# Periodicity Algorithms and a Conjecture on Overlaps in Partial Words*

F. Blanchet-Sadri[1] Robert Mercaş[2] Abraham Rashin[3]
Elara Willett[4]

February 8, 2012

**Abstract**

We propose an algorithm that given as input a full word $w$ of length $n$, and positive integers $p$ and $d$, outputs, if any exists, a maximal $p$-periodic partial word contained in $w$ with the property that no two holes are within distance $d$ (so-called $d$-valid). Our algorithm runs in $O(nd)$ time and is used for the study of repetition-freeness of partial words. Furthermore, we construct an infinite word over a five-letter alphabet that is overlap-free even after holes are inserted in arbitrary 2-valid positions, answering affirmatively a conjecture from Blanchet-Sadri, Mercaş, and Scott.

*Keywords*: Combinatorics on words; Partial words; Freeness; Overlaps; Algorithms; Periodicity.

## 1 Introduction

In [7], Manea and Mercaş extend the concept of repetition-freeness of full words to partial words which are sequences over a finite alphabet that may

*This material is based upon work supported by the National Science Foundation under Grant No. DMS–0754154. The Department of Defense is also gratefully acknowledged. Part of this material was presented at LATA'09 [2]. We thank the referees of preliminary versions of this paper for their very valuable comments and suggestions. This work was done during the second author's stay at the University of North Carolina at Greensboro.

[1]Department of Computer Science, University of North Carolina, P.O. Box 26170, Greensboro, NC 27402–6170, USA, blanchet@uncg.edu

[2]GRLMC, Universitat Rovira i Virgili, Departament de Filologies Romàniques, Av. Catalunya 35, Tarragona, 43002, Spain, robertmercas@gmail.com

[3]Department of Mathematics, Rutgers University, 110 Frelinghuysen Rd., Piscataway, NJ 08854–8019, USA

[4]Department of Mathematics, Oberlin College, 10 North Professor St., King 205, Oberlin, OH 44074-1019, USA

contain some "do not know" symbols called "holes." There, several problems regarding cube-freeness are investigated. Following the same lines, in [3], Blanchet-Sadri, Mercaş and Scott consider the concepts of square- and overlap-freeness. In these papers, the authors investigate the existence of infinite full words into which arbitrarily many holes can be inserted without introducing repetitions (inserting a hole is defined as replacing a letter with a hole in a fixed position of a word). A constraint that no two holes can be placed one or two positions apart is needed, to avoid trivial squares and cubes. This problem is equivalent to determining whether an infinite partial word $w$ can be found such that, none of its factors of length $lp$, for any positive integer $p$ and rational $l \geq 2$ such that $lp$ is an integer, is 2-valid and $p$-periodic. A partial word is called $d$-valid if any positions $i, j$ satisfying $0 < |i - j| \leq d$ are not both holes.

A well-known result of Thue states that over a binary alphabet there exist infinitely many overlap-free infinite full words [10, 11]. This result has proven useful for many problems in different fields [1]. In [7], the question was raised as to whether there exist overlap-free infinite partial words, and to construct them over a binary alphabet if such exist. In [3] and [5], the authors settle this question by showing that over a two-letter alphabet there exist overlap-free infinite partial words with one hole, and none exists with more than one hole. Moreover, in [3] it is shown that there exist infinitely many overlap-free infinite partial words with an arbitrary number of holes over a three-letter alphabet (see also the paper [4] on square-freeness). In addition, in [3], it is proved that there exists an infinite overlap-free word over a six-letter alphabet that remains overlap-free after an arbitrary selection of its letters are changed to holes, and none exists over a four-letter alphabet. The case of a five-letter alphabet remained open.

**Conjecture 1** ([3])**.** *There exists an infinite word over a five-letter alphabet that remains overlap-free after an arbitrary 2-valid insertion of holes.*

In this paper, we investigate the question of which finite full words can be made periodic by insertion of holes, with the restriction that no two holes be too close together. More precisely, we present an algorithm that determines whether a finite full word $w$ of length $n$ contains a $d$-valid $p$-periodic partial word, in $O(nd)$ time. As a consequence, we give a positive answer to Conjecture 1.

The contents of our paper is as follows: In Section 2, we review some basic concepts of combinatorics on partial words that are useful throughout the paper. In Section 3, we discuss our algorithm that transforms a full word of length $n$ into a $p$-periodic partial word by an insertion of holes so

that no two holes are within distance $d$. The algorithm has two major steps: computing our so-called constraint matrices, and then traversing a directed graph associated with those matrices. We analyze the complexity of our algorithm. We also discuss the special case of squares, giving simple criteria for transforming a full word of length $n = 2p$ so that it becomes a square after a $d$-valid insertion of holes. In Section 4, we give a polynomial time algorithm to generate all length $n$ factors of the fixed point of a morphism. In case the morphism is uniform, the time complexity is $O(n)$. In Section 5, we give our construction of an infinite word over a five-letter alphabet that settles the above mentioned conjecture. Finally in Section 6, we conclude with some remarks.

## 2 Preliminaries

Let $A$ be a nonempty finite set of symbols called an *alphabet*. Each element $a \in A$ is called a *letter*. A *full word* over $A$ is a finite sequence of letters from $A$. A *partial word* over $A$ is a finite sequence of letters from $A_\diamond = A \cup \{\diamond\}$, the alphabet $A$ extended with the hole symbol $\diamond$ (a full word is a partial word that does not contain the $\diamond$ symbol). A partial word $u$ of length $n$ over $A$ can be viewed as a function $u : \{0, \ldots, n-1\} \to A_\diamond$. The *length* of a partial word $u$ is denoted by $|u|$ and represents the total number of symbols in $u$. The *empty word* is the sequence of length zero and is denoted by $\varepsilon$. The powers of a partial word $u$ are defined recursively by $u^0 = \varepsilon$ and for $n \geq 1$, $u^n = uu^{n-1}$. The set containing all finite full words over the alphabet $A$ is denoted by $A^*$, while the set of all finite partial words over the alphabet $A$ is denoted by $A^*_\diamond$.

A *strong period* of a partial word $u$ over $A$ is a positive integer $p$ such that $u(i) = u(j)$ whenever $u(i), u(j) \in A$ and $i \equiv j \ mod \ p$. In such a case, we say $u$ is *strongly p-periodic*. A *weak period* of $u$ is a positive integer $p$ such that $u(i) = u(i+p)$ whenever $u(i), u(i+p) \in A$. In such a case, we say $u$ is *weakly p-periodic*. The word $abba\diamond bacb$ is weakly 3-periodic, but not strongly 3-periodic.

If $u$ and $v$ are two partial words of equal length, then $u$ is said to be *contained* in $v$, denoted $u \subset v$, if $u(i) = v(i)$, for all $u(i) \in A$. Partial words $u$ and $v$ are *compatible*, denoted by $u \uparrow v$, if there exists $w$ such that $u \subset w$ and $v \subset w$.

A partial word $u$ is a *factor* of a partial word $v$ if $v = xuy$ for some $x, y$. The factor $u$ is *proper* if $u \neq \varepsilon$ and $u \neq v$. We say that $u$ is a *prefix* of $v$ if $x = \varepsilon$ and a *suffix* of $v$ if $y = \varepsilon$. A full word $u$ is said to contain

an overlap if it contains a factor $avava$ (two overlapping occurrences of the word $ava$) with $a$ a letter and $v$ a word (for more details, see [6]). In [7], this definition was extended to partial words, an overlap being considered a factor $a_0v_0a_1v_1a_2$ with $v_0 \uparrow v_1$, and $a_0, a_1, a_2$ pairwise compatible symbols. It can be generalized as follows: a partial word $a_0v_0a_1v_1a_2$, where $v_0 \uparrow v_1$, is a

- *strong overlap* if $a_0, a_1, a_2$ are pairwise compatible symbols (in this case, $a_0v_0a_1v_1a_2 \subset avava$, for some letter $a$ and word $v$);
- *weak overlap* if $a_0 \uparrow a_1$ and $a_1 \uparrow a_2$, and in the case where $v_0, v_1 = \varepsilon$ we also have $a_0 \uparrow a_2$ (in this case, $a_0v_0a_1 \uparrow a_1v_1a_2$).

A partial word is *weakly overlap-free* (respectively, *strongly overlap-free*) if none of its factors is a weak (respectively, strong) overlap.

# 3 Periodic partial words with no two holes within a fixed distance

Let $w$ be a full word of length $n$ defined over an alphabet $A$ of size $k$. In this section, we present an $O(nd)$ time algorithm, which finds, for given positive integers $d$ and $p$ both less than $n$, a $d$-valid $p$-periodic partial word $u$ contained in $w$, if any exists. In other words, it determines whether it is possible to replace letters of $w$ by holes, with no two holes within distance $d$, such that the resulting partial word $u$ has strong period $p$. If this is possible, such a partial word is returned.

In order to work with partial words of length $n$ more easily, we write them in rows of length $p$. For an integer $x$, $0 \leq x < p$, we call *column* $x$ the sequence of positions (or letters at these positions) $x, x+p, \ldots, x+lp$, where $l$ is the largest integer such that $x+lp < n$. For example in Figure 1, if $w = abadecabbdeeaba$, $p = 6$, and $d = 2$, then $u = abadecab\diamond de\diamond aba$ is obtained using our algorithm (note that $u$ is strongly $p$-periodic since every column of $u$ contains no two distinct non-hole symbols).

In short, this is how the algorithm and its proof work:

1. Reduce the question of the existence of $u$ to one about choosing a letter for each column.
2. Determine how these choices interact, i.e., what are the constraints between columns?
3. Make a graph and traverse it, to take into account all the constraints.

Figure 1: The words $w$ and $u$ with columns 2 and 5 highlighted

4. During this process, either fail when a contradiction is reached, or end up with a choice of letters for the columns. Use this choice of letters to construct $u$, a $d$-valid $p$-periodic partial word contained in $w$.

### 3.1 Computing the constraint matrices

We say that positions $i, j$ in a partial word are *$d$-proximal* if $0 < |i - j| \leq d$. A partial word obeys the *hole constraint* $d$, or is $d$-valid, if no two holes are $d$-proximal. When the value of $d$ is clear from context, we may suppress reference to it, simply referring to the "hole constraint" or to "proximal" positions.

For an integer $x$, $0 \leq x < p$, let $S_x = \{w(i) \mid 0 \leq i < n, i \equiv x \bmod p\}$ be the set of distinct letters appearing in column $x$ of $w$. Call a partial word $u$, $u \subset w$, as being *induced* by the choice of letters $\omega \in S_0 \times S_1 \times \cdots \times S_{p-1}$, if $u \subset \omega^l$, for some rational $l$. Which $\omega \in \prod_{x=0}^{p-1} S_x$ induce $d$-valid partial words?

**Remark 1.** *The choice of letters $\omega \in \prod_{x=0}^{p-1} S_x$ induces a $d$-valid partial word $u$, $u \subset w$, if and only if for every two proximal positions $i, j$, $u(i) = w(i) = \omega(i \bmod p)$ or $u(j) = w(j) = \omega(j \bmod p)$.*

This suggests a geometric approach for determining which choices of letters do not cause a hole constraint violation. For $(a_0, b_0) \in A^2$, let the *cross centered at* $(a_0, b_0)$ be the set

$$+(a_0, b_0) = \{(a, b) \in A^2 \mid a = a_0 \text{ or } b = b_0\}$$

(If we draw this as a set of entries in a $k \times k$ matrix indexed by $A^2$, it looks like a cross centered at $(a_0, b_0)$.) Then, the choices of letters $\omega(x) = a$ for column $x$ and $\omega(y) = b$ for column $y$ that do not cause any hole constraint violations, are precisely those pairs $(a, b)$ in the intersection of the crosses centered at $(w(i), w(j))$, for all proximal positions $i, j$ in columns $x, y$, respectively. (If columns $x, y$ contain no proximal positions, then the intersection is all of $A^2$ and any $a \in S_x$ and $b \in S_y$ suffice.)

We can encode these constraints in a matrix. For two columns $x, y < p$, we define the $k \times k$ constraint matrix $M^{xy}$ such that for all $a, b \in A$, $M^{xy}(a, b) = *$ if for every pair of proximal positions $i, j$ in columns $x, y$, respectively, $(a, b) \in +(w(i), w(j))$, and $M^{xy}(a, b)$ remains empty otherwise. More specifically, $M^{xy}$ gives rise to a subset of $A^2$, which is the set of $(a, b)$'s such that $M^{xy}(a, b) = *$ or such that

$$(a, b) \in \bigcap_{0<|i-j|\leq d, i\equiv x \bmod p, j\equiv y \bmod p} +(w(i), w(j))$$

where the number of crosses that get intersected is the number of distinct ordered pairs $(w(i), w(j))$ where $i, j$ are proximal positions in columns $x, y$, respectively. Note that, trivially, $M^{xy}$ is the transpose of $M^{yx}$, and that $\omega \in \prod_{x=0}^{p-1} S_x$ induces a $d$-valid partial word contained in $w$ if and only if for every $x, y \in \{0, \ldots, p-1\}$, $M^{xy}(\omega(x), \omega(y)) = *$.

It turns out that the constraint matrices can be classified into a few simple forms, which enables us to compute and read them very quickly. The following theorem describes these forms.

**Theorem 1.** *Any subset of $A^2$ formed by intersecting crosses, and thus the constraint matrices for partial word $w$, can be represented by either:*

1. *FULL: the universe $A^2$;*
2. *CROSS$(a_0, b_0)$: a cross $+(a_0, b_0)$;*
3. *ROW$(a_0)$: a row $\{(a_0, b) \mid b \in A\}$;*
4. *COL$(b_0)$: a column $\{(a, b_0) \mid a \in A\}$;*
5. *TWO$((a_1, b_1), (a_2, b_2))$: a set of two points $(a_1, b_1)$ and $(a_2, b_2)$ in neither the same row nor column;*
6. *ONE$(a_0, b_0)$: a singleton set $\{(a_0, b_0)\}$;*
7. *NULL: the null set $\emptyset$.*

*Proof.* Let $T = \bigcap_{s=1}^{m} +(a_s, b_s)$. We proceed by induction on $m$, the number of crosses that are intersected. If $m = 0$, then $T = A^2$ is FULL. The form FULL is only possible for $m = 0$. If $m = 1$, then $T = +(a_1, b_1)$ is CROSS$(a_1, b_1)$. Now suppose that $m > 1$ and let $T' = \bigcap_{s=1}^{m-1} +(a_s, b_s)$. We consider what happens when we intersect $+(a_m, b_m)$ with $T'$, for $T'$ in each of the above forms.

Let $T' = \text{CROSS}(a_0, b_0)$. If $a_m = a_0$ and $b_m = b_0$, then $T' = +(a_m, b_m)$, so $T = T'$. If $a_m = a_0$ and $b_m \neq b_0$, then $T = \text{ROW}(a_0)$. If $a_m \neq a_0$

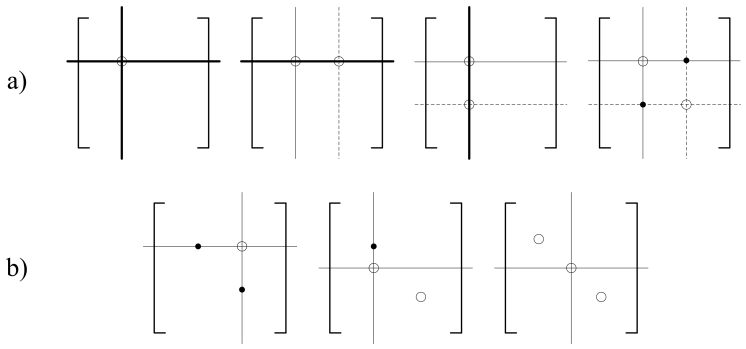


Figure 2: Intersection of different matrices

and $b_m = b_0$, then $T = \text{COL}(b_0)$. If $a_m \neq a_0$ and $b_m \neq b_0$, then $T = \text{TWO}((a_0, b_m), (a_m, b_0))$. Therefore, intersecting $+(a_m, b_m)$ with a CROSS matrix results in a CROSS, ROW, COL, or TWO matrix, as depicted in Figure 2. a).

If $T' = \text{ROW}(a_0)$ and $a_m = a_0$, then $T = T' \subset +(a_m, b_m)$. Otherwise, $T = \text{ONE}(a_0, b_m)$.

If $T' = \text{COL}(b_0)$ and $b_m = b_0$, then $T = T' \subset +(a_m, b_m)$. Otherwise, $T = \text{ONE}(a_m, b_0)$.

Let $T' = \text{TWO}((a, b), (a', b'))$. If $(a_m, b_m) = (a, b')$ or $(a_m, b_m) = (a', b)$, then $T = T' \subset +(a_m, b_m)$. If $a_m = a$ or $b_m = b$ then $(a, b) \in +(a_m, b_m)$, but $(a', b') \notin +(a_m, b_m)$, and so $T = \text{ONE}(a, b)$. Similarly, if $a_m = a'$ or $b_m = b'$ then $T = \text{ONE}(a', b')$. Finally, if $a_m \neq a$, $b_m \neq b$, $a_m \neq a'$ and $b_m \neq b'$, then $(a, b), (a', b') \notin +(a_m, b_m)$, so $T = \text{NULL}$. Therefore, intersecting $+(a_m, b_m)$ with a TWO matrix results in a TWO, ONE, or NULL matrix, as depicted in Figure 2. b).

If $T' = \text{ONE}(a_0, b_0)$ and $a_m = a_0$ or $b_m = b_0$, then $T' \subset +(a_m, b_m)$, so $T = T'$. Otherwise, $T = \text{NULL}$.

Finally, if $T' = \text{NULL}$, then $T = \text{NULL}$. □

Therefore, in practice, we store constraint matrices as objects that encode the form of the matrix (FULL, CROSS, TWO, etc.), and at most four characters to denote rows and columns (querying the positions of $*$'s in row $a_0$ of the object $\langle \text{TWO}, (a, b), (a', b') \rangle$ yields $b$ if $a_0 = a$, $b'$ if $a_0 = a'$, and NONE otherwise). These can be constructed and read in constant time.

Now, fix some variables that are shared by our algorithms: a table of constraint matrices, $M$; sets $F_{\text{ROW}}$, $F_{\text{ONE}}$, $F_{\text{TWO}}$ and $F_{\text{CROSS}}$, where

$F_{\text{FORM}}$ contains $(x, y)$ for which $M^{xy}$ is of form FORM; a list of letters $\omega$, where $\omega(x)$ is the letter chosen for column $x$.

The following remark and lemmas are useful in proving the validity of our algorithms.

**Remark 2.** *If columns $x, y$ are proximal, that is, $x, y$ contain proximal positions, then $0 < |x - y| \leq d$ or $0 < p - |x - y| \leq d$.*

**Lemma 1.** *If $0 \leq x, y < p$ with $0 < |x - y| \leq d$, then $M^{xy}$ is not FULL.*

*Proof.* The positions $x$ and $y$ in $w$ are proximal since $0 < |x - y| \leq d$. Therefore at least one cross (namely, that centered at $(w(x), w(y))$) is used in the creation of the matrix $M^{xy}$, so it cannot be FULL. □

Furthermore, it follows from Theorem 1 that the types of constraints that one column can exert on another are limited.

**Lemma 2.** *If two proximal columns $x, y$ with $0 \leq x < y < p$, each containing at least two different letters, are such that $M^{xy}$ is a CROSS matrix, then $|x - y| \geq \max\{p - d, d + 1\}$.*

*Proof.* By Remark 2, we have that $|x - y| \leq d$ or that $p - d \leq |x - y|$. Suppose that $|x - y| \leq d$, and let $y + sp$ be a position in column $y$, where $y \leq y + sp < n$. Thus, $x + sp$ is a position in column $x$, since $0 \leq x \leq x + sp < y + sp < n$. Furthermore, every position in column $y$ is proximal to some position in column $x$. Since $M^{xy}$ is a CROSS matrix, all ordered pairs $(w(i), w(j))$, for $i, j$ proximal positions in columns $x, y$, respectively, must be equal. Therefore all letters in column $y$ of $w$ are equal, a contradiction. Therefore $|x - y| > d$ and $|x - y| \geq p - d$, so $|x - y| \geq \max\{p - d, d + 1\}$. □

There exist even more restrictions regarding CROSS matrices.

**Lemma 3.** *Let $x_1, x_2, x_3$ be distinct (pairwise proximal) columns with at least two different letters each. If $M^{x_2x_3}$ and $M^{x_1x_3}$ are CROSS matrices, then $M^{x_1x_2}$ is neither a FULL nor a CROSS matrix.*

*Proof.* Let $q_1 = |x_2 - x_3|$, $q_2 = |x_1 - x_3|$ and $q_3 = |x_1 - x_2|$. By Lemma 2, we have that $q_1, q_2 \geq \max\{p - d, d + 1\} > \frac{p}{2}$. If $x_3$ laid between $x_1$ and $x_2$, we would have that $q_3 = q_1 + q_2 > \frac{p}{2} + \frac{p}{2} = p$, which is clearly a contradiction. Therefore $x_3$ lies to one side of both $x_1$ and $x_2$. But this means that $0 < q_3 = |q_1 - q_2| \leq d$, since both $q_1$ and $q_2$ lie in the interval $[p - d, p)$. Now by Lemmas 1 and 2, $M^{x_1x_2}$ can be neither a FULL nor a CROSS matrix. □

Henceforth, by *columns within* $d$ we mean columns $x, y$ such that $0 < |x-y| \leq d$ or $0 < p-|x-y| \leq d$. Any other pair of columns are necessarily related by a FULL constraint matrix and therefore can be ignored.

**Corollary 1.** *The forms (as per Theorem 1) of all the non-FULL constraint matrices for $w$ can be determined in $O(nd)$ time.*

*Proof.* By Remark 2, there are at most $2pd$ ordered pairs $(x, y)$ corresponding to columns with proximal positions (columns $x, y$ within $d$), since there are $p$ choices for $x$ and at most $2d$ corresponding choices for $y$. Initialize all these $2pd$ potentially non-FULL matrices $M^{xy}$ to FULL matrices, and successively intersect them with crosses centered at pairs $(w(i), w(j))$, where $i = x + i'p$ and $j = y + j'p$ for some appropriate $i', j'$.

The number of ordered pairs $(i, j)$ with $0 \leq i, j < n$ and $0 < |i-j| \leq d$ is at most $2nd$. Each of these ordered pairs $(i, j)$ is used in the calculation of exactly one non-FULL constraint matrix, namely $M^{(i \bmod p)(j \bmod p)}$. Since each of the cross-intersecting operations takes constant time, the overall runtime for computing all the non-FULL matrices is $O(nd)$. □

**Example 1.** *Let us consider the full word*

$$w = acbbabcaaababbaaacbbabcaa$$

*We will eventually see if it is possible to replace letters by holes that are not 2-proximal, such that the obtained partial word is 8-periodic. First, we arrange the word in rows of length 8:*

$$\begin{array}{cccccccc} a & c & b & b & a & b & c & a \\ a & a & b & a & b & b & a & a \\ a & c & b & b & a & b & c & a \\ a & & & & & & & \end{array}$$

*Now, let us look at the forms of constraint matrices created by the intersections of crosses. We see that after initialization, the matrices $M^{01}$, $M^{06}$, $M^{23}$, $M^{24}$, $M^{56}$ are of form ROW; $M^{12}$, $M^{17}$, $M^{35}$, $M^{45}$, $M^{67}$ are of form COL; $M^{02}$, $M^{07}$, $M^{57}$ are of form CROSS; $M^{13}$, $M^{34}$, $M^{46}$ are of form TWO; and all the other matrices are of form FULL. For instance, $M^{13}$ is the intersection of the crosses $+(c, b)$ with $+(a, a)$, and so has the form $TWO(((a, b), (c, a))$.*

**Algorithm 1** Fill$(x, a)$

```
 1: initialize Q to be an empty queue accepting columns
 2: choose letter a for column x
 3: add x to Q
 4: while dequeue y from Q do
 5:     let b = ω(y)
 6:     for z a neighbor of y do
 7:         let row be the b row of the matrix M^{yz}
 8:         remove edges (y, z) and (z, y)
 9:         if row has all ∗'s then
10:             go to line 6
11:         else if row has exactly one ∗, say at position c then
12:             if column z is labelled with c then
13:                 go to line 6
14:             else if column z is unlabelled then
15:                 choose letter c for column z
16:                 add z to Q
17:                 go to line 6
18:         undo all recent labellings and edge removals
19:         return false
20: return true
```

### 3.2 Traversing the graph associated with the constraint matrices

Note that given two proximal columns $x$ and $y$, and a letter $a$ chosen for column $x$, there are either zero, one, or $|S_y|$ choices of a letter for column $y$ that do not conflict with the choice of letter $a$ for column $x$. This observation suggests an algorithm for labelling multiple columns. Let us now construct a directed graph $G$ that has vertex set $\{0, \ldots, p-1\}$ and edge set consisting of edges $(x, y)$ labelled by $M^{xy}$ when columns $x, y$ are within $d$.

**Theorem 2.** *For a column $x$ and a letter $a \in S_x$, Algorithm 1 correctly chooses letters for some additional columns such that, after the completion of this algorithm no undetermined column is constrained by an already determined column. Additionally, if the constraint matrices have already been computed, the running-time of Algorithm 1 is $O(m)$, where $m$ is the number of edges that are traversed.*

*Proof.* The problem of finding a choice of letters for the columns is equivalent to finding a labelling of the vertices of $G$, such that for every vertex $x$, the

label of $x$ is a letter $\omega(x)$ that occurs in column $x$ of $w$, and such that for every edge $(x, y)$, the $(\omega(x), \omega(y))$-entry of $M^{xy}$ is a $*$. If such a labelling exists, then it induces a $p$-periodic $d$-valid partial word contained in $w$, by replacing every non-$\omega(x)$ letter in each column $x$ with a hole.

The algorithm starts by assuming a labelling of vertex $x$ by the letter $a$, $a = \omega(x)$, and then performs a breadth-first search on the graph $G$, starting at $x$. This is implemented using a queue.

Suppose that a vertex $y$ has been marked by letter $b = \omega(y)$ and that we are now traversing the edge $(y, z)$. Then one of the following occurs:

1. The choice of the letter $b$ for labelling column $y$ does not influence the choice of a letter for labelling column $z$;

2. There is only one possible choice of letter for labelling column $z$ (and this either does or does not conflict with a previous labelling for $z$);

3. There is no possible choice of letter for labelling column $z$.

In Case 3, the algorithm immediately fails. In Case 1, the $b$ row of the constraint matrix $M^{yz}$ is filled with $*$'s, so $M^{yz}$ is ignored imposing no constraint at all. In Case 2, the $b$ row of the matrix $M^{yz}$ has only one $*$, say at position $c$, so $M^{yz}$ uniquely determines the letter $c$ for labelling column $z$. Either $\omega(z)$ was already set to $c$, or $\omega(z)$ is set to $c$ and column $z$ is added to the queue for later traversal, or $\omega(z)$ was already set to a different value $c'$, say, in which case the algorithm fails because there cannot be any labelling of the vertices of $G$ with $\omega(x) = a$ and $\omega(z) = c'$. In all cases, the edge $(y, z)$ and its opposite $(z, y)$ are marked as having been traversed, so that they are not going to be visited again.

In conclusion, an undetermined column is marked exactly when it is constrained by an already determined column, thus, ensuring that at the end no determined column constrains an undetermined column. This algorithm visits $m$ edges, no more than once each. On each edge, it performs a constant time operation. Thus, Algorithm 1 runs in $O(m)$ time.

Note that undoing all recent labellings and edge removals, while keeping the algorithm's runtime within $O(m)$, is solved in constant time by implementing data structures that could be "marked" in a particular state, and reset to this state later on. These data structures are used for the set of neighbors of vertices, the sets $F_{\mathrm{FORM}}$ (of edges of each form), and the set of labelled vertices. While, all the $F_{\mathrm{FORM}}$'s and labellings can be reset in constant time, the vertex neighbor sets can be reset in $O(l)$ time, where $l$ is the number of vertices visited during this run of the algorithm. Since the

number of vertices visited is less than the number of edges visited, $l < m$, the overall algorithm runs in $O(m)$ time. □

**Example 2.** *Referring to Example 1, let us now run Algorithm 1 for column* $x = 1$ *and letter* $c$*, that is, Fill*$(1, c)$*. For neighbor* $0$*, the* $c$ *row of* $M^{10}$*, which has form COL*$(a)$ *being the intersection of the crosses* $+(c, a)$ *with* $+(a, a)$*, gives a unique* $*$ *at position* $a$*, and so column* $0$ *gets labelled with* $a$*. We process similarly the neighbors* $2, 3$ *and* $7$*, giving respectively the labellings* $b, a$ *and* $a$*. Now, dequeueing* $0$ *from the queue, gives no new labelling, the same being true for* $2$*. So the next member to be dequeued is* $3$*, which gives labellings* $a, b$ *for columns* $4, 5$*, respectively. Continuing like this, all labellings of vertices that follow from a breadth-first search starting at* $1$ *with label* $c$*, are* $a, c, b, a, a, b, a, a$ *for columns* $0, 1, 2, 3, 4, 5, 6, 7$ *respectively. We conclude that Fill*$(1, c)$ *returns true.*

The next lemma helps us prove that we never need to run Algorithm 1 ("Fill$(x, a)$") on a vertex more than twice.

**Lemma 4.** *Suppose that* $x$ *and* $y$ *are vertices of* $G$ *such that* $M^{xy} = TWO$ $((a, b), (a', b'))$*, Fill*$(x, a)$ *returns true, and* $\omega \in \prod_{z=0}^{p-1} S_z$ *induces a* $d$*-valid partial word* $u$ *with* $\omega(x) = a'$*. Then, there exists a choice* $\omega'$ *of letters for the columns, that induces a* $d$*-valid partial word with* $\omega'(x) = a$*.*

*Proof.* Let $T$ be the set of vertices of $G$ that are labelled by Fill$(x, a)$, and $Q$ be the labelling of $T$. For every vertex $x$ of $G$, let $\omega'(x) = Q(x)$ if $x \in T$ and $\omega'(x) = \omega(x)$ otherwise. Since the labelling $Q$ of $T$ was generated by Fill$(x, a)$, we know that no letter choice for a vertex outside $T$ is constrained by any of the letter choices specified in $Q$. Furthermore, since $\omega$ induced a $d$-valid partial word, we know that no constraint matrix is violated by two letter choices in $\omega$. Therefore the letter choices in $\omega'$ do not violate any constraint matrices, so $\omega'$ induces a $d$-valid partial word. Also, clearly $\omega'(x) = a$, so we have our result. □

Algorithm 2 traverses all edges corresponding to non-FULL matrices and finds a consistent labelling of the vertices of $G$ if any exists.

**Theorem 3.** *Given a full word* $w$ *of length* $n$*, and two positive integers* $p, d$ *both less than* $n$*, Algorithm 2 returns a* $d$*-valid* $p$*-periodic partial word contained in* $w$*, unless no such word exists. The running-time of the algorithm is* $O(nd)$*.*

**Algorithm 2** Traversing the entire graph

initialize matrices
**for** $(x, y)$ columns within $d$ **do**
    **if** $M^{xy} = \text{NULL}$ **then**
        **return** false
    add $(x, y)$ to $F_{\text{FORM}}$
**for** column $x$ **do**
    **if** $|S_x| = 1$ **then**
        Fill$(x, w(x))$
**while** exists $(x, y)$ with $M^{xy}$ of form ROW$(a)$, in $F_{\text{ROW}}$ **do**
    if not Fill$(x, a)$ then return false
**while** exists $(x, y)$ with $M^{xy}$ of form ONE$(a, b)$, in $F_{\text{ONE}}$ **do**
    if not Fill$(x, a)$ then return false
**while** exists $(x, y)$ with $M^{xy}$ of form TWO$((a, b), (a', b'))$, in $F_{\text{TWO}}$ **do**
    if not Fill$(x, a)$ and not Fill$(x, a')$ then return false
**for** column $x$ **do**
    **if** column $x$ is unlabelled **then**
        choose $w(x)$ for column $x$
**for** $i$ from 0 to $n - 1$ **do**
    let $u(i) = w(i)$ if $w(i) = \omega(i \bmod p)$ and $u(i) = \diamond$ otherwise
**return** $u$

*Proof.* If there is a NULL matrix $M^{xy}$ with columns $x, y$ within $d$, then no consistent labelling of the vertices exists, so the algorithm fails. If any column in $w$ has all letters equal, then that letter must be assigned for the column, and we run Fill$(x, w(x))$. There can only be one consistent labelling of vertices if it succeeds (note that the determination of whether a column has only one character can be performed in $O(\frac{n}{p})$ time, and thus, it can be performed for all columns in $O(n)$ time). Similarly, if there is a ROW or a ONE matrix $M^{xy}$ with a $*$ in row $a$, then $a$ must be chosen for column $x$. We run Fill$(x, a)$, and it must succeed for there to be a consistent labelling of the vertices of $G$.

If there is a matrix $M^{xy}$ of the form TWO$((a, b), (a', b'))$, then we know that any consistent labelling of the vertices of $G$ must have column $x$ labelled with either $a$ or $a'$. But by Lemma 4, if some consistent labelling of $G$ exists and Fill$(x, a)$ returns true, then there exists a consistent labelling of $G$ that agrees on all choices of letters made by Fill$(x, a)$. Therefore in this case we can simply continue. Otherwise we try Fill$(x, a')$. If this fails, then the

algorithm returns false.

At this point in the algorithm, any unlabelled vertices $x, y$ (associated with an edge $(x, y)$) are related by either a FULL or CROSS matrix, since all other forms of matrices have already been taken into account. Consider a graph $G'$ with the so-far unlabelled vertices of $G$ as the vertex set, and an edge between $x$ and $y$ if and only if $M^{xy}$ is a CROSS matrix. We can satisfy all remaining constraints (the CROSS matrices) by considering every connected component of $G'$ separately. But, by Lemma 3, this graph has no connected components of size greater than two (since only crosses are left, connecting more than two of them falls in Lemma 3).

We claim that we can label any remaining vertex $x$ with $w(x)$ (the first letter appearing in column $x$) without introducing any new contradictions. This is clearly true for any isolated vertex in $G'$, since these are unconstrained. Now consider $x, y$ vertices in $G'$ related by CROSS$(a, b)$. Every proximal pair of positions $i, j$ in columns $x, y$ must have $w(i) = a$ and $w(j) = b$. But between any two columns that have proximal pairs, at least one of them has its first (top) position proximal to some position in the other column. Therefore $w(x) = a$ or $w(y) = b$ (or both). Therefore these choices satisfy the constraint matrix. If the algorithm reaches this step, then there exists a $p$-periodic $d$-valid partial word contained in $w$, namely the one induced by $\omega$.

Each matrix is visited at most twice (this worst case scenario is achieved precisely if the edge is examined twice in the loop starting on line 13). There are at most $2pd$ matrices in question, and analyzing a row of a matrix takes constant time. Thus, the running-time is $O(pd)$, plus the running-time of checking which columns are uniform, $O(n)$, plus the running-time of constructing the constraint matrices, $O(nd)$ by Corollary 1. Therefore, the total running-time of Algorithm 2 is $O(nd)$. □

**Example 3.** *Referring to Examples 1 and 2, let us now run Algorithm 2 on $w$. The forms of all non-NULL constraint matrices have already been determined. Then run Fill$(0, w(0))$, Fill$(2, w(2))$, Fill$(5, w(5))$ and Fill$(7, w(7))$ which all return true. They label columns $0, 2, 5, 7$ with $a, b, b, a$ respectively. Eventually, the edge $(1, 3)$ gets considered with matrix $M^{13}$ that has form TWO$((a, b), (c, a))$. This determines a choice of $a$ or $c$ for labelling column 1. As we saw in Example 2, Fill$(1, c)$ returns true, labelling the rest of the vertices as described there. Note that for the other TWO matrices, $M^{34}$ and $M^{46}$, both Fill$(3, a)$ and Fill$(4, a)$ return true. The algorithm ends correctly, outputting the 2-valid 8-periodic partial word acb⋄ab⋄aa⋄ba⋄baaacb⋄ab⋄aa contained in $w$:*

$$
\begin{array}{cccccccc}
a & c & b & \diamond & a & b & \diamond & a \\
a & \diamond & b & a & \diamond & b & a & a \\
a & c & b & \diamond & a & b & \diamond & a \\
a & & & & & & &
\end{array}
$$

### 3.3 Dealing with squares

In the case where $n = 2p$, there are simple criteria for finding a square contained in $w$. More specifically, the problem can be reduced to the 2-coloring of a special graph $G$. Let each non-uniform column be a vertex, that is, a column containing two different letters. Put an edge between columns $x < y$ if $y - x \leq d$, i.e. $x$ is proximal to $y$ and $x + p$ is proximal to $y + p$. Constraints imposed by columns where $x + p$ is proximal to $y$ will be taken into account later. We deal here only with the case when $2(d+1) \leq p$, because it simplifies these additional constraints. The remaining cases are not difficult.

For each vertex $x$, if $x$ is colored blue then we choose letter $w(x)$ for column $x$ and thus, replace position $x + p$ of $w$ with a hole. If we color $x$ red then we choose letter $w(x + p)$ for column $x$ and replace position $x$ of $w$ with a hole. Note that if we color $x$ blue then all its neighbors must be colored red, and vice versa.

**Remark 3.** *Consider three non-uniform columns $x, y, z$ where $x < y < z$ and $z - x \leq d$. Then these form a triangle, so $G$ is not 2-colorable.*

If no such triple exists, every column $x$ has at most one neighbor less than $x$ and at most one neighbor greater than $x$. Therefore, $G$ must be a union of disjoint chains. Therefore $G$ has a valid coloring. Furthermore, we can color each section by simply alternating between the two colors.

**Theorem 4.** *A full word $w$* cannot *be made into a square by a $d$-valid insertion of holes if and only if:*

1. *there exist three non-uniform columns $x < y < z$, such that $z - x \leq d$ or*

2. *there exist four non-uniform columns $x < y < z < w$, such that $x + p$ is proximal to $z$, $x + p$ is proximal to $w$, $y + p$ is proximal to $w$, and either $y + p$ is proximal to $z$ or $G$ is an odd connected chain.*

*Proof.* Condition 1 follows directly from Remark 3. Thus, assume Condition 1 does not hold.

Suppose that $y, z$ are columns such that $y+p$ is proximal to $z$. A coloring of these two columns is invalid if and only if it corresponds to replacing $y+p$ and $z$ with holes. In other words, we cannot color vertex $z$ red and vertex $y$ blue. Now, it must be the case that $0 \leq y < d$ and $p - d \leq z < p$. So let $S$ be the set of columns $y$ such that $0 \leq y < d$ and $y + p$ is proximal to $z$ for some $p - d \leq z < p$. Let $T$ be the set of columns $z$ such that $p - d \leq z < p$ and $y + p$ is proximal to $z$ for some $0 \leq y < d$. Any two columns in $S$ are within $d$, so Remark 3 tells us that $S$ contains at most 2 columns. Similarly, $T$ contains at most 2 columns.

If $S$ contains only one column, color it red. Then the columns in $T$ are no longer constrained by the one in $S$, so we can easily 2-color the rest of the graph. Similarly, if $T$ contains only one column, color it blue. Again this takes care of the special constraints, so there is a valid coloring.

Suppose $S$ and $T$ are of size two. Say columns $x < y$ are in $S$ and columns $z < w$ are in $T$. This means that $x + p$ is proximal to $z$, $x + p$ is proximal to $w$, and $y + p$ is proximal to $w$. Note that since the two columns in $S$ (respectively, $T$) are within $d$, they must be colored different colors. So one column in $S$ must be blue and one column in $T$ must be red. But if $y + p$ is proximal to $z$, all columns in $S$ have an element proximal to all columns in $T$, so there is no valid coloring. If $y + p$ is not proximal to $z$, we must color $x$ and $z$ red and $y$ and $w$ blue. From this we can get a valid coloring unless $G$ is connected and of odd length. □

Say columns $i$ through $i + d - 1$ are uniform. If $x < i$ and $y \geq i + d$, then $y - x > d$. Thus, $d$ consecutive uniform columns partition $G$. Furthermore, if we have only $d - 1$ consecutive columns, then the columns on either side have difference $d$. So it takes greater than or equal to $d$ consecutive uniform columns to partition $G$. If follows that $G$ is odd and connected if and only if $w$ has an odd number of non-uniform columns and no $d$ consecutive uniform columns. Thus, we do not have to construct $G$ to apply the criteria from Theorem 4, we need only consider which columns are non-uniform.

## 4 Short factors in the images of morphisms

We now discuss a technique for finding all length $n$ factors of an infinite word, which is the image of a morphism. The lemmata of this section is useful in proving the main result of Section 5.

Let $\beta : A^* \to A^*$ be a non-erasing prolongable morphism on $z_0 \in A$. For $m \geq 0$, set $z_{m+1} = \beta(z_m)$, and let $w = \lim_{m \to \infty} z_m$ be the fixed point of $\beta$.

Also, let $F_m(y)$ denote the set of length $m$ factors of word $y$. Since $z_m$ is a proper prefix of $z_{m+1}$, for any $n \geq 1$:

$$F_n(z_0) \subset F_n(z_1) \subset F_n(z_2) \subset \cdots \subset \bigcup_{m \geq 0} F_n(z_m) = F_n(w)$$

But since $F_n(w)$ is finite, having at most $|A|^n$ elements, using the pigeonhole principle, the chain must become constant after finitely many steps.

**Lemma 5.** *Let $m \geq 0, n \geq 1$ be such that $\emptyset \neq F_n(z_m) = F_n(z_{m+1})$. Then $F_n(z_m) = F_n(w)$.*

*Proof.* Since $F_n(z_m)$ is nonempty, $z_m$ must have length at least $n$. We show by induction on $i$ that $F_n(z_{m+i}) = F_n(z_{m+i+1})$ for all $i \geq 0$. The basis $i = 0$ holds by assumption, so assume that $F_n(z_{m+i}) = F_n(z_{m+i+1})$. Since $z_{m+i}$ and $z_{m+i+1}$ have $z_m$ as a proper prefix, they must have length at least $n$. Thus,

$$F_n(z_{m+i+2}) = F_n(\beta(z_{m+i+1})) = \bigcup_{u \in F_n(z_{m+i+1})} F_n(\beta(u)) =$$

$$= \bigcup_{u \in F_n(z_{m+i})} F_n(\beta(u)) = F_n(\beta(z_{m+i})) = F_n(z_{m+i+1})$$

Therefore, the union $F_n(w)$ of all the $F_n(z_j)$'s is equal to $F_n(z_m)$. □

Now, set $q = \min\{|\beta(a)| \mid a \in A\}$ and $s = \max\{|\beta(a)| \mid a \in A\}$.

**Remark 4.** *Let $v$ be a factor of length $n$ of some $\beta(y)$, where $y$ has length at least $n'$. Suppose that $q(n'-1) \leq n-1$. Then there is a length $n'$ factor $u$ of $y$ such that $v$ is a factor of $\beta(u)$. Consequently,*

$$F_n(\beta(y)) = \bigcup_{u \in F_{n'}(y)} F_n(\beta(u))$$

**Lemma 6.** *Suppose that $q \geq 2$ and $F_2(z_m) = F_2(w)$. Then for all $i \geq 0$, $F_{q^i+1}(z_{m+i}) = F_{q^i+1}(w)$.*

*Proof.* In other words, our assumption says that $\beta$ maps every letter of the alphabet to a word of length at least two, and all length two factors of $w$ are factors of $z_m$. We proceed by induction on $i$. If $i = 0$, then $q^i + 1 = 2$ and since $F_2(z_m) = F_2(w)$ by assumption, we have our base case.

Now suppose that $i \geq 0$ and that $F_{q^i+1}(z_{m+i}) = F_{q^i+1}(w)$. Since $F_{q^i+1}(w)$ is nonempty and $q((q^i+1)-1) = (q^{i+1}+1)-1$, by Remark 4 we get that

$$F_{q^{i+1}+1}(w) = F_{q^{i+1}+1}(\beta(w)) = \bigcup_{u \in F_{q^i+1}(w)} F_{q^{i+1}+1}(\beta(u)) =$$

$$\bigcup_{u \in F_{q^i+1}(z_{m+i})} F_{q^{i+1}+1}(\beta(u)) = F_{q^{i+1}+1}(\beta(z_{m+i})) = F_{q^{i+1}+1}(z_{m+i+1})$$

It follows by induction that for all $i \geq 0$, $F_{q^i+1}(z_{m+i}) = F_{q^i+1}(w)$. □

The following lemma holds trivially.

**Lemma 7.** *The prefix of length $n$ of $w$ can be computed in $O(n)$ time.*

**Lemma 8.** *The set $F_n(w)$ can be computed in $O(n^{\log_q s})$ time.*

*Proof.* Since

$$|z_{m+\lceil \log_q(n-1) \rceil}| \leq s^{m+\lceil \log_q(n-1) \rceil} \leq s^{m+1} s^{\log_q(n-1)} = s^{m+1}(n-1)^{\log_q s}$$

we need $O(n^{\log_q s})$ time to compute the prefix of length $|z_{m+\lceil \log_q(n-1) \rceil}|$ of $w$ by Lemma 7 (we keep adding the value of the morphism at the end of the word until the length is reached). Let us recall that $m$ is fixed and can be bounded (the upper bound $|A|^2+1$ is not dependent on the morphism). Furthermore, by Lemma 6, we can identify all factors of length $n$ using a window of length $n$ and going once through the prefix $z_{m+\lceil \log_q(n-1) \rceil}$ (a hashset can be used to store these elements). □

Hence, for any $n \geq 2$, $z_{m+\lceil \log_q(n-1) \rceil}$ has all length $n$ factors of $w$, and this set can be computed in polynomial time. Furthermore, if $\beta$ is a uniform morphism, we have $q = s$ and $F_n(w)$ is computable in $O(n)$ time. Note that in some cases we can discard the requirement $q \geq 2$, by taking a higher iteration of the morphism (for $\mu : a \mapsto abc$, $b \mapsto ac$, $c \mapsto b$, the square $\mu^2 : a \mapsto abcacb$, $b \mapsto abcb$, $c \mapsto ac$, can be used in the above lemmas, since it generates the same fixed point).

## 5 An overlap-free word over a size five alphabet

Note that the definition of weak overlap of Section 2 generalizes the overlap definitions used in [3] and [5], since here a factor is considered to be an

overlap of length $2p+1$ if it has $p$ as a weak period, while in [3, 5], the factor had to have a strong period $p$. In this section, we generate an infinite full word over a 5-letter alphabet, which remains weakly overlap-free after any 2-valid insertion of holes.

Define a morphism $\gamma : \{a,b,c,d\}^* \to \{a,b,c,d\}^*$ with $\gamma(a) = ad$, $\gamma(b) = ac$, $\gamma(c) = cb$, and $\gamma(d) = ca$. Since $a$ is a prefix of $\gamma(a)$, $\gamma$ is prolongable. Thus, we define the fixed point of $\gamma$, $\Gamma = \lim_{n\to\infty} \gamma^n(a)$.

Let us now consider some properties of $\Gamma$.

**Remark 5.** *Both $\gamma^3(a) = adcacbad$ and $\gamma^4(a) = adcacbadcbacadca$ have only $ac, ad, ba, ca, cb$ and $dc$ as their length two factors. Thus, by Lemma 5, these are the only length two factors of $\Gamma$.*

**Lemma 9.** *The infinite full word $\Gamma$ is square-free.*

*Proof.* It suffices to show that every $\gamma^n(a)$ is square-free. Clearly $\gamma^0(a) = \varepsilon$ is square-free. Now let $n \geq 0$ and assume that $\gamma^n(a)$ is square-free. Suppose, for contradiction, that $\gamma^{n+1}(a)$ has a square factor of length $2p$ starting at position $i$, with $p$ minimal. Since the letters $b$ and $d$ appear only at odd positions of $\gamma^{n+1}(a)$, if $p$ is odd, the factor would be in $\{a,c\}^*$. Following the construction of $\gamma$ and Remark 5, there exists no factor in $\{a,c\}^*$ of length greater than three, and, moreover, all those of length smaller than three are not squares.

Therefore, $p$ must be even. If $i$, the position the square starts at, is even, it follows that the square represents the image of a word $xx$ through $\gamma$. Thus, from $\gamma^{n+1}(a) = \gamma(\gamma^n(a))$ it follows that $\gamma^n(a)$ contains a square, contradicting the initial assumption. Hence, $i$ is odd. Since, $\gamma(f)$ ends in a different letter for all $f \in \{a,b,c,d\}$, it follows that we have a factor of $\gamma^{n+1}(a)$ that is a square starting at position $i-1$, which is an even position. Following the previous reasoning we again reach a contradiction. □

Now let $\delta : \{a,b,c,d\}^* \to \{f,g,h,i,j\}^*$ be a morphism defined by $\delta(a) = fgifh$, $\delta(b) = fghij$, $\delta(c) = jigjh$, and $\delta(d) = jihgf$. We claim that $\delta(\Gamma)$ is overlap-free after an arbitrary 2-valid insertion of holes.

**Lemma 10.** *There are no factors of $\delta(\Gamma)$ of length $\leq 17$ that can be turned into weak overlaps by any 2-valid insertion of holes.*

*Proof.* Using Lemmas 6 and 8, one can find quite fast, within the 5th iteration of the morphism $\gamma$, all factors of length 4. Applying $\delta$ to all of these, gives us all the possibilities for factors of length 17 that we need to check. Using a variant of Algorithm 2, we can check that none of these factors contains a 2-valid weakly-$p$-periodic word. □

Let us recall the following result.

**Remark 6.** *[3] Full words $t = t_0t_1t_2$ and $s = s_0s_1s_2$ contain compatible 2-valid partial words if and only if for some $i$, $t_i = s_i$.*

**Lemma 11.** *In $\delta(\Gamma)$, any two length seven sequences starting with the same character contain at least three consecutive mismatches if they are not identical.*

*Proof.* According to Remark 5 the only length two factors of $\Gamma$ are $ac$, $ad$, $ba$, $ca$, $cb$ and $dc$. We prove the lemma for sequences of length seven starting with letter $f$, the other cases being similar. If a sequence starts with $f$, then it must be either $fgifhji$, a prefix of both $\delta(ac)$ and $\delta(ad)$, $fghijfg$, prefix of $\delta(ba)$, $fjigjhf$, first factor starting with $f$ in both $\delta(dca)$ and $\delta(dcb)$, or, $fhjigjh$ and $fhjihgf$, suffixes of $\delta(ac)$ and $\delta(ad)$. It is easy to check that each two of these blocks contain three consecutive mismatches once aligned. □

**Lemma 12.** *No factor of $\delta(\Gamma)$ of length $2p + 1 > 17$ can be turned into a weak overlap by a 2-valid insertion of holes.*

*Proof.* Assume that there exists $a_0v_0a_1v_1a_2$, a factor that can be transformed into a weak overlap after a 2-valid insertion of holes, where each $v_i$ is a word of length $p - 1$ and the $a_j$'s are letters. If the second letters of $a_0v_0$ and $a_1v_1$ are equal, then we get a contradiction by Lemma 11 (here no two corresponding length seven factors in each half starting with the same character can be identical). If the two positions do not match, following Remark 6 it must be that either the first or the third positions must match. Using the same technique we get a contradiction in both these cases. Therefore, no factor of $\delta(\Gamma)$ of length $2p + 1 > 17$ can be turned into a weak overlap. □

**Theorem 5.** *The infinite word $\delta(\Gamma)$ over a 5-letter alphabet is weakly overlap-free after an arbitrary 2-valid insertion of holes.*

*Proof.* This follows directly from Lemmas 10 and 12. □

Since strong periodicity implies weak periodicity, the theorem answers an open problem of [3] regarding how large an alphabet must be to create an infinite word that is strongly overlap-free despite arbitrary 2-valid insertions of holes.

**Corollary 2.** *The infinite word $\delta(\Gamma)$ over a 5-letter alphabet is strongly overlap-free after a 2-valid arbitrary insertion of holes.*

Please note that the lower bound of five letters presented in [3] stands, since for alphabets of size smaller than five, all infinite words contain factors of length $2p+1$ that are strongly $p$-periodic, and therefore weakly $p$-periodic. Also note that the use of the terms weakly and strongly overlap-free word comes from the concepts of weak- and strong-periodicity (when looking at overlaps from this point of view, the terminology comes naturally).

# 6 Conclusion

Our $O(nd)$ time algorithm of Section 3, Algorithm 2, whenever it outputs a $d$-valid $p$-periodic partial word contained in a given full word $w$ of length $n$, actually outputs a *maximal* partial word satisfying these conditions. Indeed, let $U$ be the set of $p$-periodic (but not necessarily $d$-valid) partial words contained in $w$. We say $x \in U$ is maximal, if $y \in U$ and $x \subset y$ imply that $y = x$. Denoting by $U_{\max}$ the set of maximal partial words in $U$, assume there exists a $d$-valid element, call it $v$, in $U$. Since $U$ is a finite set, $v$ must be contained in some maximal element $u \in U_{\max}$. Since $u \in U$, $u$ is $p$-periodic and is contained in $w$. Furthermore, since $v \subset u$, $u$ is $d$-valid. Therefore if there are any $d$-valid $p$-periodic partial words contained in $w$, there is one in $U_{\max}$.

There are other problems related to the one of finding $d$-valid, $p$-periodic partial words. Let us mention, for instance, the problem of finding "approximate repetitions" in words. Reference [9], in particular, shows how to find all approximate matchings between substrings of two given strings $u$ and $v$, that is, from a substring of $u$ we can, by substitution and deletion of letters, obtain a substring of $v$ such that the number of these operations is minimum; another problem is that of finding the occurrences of some given string $u$ in some given string $v$, such that there are at most $k$ mismatches between $u$ and the identified factors of $v$, in other words, the Hamming distance between $u$ and the identified factors of $v$ is at most $k$.

Our Algorithm 2 is fast, its time complexity not depending at all on the given period $p$. Recently, in [8], the authors got rid of the distance $d$ using some predefined data structures and using an approach quite different from ours.

The space complexity of Algorithm 2 is $O(\max(pd, n))$. More precisely, for each column $x$ we only need to check the $d$ columns that are related to it in order to create the crosses that are saved as triplets. All other crosses created with the help of column $x$ are automatically FULL. Obviously, the space needed cannot be less than linear.

Our construction in Section 5 gives a partial word over a five-letter alphabet having a very strong repetition-freeness property. It may end up being useful in several areas, Thue's overlap-free construction having found many uses. In addition to overlap-freeness, the fixed point of Thue's morphism has several nice properties. Among them we recall the Thue-Morse constant (obtained by interpreting the sequence as a concatenation of binary digits), a recurrence plot, the "odious" and "even" numbers, as well as its involvement in several interesting products [1].

A World Wide Web server interface has been established at

`www.uncg.edu/cmp/research/freeness2`

for automated use of a program that implements our $O(nd)$ time algorithm of Section 3, as well as another program that implements our polynomial time algorithm of Section 4.

## References


[1] J.-P. Allouche and J. Shallit. *Automatic Sequences: Theory, Applications, Generalizations*. Cambridge University Press, 2003.

[2] F. Blanchet-Sadri, R. Mercaş, A. Rashin, and E. Willett. An answer to a conjecture on overlaps in partial words using periodicity algorithms. In A.-H. Dediu, A. M. Ionescu, and C. Martín-Vide, editors, *LATA 2009, 3rd International Conference on Language and Automata Theory and Applications, Tarragona, Spain*, volume 5457 of *Lecture Notes in Computer Science*, pages 188–199, Berlin, Heidelberg, 2009. Springer-Verlag.

[3] F. Blanchet-Sadri, R. Mercaş, and G. Scott. A generalization of Thue freeness for partial words. *Theoretical Computer Science*, 410:793–800, 2009.

[4] V. Halava, T. Harju, and T. Kärki. Square-free partial words. *Information Processing Letters*, 108:290–292, 2008.

[5] V. Halava, T. Harju, T. Kärki, and P. Séébold. Overlap-freeness in infinite partial words. *Theoretical Computer Science*, 410:943–948, 2009.

[6] M. Lothaire. *Combinatorics on Words*. Cambridge University Press, Cambridge, 1997.

[7] F. Manea and R. Mercaş. Freeness of partial words. *Theoretical Computer Science*, 389:265–277, 2007.

[8] F. Manea, R. Mercaş, and C. Tiseanu. Periodicity algorithms for partial words. In F. Murlak and P. Sankowski, editors, *MFCS 2011, 36th International Symposium on Mathematical Foundations of Computer Science*, volume 6907 of *Lecture Notes in Computer Science*, pages 472–484, Berlin, Heidelberg, 2011. Springer-Verlag.

[9] J. P. Schmidt. All highest scoring paths in weighted grid graphs and their application to finding all approximate repeats in strings. *SIAM Journal on Computing*, 27:972–992, 1998.

[10] A. Thue. Über unendliche Zeichenreihen. *Norske Vid. Selsk. Skr. I, Mat. Nat. Kl. Christiana*, 7:1–22, 1906. (Reprinted in *Selected Mathematical Papers of Axel Thue*, T. Nagell, editor, Universitetsforlaget, Oslo, Norway (1977), pp. 139–158).

[11] A. Thue. Über die gegenseitige Lage gleicher Teile gewisser Zeichenreihen. *Norske Vid. Selsk. Skr. I, Mat. Nat. Kl. Christiana*, 1:1–67, 1912. (Reprinted in *Selected Mathematical Papers of Axel Thue*, T. Nagell, editor, Universitetsforlaget, Oslo, Norway (1977), pp. 413–478).