{=}(2, 1|3, 3)$ and $y{=}(2, 0, 3|4)$, schematically represented in Figure 3. The associated CNF formula is:

$$\begin{aligned}\phi = {} & w_{0,2} \wedge (w_{1,1} \vee w_{1,3}) \wedge (\neg w_{1,1} \vee \neg w_{1,3}) \wedge w_{2,3} \quad \textit{//(4.1) one valid assignment to } x\\ & \wedge (z_{2,3,3} \vee z_{2,3,4}) \wedge (\neg z_{2,3,3} \vee \neg z_{2,3,4}) \quad \textit{//(4.2) assignment to indeterminate } y \textit{ positions}\\ & \wedge (\neg z_{0,2,2} \vee w_{0,2}) \wedge (\neg z_{1,1,0} \vee w_{1,1}) \wedge (\neg z_{1,3,0} \vee w_{1,3}) \wedge (\neg z_{2,3,3} \vee w_{2,3})\\ & \qquad \wedge (\neg z_{2,3,4} \vee w_{2,3}) \qquad \textit{//(4.3) implications between } x \textit{ and } y\textit{: } z_{i,\$x[i],\$y[i]} \Rightarrow w_{i,\$x[i]}\\ & \wedge (\neg z_{0,0,2} \vee \neg z_{1,3,0}) \wedge (\neg z_{1,3,0} \vee \neg z_{2,3,3}) \wedge (\neg z_{1,3,0} \vee \neg z_{2,3,4}) \quad \textit{//4.4 character conflicts}\end{aligned}$$

▶ **Theorem 11.** *When $r{=}2$, the μOPPM problem for two indeterminate strings of equal length is reducible to a 2-satisfiability problem over a CNF formula with $O(m^2)$ size.*

Proof. The reduced formula (4) is in the two conjunctive normal form (2CNF). (4.1), (4.2) and (4.3) clauses have at most two literals given $r{=}2$. (4.4) clauses contain inequalities dynamically assigned to true or false during the reduction phase, producing clauses with at most two literals. There are at most $2m$ clauses given by (4.1) as x has at most 2 characters per position; and at most $4m$ clauses given by (4.2) (as well as $4m$ clauses given by (4.3)) resulting from the combination of possible characters from a position in x and y. Since there is a maximum of $O(m)$ orders per position, there can be at most $O(m^2)$ order conflicts between characters and thus $O(m^2)$ clauses given by (4.4) of the form $(\neg z_{i,\$x[i],\$y[i]} \vee \neg x_{j,\$x[j],\$y[j]})$. ◀

▶ **Theorem 12.** *μOPPM indeterminate strings of equal length is in $O(m^2)$ time when r=2.*

Proof. Given *Theorem 11* and the ability to solve 2SAT tasks linearly in the size of the CNF formula [10], the proof of this theorem follows naturally. ◀

As linear time algorithms to solve 2SAT problems require linear space (see appendix) and the size of the mapped satisfiability formula ϕ is $O(m^2)$ (*Theorem 11*), the memory complexity of the μOPPM problems between indeterminate strings with r=2 and equal length is $O(m^2)$.

4 Polynomial time μOPPM

▶ **Lemma 13.** *Given a pattern string of length m and a text string of length n, one being indeterminate, the μOPPM problem can be solved in $O(nmr \lg r)$ time. When both the pattern and text are indeterminate with r=2, the μOPPM problem can be solved in $O(nm^2)$ time.*

Proof. From *Lemmas 7* and *12*: verifying if two strings of length m op-match can be either done in $O(mr \lg r)$ time (indetermination in one string) or $O(m^2)$ time (indetermination on both strings and r=2). At most $n{-}m{+}1$ verifications need to be performed. ◀

Lemma 13 confirms that the μOPPM problem with one indeterminate strings or uncertainties between characters (r=2) is in class **P**. This lemma further triggers the research question "*Are $O(nmr)$ and $O(nm^2)$ tight bounds to solve the μOPPM?*", here left as an open research question.

Irrespectively of the answer, the analysis of the average complexity is of complementary relevance. State-of-the-art research on the exact OPPM problem shows that the average performance of algorithms in $O(nm)$ time can outperform linear algorithms [20, 17, 19].

Motivated by the evidence gathered by these works, we suggest the use of filtration procedures to improve the average complexity of the proposed μOPPM algorithm while still preserving its complexity bounds. A filtration procedure encodes the input pattern and text, and relies on this encoding to efficiently find positions in the text with a high likelihood to op-match a given pattern. Despite the diversity of string encodings, simplistic binary encodings are considered to be the state-of-the-art in OPPM research [20, 17]. In accordance with Chhabra et al. [20], a pattern p can be mapped into a binary string p' expressing increases (1), equalities (0) and decreases (0) between subsequent positions. By searching for exact pattern matches of p' in an analogously transformed text string t', we guarantee that the verification of whether $p[0..m\text{-}1]$ and $t[i..i{+}m\text{-}1]$ orders are preserved is only performed when exact binary matches occur. Illustrating, given p=(3,1,2,4) and t=(2,4,3,5,7,1,4,8), then p'=(1,0,1,1) and t'=(1,1,0,1,1,0,1,1,0), revealing two matches $t'[1..4]$ and $t'[4..7]$: one spurious match $t[1..5]$ and one true match $t[4..8]$.

When handling indeterminate strings the concept of increase, equality and decrease needs to be redefined. Given an indeterminate string x, consider $x'[i]$=1 if $max(x[i]){<}min(x[i+1])$, $x'[i]$=0 if $min(x[i]){\geq}max(x[i+1])$, and $x'[i]{=}*$ otherwise. Under this encoding, the pattern matching problem is identical under the additional guard that a character in p' always matches a don't care position, $t'[i]{=}*$, and vice-versa. Illustrating, given p=(6,2|3,5) and t=(3|4,5,6|8,6|7,3,5,4|6,7|8,4), then p'=(0,1) and t'=(11∗01∗10), leading to one true match $t[3..5]$ – e.g. $t[3..5]=(6,3,5) – and one spurious match $t[5..7]$. Exact pattern matching algorithms, such as Knuth-Morris-Pratt and Boyer-Moore, can be adapted to consider don't care positions while preserving complexity bounds [40, 13].

The properties of the proposed encoding guarantee that the exact matches of p' in t' cannot skip any op-match of p in t. Thus, when combining the premises of *Lemma 13* with the previous observation, we guarantee that the computed μOPPM solution is sound.

The application of this simple filtration procedure prevents the recurring $O(mr \lg r)$ or $O(m^2)$ verifications $n{-}m{+}1$ times. Instead, the complexity of the proposed method to solve the μOPPM problem becomes $O(dmr \lg r + n)$ (when one string is indeterminate) or $O(dm^2 + n)$ (when both strings are indeterminate and $r{=}2$) where d is the number of exact matches ($d \ll n$). According to previous work on exact OPPM with filtration procedures [20], SBNDM2 and SBNDM4 algorithms [28] (Boyer-Moore variants) were suggested to match binary encodings. In the presence of small patterns, Fast Shift-Or (FSO) [30] can be alternatively applied [20].

A given string text can be read and encoded incrementally from the standard input as needed to perform μOPPM, thus requiring $O(mr)$ space. When filtration procedures are considered, the aforementioned algorithms for exact pattern matching require $O(m)$ space [20], thus μOPPM space requirements are bound by substring verifications (*Section 3*): $O(mr)$ space when one string is indeterminate and $O(m^2)$ when indetermination is considered on both strings and $r{=}2$.

5 Concluding remark

This work addressed the relevant yet scarcely studied problem of finding order-preserving pattern matches on indeterminate strings (μOPPM). We showed that the problem has a polynomial solution when uncertainties are verified between two characters by reducing the μOPPM problem to a 2-satisfiability problem. To this end, we first demonstrated that the problem of matching two strings with equal length can be solved in linear time and space when considering indetermination in one string and in quadratic time when considering indetermination on both the pattern and text strings. Finally, we showed that the μOPPM problem can be efficiently solved in polynomial time by combining the proposed verifications with filtration procedures.

References

1 Amihood Amir, Yonatan Aumann, Piotr Indyk, Avivit Levy, and Ely Porat. Efficient computations of l_1 and l_{infinity} rearrangement distances. *Theor. Comput. Sci.*, 410(43):4382–4390, 2009. doi:10.1016/j.tcs.2009.07.019.

2 Amihood Amir, Yonatan Aumann, Gad M. Landau, Moshe Lewenstein, and Noa Lewenstein. Pattern matching with swaps. *J. Algorithms*, 37(2):247–266, 2000. doi:10.1006/jagm.2000.1120.

3 Amihood Amir, Yonatan Aumann, Moshe Lewenstein, and Ely Porat. Function matching. *SIAM Journal on Computing*, 35(5):1007–1022, 2006.

4 Amihood Amir, Richard Cole, Ramesh Hariharan, Moshe Lewenstein, and Ely Porat. Overlap matching. *Inf. Comput.*, 181(1):57–74, 2003. doi:10.1016/S0890-5401(02)00035-4.

5 Amihood Amir and Martin Farach. Efficient 2-dimensional approximate matching of half-rectangular figures. *Inf. Comput.*, 118(1):1–11, 1995. doi:10.1006/inco.1995.1047.

6 Amihood Amir, Martin Farach, and S. Muthukrishnan. Alphabet dependence in parameterized matching. *Inf. Process. Lett.*, 49(3):111–115, 1994. doi:10.1016/0020-0190(94)90086-8.

7 Amihood Amir, Ohad Lipsky, Ely Porat, and Julia Umanski. Approximate matching in the l1 metric. In *CPM*, volume 5, pages 91–103. Springer, 2005.

8 Amihood Amir and Igor Nor. Generalized function matching. *J. Discrete Algorithms*, 5(3):514–523, 2007. doi:10.1016/j.jda.2006.10.001.

9 Alberto Apostolico. General pattern matching. In Mikhail J. Atallah and Marina Blanton, editors, *Algorithms and Theory of Computation Handbook*, pages 15–15. Chapman & Hall/CRC, 2010.

10 Bengt Aspvall, Michael F Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.

11 Brenda S Baker. A theory of parameterized pattern matching: algorithms and applications. In *ACM symposium on Theory of computing*, pages 71–80. ACM, 1993.

12 Djamal Belazzougui, A. Pierrot, M. Raffinot, and Stéphane Vialette. Single and multiple consecutive permutation motif search. In *Int. Symposium on Algorithms and Computation*, pages 66–77. Springer, 2013.

13 Robert S Boyer and J Strother Moore. A fast string searching algorithm. *Communications of the ACM*, 20(10):762–772, 1977.

14 Emilios Cambouropoulos, M. Crochemore, C. Iliopoulos, L. Mouchard, and Yoan Pinzon. Algorithms for computing approximate repetitions in musical sequences. *Int. Journal of Computer Mathematics*, 79(11):1135–1148, 2002.

15 Domenico Cantone, Salvatore Cristofaro, and Simone Faro. An efficient algorithm for δ-approximate matching with α-bounded gaps in musical sequences. In *IW on Experimental and Efficient Algorithms*, pages 428–439. Springer, 2005.

16 Domenico Cantone, Salvatore Cristofaro, and Simone Faro. On tuning the (δ, α)-sequential-sampling algorithm for δ-approximate matching with alpha-bounded gaps in musical sequences. In *ISMIR*, pages 454–459, 2005.

17 Domenico Cantone, Simone Faro, and M Oguzhan Külekci. An efficient skip-search approach to the order-preserving pattern matching problem. In *Stringology*, pages 22–35, 2015.

18 Tamanna Chhabra, Simone Faro, M. Oguzhan Külekci, and Jorma Tarhio. Engineering order-preserving pattern matching with SIMD parallelism. *Softw., Pract. Exper.*, 47(5):731–739, 2017. doi:10.1002/spe.2433.

19 Tamanna Chhabra, M Oguzhan Külekci, and Jorma Tarhio. Alternative algorithms for order-preserving matching. In *Stringology*, pages 36–46, 2015.

20 Tamanna Chhabra and Jorma Tarhio. A filtration method for order-preserving matching. *Inf. Process. Lett.*, 116(2):71–74, 2016. doi:10.1016/j.ipl.2015.10.005.

21 Sukhyeun Cho, Joong Chae Na, Kunsoo Park, and Jeong Seop Sim. Fast order-preserving pattern matching. In *Combinatorial Optimization and Applications*, pages 295–305. Springer, 2013.

22 Sukhyeun Cho, Joong Chae Na, Kunsoo Park, and Jeong Seop Sim. A fast algorithm for order-preserving pattern matching. *Information Processing Letters*, 115(2):397–402, 2015.

23 Peter Clifford, Raphaël Clifford, and Costas Iliopoulos. Faster algorithms for δ, γ-matching and related problems. In *Annual Symposium on Combinatorial Pattern Matching*, pages 68–78. Springer, 2005.

24 Raphaël Clifford and C Iliopoulos. Approximate string matching for music analysis. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 8(9):597–603, 2004.

25 Richard Cole, C. Iliopoulos, T. Lecroq, W. Plandowski, and Wojciech Rytter. On special families of morphisms related to δ-matching and don't care symbols. *Information Processing Letters*, 85(5):227–233, 2003.

26 Maxime Crochemore, Costas S Iliopoulos, Thierry Lecroq, Wojciech Plandowski, and Wojciech Rytter. Three heuristics for delta-matching: delta-bm algorithms. In *CPM*, pages 178–189. Springer, 2002.

27 Erik D Demaine, Alejandro López-Ortiz, and J Ian Munro. Adaptive set intersections, unions, and differences. In *In Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA*. Citeseer, 2000.

28 Branislav Ďurian, Jan Holub, Hannu Peltola, and Jorma Tarhio. Improving practical exact string matching. *Information Processing Letters*, 110(4):148–152, 2010.

29 Michael L. Fredman. On computing the length of longest increasing subsequences. *Discrete Mathematics*, 11(1):29–35, 1975. `doi:10.1016/0012-365X(75)90103-X`.

30 Kimmo Fredriksson and Szymon Grabowski. Practical and optimal string matching. In *SPIRE*, volume 3772, pages 376–387. Springer, 2005.

31 Kimmo Fredriksson and Szymon Grabowski. Efficient algorithms for pattern matching with general gaps, character classes, and transposition invariance. *Information Retrieval*, 11(4):335–357, 2008.

32 Xianping Ge. Pattern matching in financial time series data. *final project report for ICS*, 278, 1998.

33 Rui Henriques. *Learning from High-Dimensional Data using Local Descriptive Models*. PhD thesis, Instituto Superior Tecnico, Universidade de Lisboa, Lisboa, 2016.

34 Rui Henriques, Cláudia Antunes, and SaraC. Madeira. Methods for the efficient discovery of large item-indexable sequential patterns. In *New Frontiers in Mining Complex Patterns*, volume 8399 of *LNCS*, pages 100–116. Springer International Publishing, 2014.

35 Rui Henriques and Sara C Madeira. Bicspam: flexible biclustering using sequential patterns. *BMC bioinformatics*, 15(1):130, 2014.

36 Rui Henriques and Ana Paiva. Seven principles to mine flexible behavior from physiological signals for effective emotion recognition and description in affective interactions. In *PhyCS*, pages 75–82, 2014.

37 Jan Holub, William F. Smyth, and Shu Wang. Fast pattern-matching on indeterminate strings. *J. Discrete Algorithms*, 6(1):37–50, 2008. `doi:10.1016/j.jda.2006.10.003`.

38 Shuichi Kawashima and Minoru Kanehisa. Aaindex: amino acid index database. *Nucleic acids research*, 28(1):374–374, 2000.

39 Jinil Kim, Peter Eades, Rudolf Fleischer, Seok-Hee Hong, Costas S Iliopoulos, Kunsoo Park, Simon J Puglisi, and Takeshi Tokuyama. Order-preserving matching. *Theoretical Computer Science*, 525:68–79, 2014.

40 Donald E Knuth, James H Morris, Jr, and Vaughan R Pratt. Fast pattern matching in strings. *SIAM journal on computing*, 6(2):323–350, 1977.

41 Marcin Kubica, Tomasz Kulczyński, Jakub Radoszewski, Wojciech Rytter, and Tomasz Waleń. A linear time algorithm for consecutive permutation pattern matching. *Information Processing Letters*, 113(12):430–433, 2013.

42 Inbok Lee, Raphaël Clifford, and Sung-Ryul Kim. Algorithms on extended (δ, γ)-matching. *Computational Science and Its Applications-ICCSA 2006*, pages 1137–1142, 2006.

43 Inbok Lee, Juan Mendivelso, and Yoan J Pinzón. $\delta\gamma$–parameterized matching. In *International Symposium on String Processing and Information Retrieval*, pages 236–248. Springer, 2008.

44 Ohad Lipsky and Ely Porat. Approximate matching in the $\mathrm{l}_{\mathrm{infinity}}$ metric. *Inf. Process. Lett.*, 105(4):138–140, 2008. `doi:10.1016/j.ipl.2007.08.012`.

45 Juan Mendivelso, Inbok Lee, and Yoan J Pinzón. Approximate function matching under δ-and γ-distances. In *SPIRE*, pages 348–359. Springer, 2012.

46 S Muthukrishnan. New results and open problems related to non-standard stringology. In *Combinatorial Pattern Matching*, pages 298–317. Springer, 1995.

47 Ely Porat and Klim Efremenko. Approximating general metric distances between a pattern and a text. In *ACM-SIAM symposium on Discrete algorithms*, pages 419–427. SIAM, 2008.