McMaster University

Advanced Optimization Laboratory

Title:

Computational Substantiation of the d-step Conjecture for Distinct Squares Revisited

Authors:

Frantisek Franek and Michael Liut

AdvOL-Report No. 2021/2

July 2021, Hamilton, Ontario, Canada

Computational Substantiation of the d-step Conjecture for Distinct Squares Revisited

Frantisek Franek
McMaster University, Hamilton, Ontario, Canada
email: franek@mcmaster.ca

Michael Liut
University of Toronto, Mississauga, Ontario, Canada
email: michael.liut@utoronto.ca

Abstract

The maximum number of distinct squares problem was introduced in 1998 by Fraenkel and Simspon. They provided a bound of $2n$ for a string of length n and conjectured that the bound should be at most n. Though there have been several improvements since, the conjecture is still unresolved. In 2011, Deza et al. introduced the d-step conjecture for the maximum number of distinct primitively-rooted squares: for a string of length n with d distinct symbols, the number of distinct squares is bounded by the value of $n-d$. In 2016, Deza et al. presented a framework for computer search for strings exhibiting the maximum number of distinct primitively-rooted squares. The framework was based on the d-step method and the main tool, the S-cover, allowed them to approximately double the length of strings that can be analyzed in comparison to the brute force. For instance, they were able to compute the values for binary strings up to length 70. We present a framework for computer search for counterexamples to the d-step conjecture. This change of focus, combined with additional novel analysis, allow us to constrain the search space to a larger degree, thus enabling a verification of the d-step conjecture to higher lengths. For instance, we have fully verified the d-step conjecture for all combinations of n and d such that $n-d \leq 24$ and for binary strings up to length 90. The computational efforts are still continuing. Since neither the maximum number of distinct squares conjecture nor the d-step conjecture can be resolved using a computer, the usefulness of our effort is twofold. Firstly, the primary

aspiration is that with the identification of sufficient constraints, the non-existence of counterexamples can be established analytically. Secondly, the verification of the conjectures for higher lengths acts indirectly as evidence of the validity of the conjectures, which indicates that effort should be directed towards proving the conjectures rather than disproving them.

Keywords: *string, square, root of square, primitively-rooted square, number of distinct squares, maximum number of distinct squares conjecture, d-step method, d-step conjecture*

1 Introduction

Counting distinct squares means counting each type of square only once, not the occurrences – a problem introduced by Fraenkel and Simpson, [9, 10]. They provided a bound of $2n$ for strings of length n and conjectured that it should be bounded by n. Their main idea was to count only the rightmost occurrences of squares, or what we call in this paper, the *rightmost squares*. That simplified the combinatorics of many squares starting at the same position to at most two, allowing them to bound the number of rightmost squares by $2n$. Their conjecture, known as the *maximum number of distinct squares conjecture* remains unresolved. In 2007, Ilie presented an asymptotic bound of $2n-\Theta(\log\ n)$, [11]. In 2015, Deza et al. in [8] presented a result on the bound for the maximum number of FS-double squares (i.e., two rightmost squares starting at the same position) $\frac{5}{6}n$, which in turn gives $\frac{11}{6}n \approx 1.83n$ bound for distinct squares. The work is a careful analysis of the combinatorial relationships of FS-double squares based on the pioneering work of Lam, [13]. In 2020, Thierry posted a preprint in arXiv in which he claims a bound of $1.5n$ based on the same techniques used in [8], however, as far as we know, the preprint has not been submitted for peer review, and there are some aspects of the proof that are not clear to us, see [15].

In 2011, Deza et al. introduced the d-step conjecture for the maximum number of distinct primitively-rooted squares: for a string of length n with d distinct symbols, the number of distinct primitively-rooted squares is bounded by the value of $n-d$, see [5] and an overview in [3]. The d-step conjecture also remains unresolved. In 2014, Janoska et al., [12], proposed a slightly stronger conjecture for binary strings – namely that the number of distinct squares in a binary string of length n is bounded by $\frac{2k-1}{2k+2}n$, where k is the number of occurrences of the symbol occurring the least number of times in the string. Since $k \leq \lfloor \frac{n}{2} \rfloor$, it follows that $\frac{2k-1}{2k+2}n \leq n-2$ when $n \geq 4$, and thus it is a slight strengthening of the d-step conjecture. They show several classes of binary strings for which their conjecture holds true. In 2015, Manea and Seki, [14], introduced the notion of square-density of a string as a ratio of the number of distinct squares and the length of the string. They showed that binary strings exhibit the largest square densities in the form that for any string over a ternary or higher alphabet, there is a binary string with a higher square-density. To that end, they presented a homomorphism that transforms a given string to a binary string with a higher square-density of significantly longer length. Since the lengths of the original string and its

homomorphic transformation are quite different, this result cannot be used for direct comparison of strings of the same length over different alphabets that is need to resolve the d-step conjecture. Moreover, the consequence of the work [14] is that if the maximum number of distinct squares conjecture is shown to hold for strings over a particular non-unary fixed alphabet, then it holds true for strings over any other no-unary fixed alphabet. In 2015, Blanchet-Sadri et al. established upper bounds for the number of primitively-rooted squares in partial words with holes, see [1]. Many researchers intuitively believe that a "heavy" concentration of double squares at the beginning of a string prolongs the string and is compensated by a "light" concentration of squares at the tail. In 2017, Blanchet-Sadri et al. investigated the density of distinct primitively-rooted squares in a string concluding a that a string starting with m consecutive FS-double squares must be at least $7m+3$ long, thus quantifying a part of the intuitive belief, see [2].

In 2016, Deza et al., [7], presented a framework for computer search for strings exhibiting a maximum number of distinct primitively-rooted squares. The main tool, the S-cover, allowed them to approximately double the length of strings that can be analyzed in comparison to the brute force approach. For instance, they were able to compute the values for binary strings up to length 70 and thus verify the d-step conjecture for binary strings up to length 70.

A naive approach to computing the maximum number of distinct squares for strings of length n is to generate all strings and compute the number of distinct squares in each while recording the maximum. The major cost of this approach is generating d^n many strings, however, with a bit of ingenuity you only need to generate a bit less than $d^{(n-d)}$. This is because the first d distinct symbols can be fixed as the number of distinct squares in a string does not change when the symbols in the string are permuted. Thus, it is important to constrain the generation of the string as early as possible, so the tail of the string does not need to be generated. Deza et al. achieved this by only generating the S-covered strings, reducing the number of strings generated to approximately $d^{\frac{n}{2}}$. A major focus of their paper was devoted to showing why the strings that have no S-cover do not need to be considered, and why many of the strings that may have an S-cover do not need to be considered. The emphasis of the computation was to identify the square-maximal strings and thus care was applied in making sure that only strings that cannot achieve a maximum were eliminated. For smaller lengths, all square-maximal strings were computed, while for longer strings in situations when the number of distinct squares was constrained, only a single square-maximal string was produced, see [4]. Since the strings produced cannot be independently verified to be square-maximal, the verification of the result is the computer code – if the code is correct, the strings are truly square-maximal.

We took the effort one step further, we are no longer searching for square-maximal candidates, instead we are searching for strings of length n with d distinct symbols that violate the d-step conjecture, i.e., the strings that have strictly more than $n-d$ distinct squares. This allows us to constrain the search space even more and thus verify the d-step conjecture to higher values. For instance, we have fully verified the d-step conjecture for all combinations of n and d such that

$n-d \leq 24$ and for binary strings up to length 90. The computational efforts are still continuing. Since the result of the computation is an empty set of counterexamples, the result cannot be independently verified; as with [7], the verification of the result is the computer program – if the program is correct, the set of counterexamples is truly empty. It is clear that the d-step conjecture cannot be validated by computer search. So, the question "why bother?" comes to mind. Well, the ultimate goal of our approach is to analytically establish that the set of counterexample strings is empty. The constraints for generation may be strengthened to the point that it would be viable to show analytically that such strings do not exist – for example, in Lemma 5 it is shown that a counterexample string cannot be a square. Meanwhile, it strengthens the empirical evidence that the conjecture is plausible and more effort should be directed towards proving it rather than disproving it.

2 Terminology and Notation

An integer range is denoted by $..$, i.e., $a..b = \{a, a+1, a+2, ..., b-2, b-1, b\}$. A bold font is reserved for strings; $\boldsymbol{x}$ identifies a string named $\boldsymbol{x}$, while x can be used for any other mathematical entity such as integer, or length, etc. For a string $\boldsymbol{x}$ of length n we use the array notation indexing from 1: thus $\boldsymbol{x}[1..n] = \boldsymbol{x}[1]\boldsymbol{x}[2]...\boldsymbol{x}[n-1]\boldsymbol{x}[n]$. The term (d,n)-*string* is used for a string of length n with d distinct symbols. For instance, *ababb* is a $(2,5)$-string. A *square* is a tandem repeat (concatenation) of the same string, e.g., $\boldsymbol{uu}$. A power of a string $\boldsymbol{u}$ means more than one concatenation of $\boldsymbol{u}$, e.g., $\boldsymbol{u}^2$ means $\boldsymbol{uu}$, $\boldsymbol{u}^3$ means $\boldsymbol{uuu}$, etc. If a string is not a power, it is said to be *primitive*. A *root* of a square $\boldsymbol{uu}$ is the string $\boldsymbol{u}$. A *primitively-rooted* square $\boldsymbol{uu}$ means that the root $\boldsymbol{u}$ is primitive. We identify a square in a string $\boldsymbol{x}[1..n]$ with a pair (a,b) of positions $1 \leq a < b \leq n$, where a is the starting position and b the ending position of the square, i.e., $\boldsymbol{x}[a..b]$ is the square. A square (a,b) is *rightmost*, if it is the rightmost occurrence of the square in $\boldsymbol{x}$. For a string $\boldsymbol{x}$, $\mathcal{A}_{\boldsymbol{x}}$ is the string's *alphabet*, i.e., the set of all symbols occurring in $\boldsymbol{x}$. If a symbol occurs only at one position in the string, it is referred to as a *singleton*. A string $\boldsymbol{x}$ is *singleton-free* if every symbol of its alphabet occurs in $\boldsymbol{x}$ at least twice. Symbol $\overleftarrow{\boldsymbol{x}}$ indicates the reversed string $\boldsymbol{x}$, i.e., $\overleftarrow{\boldsymbol{x}}[1..n] = \boldsymbol{x}[n]\boldsymbol{x}[n-1]...\boldsymbol{x}[2]\boldsymbol{x}[1]$. $|\boldsymbol{x}|$ indicate the *size* (*length*) of the string $\boldsymbol{x}$. If $\boldsymbol{x} = \boldsymbol{uvw}$, then $\boldsymbol{u}$ is a *prefix*, $\boldsymbol{w}$ is a *suffix*, and $\boldsymbol{v}$ is a *substring* or *factor* of $\boldsymbol{x}$. If $|\boldsymbol{u}| < |\boldsymbol{x}|$ we speak of a *proper prefix*, if $|\boldsymbol{w}| < |\boldsymbol{x}|$ we speak of a *proper suffix*, and if $|\boldsymbol{v}| < |\boldsymbol{v}|$ we speak of a *proper substring* or *proper factor*. The symbol $s(\boldsymbol{x})$ is used for the number of rightmost squares of the string $\boldsymbol{x}$. The symbol $\sigma_d(n)$ is used for the maximum number of rightmost primitively-rooted squares over all strings of length n with d distinct symbols, i.e., $\sigma_d(n) = \max\{s(\boldsymbol{x}) \mid \boldsymbol{x}$ *is a* (d,n)-*string*$\}$. For a string $\boldsymbol{x}$ of length n, $B_{\boldsymbol{x}}(i,j)$, $1 \leq i \leq j \leq n$, is defined as the number of rightmost primitively-rooted squares that start in the interval $i..j$, while $E_{\boldsymbol{x}}(i,j)$, $1 \leq i \leq j \leq n$, is defined as the number of rightmost primitively-rooted squares that end in the interval $i..j$. Note that $B_{\boldsymbol{x}}(1,i) - E_{\boldsymbol{x}}(1,k)$ is the number of rightmost primitively-rooted squares that start in $1..i$ and end in $k+1..n$. If it is clear from the context, we drop the subscript $\boldsymbol{x}$ for $B_{\boldsymbol{x}}$ and $E_{\boldsymbol{x}}$.

3 Basic Facts

The following lemma indicates that it makes sense to try using induction over $n-d$ rather than over n. The columns in the $(n, d-n)$-table, see [4], are completely filled in up to $n-d = 19$. In other words, we have a base case for induction over $n-d$.

Lemma 1. *Let $\boldsymbol{x}$ be a singleton-free (d,n)-string, $1 \le d < n$ and let d_1 be the number of distinct symbols in a non-empty proper prefix (resp. suffix) of $\boldsymbol{x}$ of length n_1. Then, $n_1-d_1 < n-d$.*

Proof. Consider a non-empty proper prefix $\boldsymbol{x}[1\,..\,n_1]$ for some $1 \le n_1 < n$. First we consider the extreme case $n_1 = 1$: then $d_1 = 1$ and so $n_1-d_1 = 1-1 = 0 < n-d$. Thus, we can assume that $1 < n_1 < n$.
Let $n_2 = n-(n_1+1)+1 = n-n_1$, i.e., the length of $\boldsymbol{x}[n_1+1\,..\,n]$, and let d_2 be the number of distinct symbols in $\boldsymbol{x}[n_1+1\,..\,n]$. If $d_1 = d$, then $n_1-d_1 = n_1-d < n-d$. So we can assume that $1 \le d_1 < d$.
Let $r = d-d_1$. Then $r > 0$ and $1 \le d_1, d_2 \le d$ and $d_1+d_2 \ge d$. Since r is the number of distinct symbols from $\boldsymbol{x}$ that are missing in $\boldsymbol{x}[1\,..\,n_1]$, they must occur in $\boldsymbol{x}[n_1+1\,..\,n]$ and hence $d_2 \ge r$.
First let us show that $n_2 \ge r$: if $n_2 < r$, then $d_2 < r$ as $d_2 \le n_2$. Hence $d_1+d_2 < d_1+r = d$, a contradiction.
Second let us show that in fact $n_2 > r$: if $n_2 = r$, then $d_2 \le n_2 = r$ and also $d_2 \ge r$, so $d_2 = r = n_2$. Thus, the number of distinct symbols in $\boldsymbol{x}[n_1+1\,..\,n]$ equals the length, i.e., they are all singletons in $\boldsymbol{x}[n_1+1\,..\,n]$, and since $\boldsymbol{x}[n_1+1\,..\,n]$ must contain the r symbols missing in $\boldsymbol{x}[1\,..\,n_1]$, they must be singletons in $\boldsymbol{x}$ as well, a contradiction.
Thus, $n_2 > r$ and so $n_1 = n-n_2 < n-r$, giving $n_1-d_1 < n-r-d_1 = n-d$.
For a non-empty proper suffix $\boldsymbol{x}[j\,..\,n]$, $1 < j \le n$, let $n_1 = n-j+1$, i.e., the length of the suffix. Let d_1 be the number of distinct symbols in $\boldsymbol{x}[j\,..\,n]$. Consider the string $\overleftarrow{\boldsymbol{x}}$ which is the string $\boldsymbol{x}$ reversed. Then $\overleftarrow{\boldsymbol{x}}[1\,..\,n_1]$ is a non-empty proper prefix of $\overleftarrow{\boldsymbol{x}}$ of length n_1 with d_1 number of distinct symbols. Therefore, $n_1-d_1 < n-d$.

□

The next lemma shows, that if the starts of the rightmost squares are not tightly packed, the string cannot be a first counterexample to the d-step conjecture.

Lemma 2. *Assume that $\sigma_{d'}(n') \le n'-d'$ for any $n'-d' < n-d$. Let $\boldsymbol{x}$ be a singleton-free (d,n)-string, $2 \le d < n$. Let $1 \le k < n$ and let d_2 be the number of distinct symbols of $\boldsymbol{x}[k+1\,..\,n]$ and let e be the number of distinct symbols occurring in both $\boldsymbol{x}[1\,..\,k]$ and $\boldsymbol{x}[k+1\,..\,n]$.*

(i) If $B_{\boldsymbol{x}}(1,k) \le k-d+d_2$, then $s(\boldsymbol{x}) \le n-d$.

(ii) If $B_{\boldsymbol{x}}(1,k)-E_{\boldsymbol{x}}(1,k) \le e$, then $s(\boldsymbol{x}) \le n-d$.

Proof. Let $\boldsymbol{x}_1 = \boldsymbol{x}[1\,..\,k]$, $n_1 = k$, and d_1 be the number of distinct symbols of $\boldsymbol{x}_1$. Thus, $\boldsymbol{x}_1$ is a (d_1, n_1)-string. Let $\boldsymbol{x}_2 = \boldsymbol{x}[k+1\,..\,n]$, $n_2 = n-k$. Thus, $\boldsymbol{x}_2$ is a (d_2, n_2)-string. Let e_1 be the number

of distinct symbols of $\boldsymbol{x}$ that occur only in $\boldsymbol{x}_1$, and let e_2 be the number of distinct symbols of $\boldsymbol{x}$ that occur only $\boldsymbol{x}_2$. Then $d_1 = e_1+e$, $d_2 = e_2+e$, and $d = e_1+e_2+e$, $d_1+d_2 = e_1+e_2+2e = d+e$.

(i) Assume $B_{\boldsymbol{x}}(1,k) \leq k-d+d_2$.

- Case $d_2 = 1$.
  If $k = n-1$, then $s(\boldsymbol{x}) = B_{\boldsymbol{x}}(1,k) \leq k-d+d_2 = k-d+1 = (n-1)-d+1 = n-d$.
  If $k \leq n-2$, then $s(\boldsymbol{x}_2) = 1$ as $d_2 = 1$, and so $s(\boldsymbol{x}) = B_{\boldsymbol{x}}(1,k)+s(\boldsymbol{x}_2) = B_{\boldsymbol{x}}(1,k)+1 \leq (k-d+1)+1 \leq n-2-d+2 = (k-d)+2 \leq (n-2-d)+2 = n-d$.
- Case $d_2 \geq 2$.
  Then $n_2 \geq d_2$ and so $k = n_1 \leq n-d_2$. By Lemma 1, $n_2-d_2 < n-d$ and so by the assumption of this lemma, $s(\boldsymbol{x}_2) \leq n_2-d_2$. Thus, $s(\boldsymbol{x}) = B_{\boldsymbol{x}}(1,k)+s(\boldsymbol{x}_2) \leq (k-d+d_2)+(n_2-d_2) = k-d+n_2 \leq k-d+(n-k) = n-d$.

(ii) Assume that $B_{\boldsymbol{x}}(1,k)-E_{\boldsymbol{x}}(1,k) \leq e$.
Since $\boldsymbol{x}_1$ is a proper prefix of $\boldsymbol{x}$, and since $\boldsymbol{x}_2$ is a proper suffix of $\boldsymbol{x}$, by Lemma 1, $n_1-d_1, n+2-d_2 < n-d$, and by the assumption of this lemma, $s(\boldsymbol{x}_1) \leq n_1-d_1$ and $s(\boldsymbol{x}_2) \leq n_2-d_2$. Thus, $s(\boldsymbol{x}) = \big(B_{\boldsymbol{x}}(1,k)-E_{\boldsymbol{x}}(1,k)\big)+s(\boldsymbol{x}_1)+s(\boldsymbol{x}_2) \leq \big(B_{\boldsymbol{x}}(1,k)-E_{\boldsymbol{x}}(1,k)\big)+(n_1-d_1)+(n_2-d_2) = \big(B_{\boldsymbol{x}}(1,k)-E_{\boldsymbol{x}}(1,k)\big)+n-d-e \leq e+n-d-e = n-d$.

□

Corollary 3. *Assume that $\sigma_{d'}(n') \leq n'-d'$ for any $n'-d' < n-2$. Let $\boldsymbol{x}$ be a singleton-free binary string of length n, $n \geq 3$. Let $1 \leq k < n-2$.*
(i) If $B_{\boldsymbol{x}}(1,k) \leq k$, then $s(\boldsymbol{x}) \leq n-2$.
(ii) If $B_{\boldsymbol{x}}(1,k)-E_{\boldsymbol{x}}(1,k) \leq 2$, then $s(\boldsymbol{x}) \leq n-2$.

Proof. Let $\boldsymbol{x}_1 = \boldsymbol{x}[1\,..\,k]$, d_1 be the number of distinct symbols of $\boldsymbol{x}_1$, and $n_1 = k$ be its length. Let $\boldsymbol{x}_2 = \boldsymbol{x}[k+1\,..\,n]$, d_2 be the number of distinct symbols of $\boldsymbol{x}_2$, and $n_2 = n-k$ be its length. Let e be the number of distinct symbols common to both $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.
For (i):

- If $d_2 = 2$, it follows directly from Lemma 2.
- If $d_2 = 1$, then $s(\boldsymbol{x}_2) \leq 1$. So $s(\boldsymbol{x}) = B_{\boldsymbol{x}}(1,k)+s(\boldsymbol{x}_2) \leq k+1 \leq n-3+1 = n-2$.

For (ii):

- If $d_1 = d_2 = 2$, it follows directly from Lemma 2 as $e = 2$.
- If $d_1 = 1$ and $d_2 = 2$, then $s(\boldsymbol{x}_1) \leq 1$, and so $s(\boldsymbol{x}) \leq 1+s(\boldsymbol{x}_2) \leq 1+n_2-2 = n_2-1 \leq (n-1)-1 = n-2$.
- If $d_1 = 2$ and $d_2 = 1$, we proceed as in the previous case.

- If $d_1 = d_2 = 1$, it follows that $e = 0$ and $B_{\boldsymbol{x}}(1,k) - E_{\boldsymbol{x}}(1,k) = 0$. Then $s(\boldsymbol{x}) \leq B_{\boldsymbol{x}}(1,k) + s(\boldsymbol{x}_1) + s(\boldsymbol{x}_2)$.
  If $n = 3$, if $k = 1$ then $s(\boldsymbol{x}_1) = 0$ and $s(\boldsymbol{x}_2) \leq 1$, and if $k = 2$, then $s(\boldsymbol{x}_1) \leq 1$ and $s(\boldsymbol{x}_2) = 0$, so $s(\boldsymbol{x}_1) + s(\boldsymbol{x}_2) \leq 1$. Thus, $s(\boldsymbol{x}) \leq 1 = 3-2 = n-2$.
  If $n \geq 4$, then $s(\boldsymbol{x}) = s(\boldsymbol{x}_1) + s(\boldsymbol{x}_2) \leq 1+1 = 2 \leq n-2$.

□

In the following, by the *first* counterexample we mean a (d,n)-string from the first column of the $(d, n-d)$ table (see [3, 5, 6, 7]) that does not satisfy the d-step conjecture. Note that the assumption of Corollary 4 is that up to $n-d$, all columns of the table satisfy the d-step conjecture, so the column $n-d$ may harbor the first counterexample. When a square is both the rightmost and the leftmost occurrence in a string, we refer to such a square as *unique*.

Thus, the next lemma shows that a first counterexample to the d-step conjecture $\boldsymbol{x}[1\,..\,n]$ must start with two rightmost (and hence unique) squares, and a rightmost (and hence unique) square at the second position. Since $s(\overleftarrow{\boldsymbol{x}}) = s(\boldsymbol{x})$, $\overleftarrow{\boldsymbol{x}}$ must start with two unique squares, and a unique square at the second position, i.e., $\boldsymbol{x}$ must have two unique squares ending in $\boldsymbol{x}[n]$ and a unique square ending in $n-1$.

Corollary 4. *Assume that $\sigma_{d'}(n') \leq n'-d'$ for any $n'-d' < n-d$. Let $\boldsymbol{x}$ be a singleton-free (d,n)-string, $2 \leq d < n$.*

- *If $B_{\boldsymbol{x}}(1,1) \leq 1$, then $s(\boldsymbol{x}) \leq n-d$.*
- *If $B_{\boldsymbol{x}}(1,2) \leq 2$, then $s(\boldsymbol{x}) \leq n-d$.*

Proof. Since $\boldsymbol{x}$ is singleton-free, $\boldsymbol{x}[2\,..\,n]$ has d distinct symbols. If $B_{\boldsymbol{x}}(1,1) \leq 1$, then $s(\boldsymbol{x}) \leq 1 + s(\boldsymbol{x}[2\,..\,n]) \leq 1+(n-1)-d = n-d$. Thus, we can assume $B_{\boldsymbol{x}}(1,1) = 2$. It follows that $\boldsymbol{x}[3\,..\,n]$ has d distinct symbols. If $B_{\boldsymbol{x}}(1,2) \leq 2$, then $s(\boldsymbol{x}) = B_{\boldsymbol{x}}(1,2) + s(\boldsymbol{x}[3\,..\,n]) \leq 2+(n-2)-d = n-d$.

□

Lemma 5 shows that a first counterexample to the d-step conjecture cannot be a square.

Lemma 5. *Assume that $\sigma_{d'}(n') \leq n'-d'$ for any n'–$d' < n$–d. Let $2 \leq d < n$. For any (d,n)-string $\boldsymbol{x}$ that is square, $s(\boldsymbol{x}) \leq n-d$.*

Proof. The proof relies on the combinatorics of FS-double squares analyzed in [8]; the notion of FS-double squares and inversion factors are presented and discussed there, as well as the notions of α-mate, β-mate, γ-mate, δ-mate, and ϵ-mate.

For a deeper understanding of the proof, the reader must consider reading and understanding the work in [8]. Just to make the proof a little bit more self-contained, here are a few relevant facts from [8]:

- At any position, at most two rightmost squares can start.

- Two rightmost squares starting at the same positions form a so-called FS-double square; every FS-double square has a form $\boldsymbol{u}_1^p\boldsymbol{u}_2\boldsymbol{u}_1^{p+q}\boldsymbol{u}_2\boldsymbol{u}_1^q$ for some primitive $\boldsymbol{u}_1$, a non-empty proper prefix $\boldsymbol{u}_2$ of $\boldsymbol{u}_1$, and some integers $p \geq q \geq 1$; the longer square is $[\boldsymbol{u}_1^p\boldsymbol{u}_2\boldsymbol{u}_1^q][\boldsymbol{u}_1^p\boldsymbol{u}_2\boldsymbol{u}_1^q]$ and the shorter square is $[\boldsymbol{u}_1^p\boldsymbol{u}_2][\boldsymbol{u}_1^p\boldsymbol{u}_2]$.
- If $\boldsymbol{u}_1 = \boldsymbol{u}_2\bar{\boldsymbol{u}}_2$, then the so-called inversion factor $\bar{\boldsymbol{u}}_2\boldsymbol{u}_2\boldsymbol{u}_2\bar{\boldsymbol{u}}_2$ only occurs twice in the FS-double square $\boldsymbol{u}_1^p\boldsymbol{u}_2\boldsymbol{u}_1^{p+q}\boldsymbol{u}_2\boldsymbol{u}_1^q = (\boldsymbol{u}_2\bar{\boldsymbol{u}}_2)^p\boldsymbol{u}_2(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2)^{p+q}\boldsymbol{u}_2(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2)^q$, which highly constrain occurrences of squares starting after the FS-double square.
- The maximum left cyclic shift of $\boldsymbol{u}_1^p\boldsymbol{u}_2\boldsymbol{u}_1^{p+q}\boldsymbol{u}_2\boldsymbol{u}_1^q$ is determined by $lcs(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2, \bar{\boldsymbol{u}}_2\boldsymbol{u}_2)$, where *lcs* stands for *longest common suffix*, while the maximum right cyclic shift is determined by $\bar{\boldsymbol{u}}_2\boldsymbol{u}_2\boldsymbol{u}_2\bar{\boldsymbol{u}}_2$ is controlled by the value of $lcp(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2, \bar{\boldsymbol{u}}_2\boldsymbol{u}_2)$, where *lcp* stands for *longest common prefix*.
- $0 \leq lcs(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2, \bar{\boldsymbol{u}}_2\boldsymbol{u}_2)+lcp(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2, \bar{\boldsymbol{u}}_2\boldsymbol{u}_2) \leq |\boldsymbol{u}_1|-2$, see Lemma 11 of [8].
- The starting point of the first occurrence of the inversion factor $\bar{\boldsymbol{u}}_2\boldsymbol{u}_2\boldsymbol{u}_2\bar{\boldsymbol{u}}_2$ is the position L_1, while the starting point of its maximum right cyclic shift is the position R_1, hence $R_1 \leq L_1+|\boldsymbol{u}_1|-1$.
- An FS-double square $\boldsymbol{v}_1^r\boldsymbol{v}_2\boldsymbol{v}_1^{r+t}\boldsymbol{v}_2\boldsymbol{v}_1^t$ starting at a position i is an α-mate of an FS-double-square $\boldsymbol{u}_1^p\boldsymbol{u}_2\boldsymbol{u}_1^{p+q}\boldsymbol{u}_2\boldsymbol{u}_1^q$ starting at a position j if $j < i \leq R_1$ of $\boldsymbol{u}_1^p\boldsymbol{u}_2\boldsymbol{u}_1^{p+q}\boldsymbol{u}_2\boldsymbol{u}_1^q$ and it is a right cyclic shift of $\boldsymbol{u}_1^p\boldsymbol{u}_2\boldsymbol{u}_1^{p+q}\boldsymbol{u}_2\boldsymbol{u}_1^q$.
- An FS-double square $\boldsymbol{v}_1^r\boldsymbol{v}_2\boldsymbol{v}_1^{r+t}\boldsymbol{v}_2\boldsymbol{v}_1^t$ starting at a position i is a β-mate of an FS-double-square $\boldsymbol{u}_1^p\boldsymbol{u}_2\boldsymbol{u}_1^{p+q}\boldsymbol{u}_2\boldsymbol{u}_1^q$ starting at a position j if it starts at a position $j < i \leq R_1$ of $\boldsymbol{u}_1^p\boldsymbol{u}_2\boldsymbol{u}_1^{p+q}\boldsymbol{u}_2\boldsymbol{u}_1^q$ and it is a right cyclic shift of $\boldsymbol{u}_1^{p-k}\boldsymbol{u}_2\boldsymbol{u}_1^{p+q}\boldsymbol{u}_2\boldsymbol{u}_1^{q+k}$ for some k such that $p-k \geq q+k \geq 1$.
- An FS-double square $\boldsymbol{v}_1^r\boldsymbol{v}_2\boldsymbol{v}_1^{r+t}\boldsymbol{v}_2\boldsymbol{v}_1^t$ starting at a position i is a γ-mate of an FS-double-square $\boldsymbol{u}_1^p\boldsymbol{u}_2\boldsymbol{u}_1^{p+q}\boldsymbol{u}_2\boldsymbol{u}_1^q$ starting at a position j if $j < i \leq R_1$ of $\boldsymbol{u}_1^p\boldsymbol{u}_2\boldsymbol{u}_1^{p+q}\boldsymbol{u}_2\boldsymbol{u}_1^q$ and $\boldsymbol{v}_1^r\boldsymbol{v}_2\boldsymbol{v}_1^r\boldsymbol{v}_2$ is a right cyclic shift of $\boldsymbol{u}_1^p\boldsymbol{u}_2\boldsymbol{u}_1^{p+q}\boldsymbol{u}_2\boldsymbol{u}_1^q$.
- An FS-double square $\boldsymbol{v}_1^r\boldsymbol{v}_2\boldsymbol{v}_1^{r+t}\boldsymbol{v}_2\boldsymbol{v}_1^t$ starting at a position i is a δ-mate of an FS-double-square $\boldsymbol{u}_1^p\boldsymbol{u}_2\boldsymbol{u}_1^{p+q}\boldsymbol{u}_2\boldsymbol{u}_1^q$ starting at a position j if $j < i \leq R_1$ of $\boldsymbol{u}_1^p\boldsymbol{u}_2\boldsymbol{u}_1^{p+q}\boldsymbol{u}_2\boldsymbol{u}_1^q$ and $|\boldsymbol{v}_1^r\boldsymbol{v}_2\boldsymbol{v}_1^r\boldsymbol{v}_2| > |\boldsymbol{u}_1^p\boldsymbol{u}_2\boldsymbol{u}_1^{p+q}\boldsymbol{u}_2\boldsymbol{u}_1^q|$.
- An FS-double square $\boldsymbol{v}_1^r\boldsymbol{v}_2\boldsymbol{v}_1^{r+t}\boldsymbol{v}_2\boldsymbol{v}_1^t$ starting at a position i is an ϵ-mate of an FS-double-square $\boldsymbol{u}_1^p\boldsymbol{u}_2\boldsymbol{u}_1^{p+q}\boldsymbol{u}_2\boldsymbol{u}_1^q$ starting at a position j if $i > R_1$.
- For any two FS-double squares, the one starting later is either an α-mate, or β-mate, or γ-mate, or δ-mate, or ϵ-mate. There are no other possibilities.

Because $\boldsymbol{x}$ is a square, it is singleton-free. By Corollary 4, it must start with two unique squares, otherwise $s(x) \leq n-d$ and we are done. Hence $\boldsymbol{x}$ is an FS-double square $\boldsymbol{x} = \boldsymbol{u}_1^p\boldsymbol{u}_2\boldsymbol{u}_1^{p+q}\boldsymbol{u}_2\boldsymbol{u}_1^q$ for some primitive $\boldsymbol{u}_1$, a non-empty proper prefix $\boldsymbol{u}_2$ of $\boldsymbol{u}_1$, and some $1 \leq q \leq p$. Consider an FS-double square starting at a position i, $1 < i < R_1$. By Lemma 19 of [8], it must be one of the 5 cases – either it is an α-mate of the starting double square, or a β-mate, or a γ-mate, or a δ-mate, or an ϵ-mate. It cannot be an α-mate as its longer square would have the same size as $\boldsymbol{x}$ and hence would

not fit in $\boldsymbol{x}$, nor could it be a β-mate as again its longer square would not fit in $\boldsymbol{x}$, nor could it be a γ-mate as its shorter square would have the same size as $\boldsymbol{x}$ and hence would not fit, nor could it be a δ-mate as again its longer square would not fit in $\boldsymbol{x}$, nor could it be an ϵ-mate as it would have to start at the position R_1 or later. So, we must conclude that no FS-double square can start in the positions $2\,..\,R_1-1$, so at the most a single rightmost square can start at any of the positions $2\,..\,R_1-1$.

Consider a rightmost square $\boldsymbol{vv}$ starting at a some position of $2\,..\,R_1-1$. Since $|\boldsymbol{v}| < |\boldsymbol{x}|$, by Lemma 17 of [8], v must be a right cyclic shift of $\boldsymbol{u}_1^j\boldsymbol{u}_2$ for some $q < j \leq p$. Since maximal right cyclic shift is at most $|\boldsymbol{u}_1|-2$, there is no square starting at the position $H = 2+lcp(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2, \bar{\boldsymbol{u}}_2\boldsymbol{u}_2) < |\boldsymbol{u}_1|+1$: first there are the right cyclic shifts of $\boldsymbol{u}_1^p\boldsymbol{u}_2\boldsymbol{u}_1^p\boldsymbol{u}_2$, and there are at most $|\boldsymbol{u}_1|-2$ of them, and so the last one starts at the position $1+lcp(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2, \bar{\boldsymbol{u}}_2\boldsymbol{u}_2)$. If $p = q$, it is all, and thus the position $H = 2+lcp(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2, \bar{\boldsymbol{u}}_2\boldsymbol{u}_2) < |\boldsymbol{u}_1|+1$ has no square starting there. If $p > q$, then next square is the maximal left cyclic shift of $\boldsymbol{u}_1^{p-1}\boldsymbol{u}_2\boldsymbol{u}_2\boldsymbol{u}_1^{p-1}$ at the position $|\boldsymbol{u}_1|+1$, hence it starts at the position $|\boldsymbol{u}_1|+1-lcs(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2, \bar{\boldsymbol{u}}_2\boldsymbol{u}_2)$. Since $lcp(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2, \bar{\boldsymbol{u}}_2\boldsymbol{u}_2)+lcs(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2, \bar{\boldsymbol{u}}_2\boldsymbol{u}_2) \leq |\boldsymbol{u}_1|-2$, so again the position $H = 2+lcp(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2, \bar{\boldsymbol{u}}_2\boldsymbol{u}_2)$ has no square starting there.

So there is a double square at position 1, at most single squares at positions $2\,..\,H-1$, and no square at position H, so $B(1,H) \leq H$. Since $\boldsymbol{x}[H+1\,..\,n]$ contains $\boldsymbol{u}$, then d_2, the number of distinct symbols of $\boldsymbol{x}[H+1\,..\,n]$, equals d. By Lemma 2, $s(\boldsymbol{x}) \leq n-d$.

□

As indicated in Section 2, we denote a square in $\boldsymbol{x}$ that starts at the position a and ends at the position b as a pair (a,b). The notion of S-cover was introduced in [7].

Definition 6. *Consider a string $\boldsymbol{x}[1\,..\,n]$. The sequence of squares $S = \{(a_i,b_i) \;:\; 1 \leq i \leq k\}$, $1 \leq k$, is a* partial S-cover *of $\boldsymbol{x}$ if*

- *(i) each (a_i,b_i) is a rightmost primitively rooted square of $\boldsymbol{x}$,*
- *(ii) for every $i < k$, $a_i < a_{i+1} < b_i < b_{i+1}$,*
- *(iii) for every rightmost primitively-rooted square (a,b) of $\boldsymbol{x}$ where $a \leq a_k$, there is i, $1 \leq i \leq k$ so that $a_i \leq a$ and $b \leq b_i$.*

S is an S-cover *of $\boldsymbol{x}$ if it is a partial S-cover of $\boldsymbol{x}$ and $b_k = n$.*

Note 7. *If a string has an S-cover, it is necessarily singleton-free.*

Lemma 8 ([7])**.** *Consider a (d,n)-string $\boldsymbol{x}$, $2 \leq d < n$. Then either $s(\boldsymbol{x}) \leq n-d$ or $\boldsymbol{x}$ has an S-cover.*

Lemma 9 ([7])**.** *Consider a (d,n)-string $\boldsymbol{x}$, $2 \leq d < n$. If $\boldsymbol{x}$ admits an S-cover, the S-cover is unique.*

The following corollary gives a strong restriction for the S-cover of a first counterexample.

Corollary 10. *Assume that $\sigma_{d'}(n') \leq n' - d'$ for any $n'-d' < n-d$. Let $2 \leq d < n$. Consider a (d,n)-string $\boldsymbol{x}$, $2 \leq d < n$ with an S-cover $S = \{(a_i, b_i) \ : \ 1 \leq i \leq k\}$.*

(i) If S has size 1, then $s(\boldsymbol{x}) \leq n-d$.

(ii) Either $s(\boldsymbol{x}) \leq n$–d, or $(a_1, b_1) = (1, b_1)$ is an FS-double square $(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2)^p\boldsymbol{u}_2(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2)^{p+q}\boldsymbol{u}_2(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2)^q$ for some suitable $\boldsymbol{u}_2, \bar{\boldsymbol{u}}_2, p$, and q, and

(a) $\boldsymbol{x}[1] = \boldsymbol{x}[b_1+1]$ and $a_2 = 2$ and $b_1+1 \leq b_2 \leq n$, or

(b) $\boldsymbol{x}[1] \neq \boldsymbol{x}[b_1+1]$ and $a_2 = 2$ and $b_1+1 < b_2 \leq n$, or

(c) $\boldsymbol{x}[1] \neq \boldsymbol{x}[b_1+1]$ and $2 < a_2 \leq 2+lcp(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2, \bar{\boldsymbol{u}}_2\boldsymbol{u}_2)$, and there is a rightmost square of $\boldsymbol{x}[1\,..\,b_1]$ at the position 2.

Proof. For (i): If $\boldsymbol{x}$ has an S-cover of size 1, then $\boldsymbol{x}$ is a square. By Lemma 5, $s(\boldsymbol{x}) \leq n-d$.
For (ii): If $B_{\boldsymbol{x}}(1,1) < 2$, then by Corollary 4, $s(\boldsymbol{x}) \leq n-d$. Hence (a_1, b_1) must be an FS-double square $(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2)^p\boldsymbol{u}_2(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2)^{p+q}\boldsymbol{u}_2(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2)^q$ for some suitable $\boldsymbol{u}_2, \bar{\boldsymbol{u}}_2, p$, and q.

- If $\boldsymbol{x}[1] = \boldsymbol{x}[b_1+1]$ then (a_1, b_1) can be cyclically shifted to the right, and so a_2 is forced to be equal to 2. Either $(2, b_1+1)$ is the longest rightmost square of $\boldsymbol{x}$ starting at the position 2, and then $(a_2, b_2) = (2, b_1+1)$, or there is a longer rightmost square starting at the position 2. Since (a_2, b_2) is the longest rightmost square starting at the position 2, $b_2 > b_1+1$.
- If $\boldsymbol{x}[1] \neq \boldsymbol{x}[b_1+1]$ and there is no rightmost square of $\boldsymbol{x}[1\,..\,b_1]$ starting at the position 2, then by Corollary 4, there must be a rightmost square of $\boldsymbol{x}$ starting at the position 2 otherwise the number of rightmost squares of $\boldsymbol{x}$ would be bounded by $n-d$. Hence the rightmost square at the position 2 must be strictly longer than b_1, and so $b_1+1 < b_2$.
- Let $H = 2+lcp(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2, \bar{\boldsymbol{u}}_2\boldsymbol{u}_2)$. If $H < a_2$, then a rightmost square starting at a position $2\,..\,H$ must be completely covered by (a_1, b_1) due to property (iii) of S-cover, and so by Lemma 17 of [8], must be maximal left cyclic shift of $\boldsymbol{u}^j\boldsymbol{v}\boldsymbol{u}^j\boldsymbol{v}$ for some $q \leq j \leq p$. Similarly as in the proof of Lemma 5, we can show that there are at most one rightmost square starting at a position from $2\,..\,H$ (as $H < a_2$), and that there is not a square starting at the position H, so by Corollary 4, $s(\boldsymbol{x}) \leq n-d$. By Lemma 17 of [8], the size of (a_2, b_2) must be $\geq$ the size of (a_1, b_1), and so $b_1 \leq b_2-a_2+1$ giving $b_2 \geq b_1+a_2-1$.

□

4 The Computer Search Framework for a Counterexample to the d-step Conjecture

Due to the complexity of a detailed framework, we present a high-level logic outline of the algorithm with justification for each step. Given n and d, $2 \leq d < n$ and knowing that whenever $n'-d' < n-d$, then $\sigma_{d'}(n') < n'-d'$, we search for the first counterexample. This is done by recursively generating a suitable S-cover, rather than the string.

1. The recursion starts with generating all possible (a_1, b_1) in a loop. From Corollary 10, we must generate $(a_1, b_1) = (1, b_1)$ as a primitively-rooted FS-double square $(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2)^p\boldsymbol{u}_2(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2)^{p+q}\boldsymbol{u}_2(\boldsymbol{u}_2\bar{\boldsymbol{u}}_2)^q$ where $|\boldsymbol{u}_2|^{2(p+q+1)} + |\bar{\boldsymbol{u}}_2|^{2(p+q)} < n$. Each generated partial S-cover (a_1, b_1) is passed to the next recursive call. When the loop is over, we return to the caller.

2. Using a loop, the next recursive step generates all possible primitively-rooted squares (a_2, b_2) as indicated by Corollary 10. We extend the partial S-cover (a_1, b_1) by the generated (a_2, b_2) and compute all rightmost squares of $\boldsymbol{x}[1\,..\,b_2]$, $B(1, a_2-1)$ and $E(1, a_2-1)$. For each partial S-cover $(a_1, b_1), (a_2, b_2)$ we perform the following checks, and if they are all successful, the partial S-cover $(a_1, b_1), (a_2, b_2)$ is passed to the next recursive call. When the loop is over, we return to the caller.

   - If $d = 2$ and $B(1, j) \leq j$ for some $2 \leq j < a_2$, then by Corollary 4, it cannot be a beginning of a counterexample, so we abandon it and jump to the top of the loop to try another (a_2, b_2).
   - If $d = 2$ and $B(1, j) - E(1, j) \leq 2$ for some $2 \leq j < a_2$, then by Corollary 4, it cannot be a beginning of a counterexample, so we abandon it and jump to the top of the loop to try another (a_2, b_2).
   - If $d > 2$, by Lemma 2, if $B(1, j) \leq j-d+2$ for some $2 \leq j < a_2$, it cannot be a beginning of a counterexample, so we abandon it and jump to the top of the loop to try another (a_2, b_2).
   - If $d > 2$ and $B(1, j) - E(1, j) = 0$ for some $2 \leq j < a_2$, then by Lemma 2, it cannot be a beginning of a counterexample, so we abandon it and jump to the top of the loop to try another (a_2, b_2).
   - If $a_2 > 2$ and there is no rightmost square at the position 2 (there was one in $\boldsymbol{x}[1\,..\,b_1]$ but in the extension $\boldsymbol{x}[1\,..\,b_2]$ there is another occurrence, so it is no longer rightmost), we abandon further extension and jump to the top of the loop to try another (a_2, b_2).
   - If $b_2 = n$, we check if the string $\boldsymbol{x}[1\,..\,n]$ has d distinct symbols. If not, we jump to the top of the loop to try another (a_2, b_2).
   - If $b_2 = n$ and the number of the rightmost squares is $\leq n-d$ we jump to the top of the loop to try another (a_2, b_2). On the other hand, if it is $> n-d$, **we stop the execution and announce and display the found counterexample**.
   - If $b_2 < n$ and (a_1, b_1) is no longer a rightmost square (there is a further occurrence of $\boldsymbol{x}[a_1\,..\,b_1]$ in $\boldsymbol{x}[a_1\,..\,b_2]$), we abandon the generation of the rest as we are no longer building a viable S-cover and jump to the top of the loop to try another (a_2, b_2).
   - If there is a rightmost square that is not completely covered by (a_1, b_1) or (a_2, b_2), (property (iii) of Definition 6), again we are not building a viable S-cover and so further generation is abandoned and we jump to the top of the loop to try another (a_2, b_2).

- Otherwise, all checks were successful, and so we pass the partial S-cover $(a_1, b_1), (a_2, b_2)$ to the next recursive call for further extension. When the call returns, we jump to the top of the loop to try another (a_2, b_2).

3. In the next recursive step, the partial S-cover $(a_1, b_1), ..., (a_k, b_k)$ for $2 \leq k$ is extended in a loop by all possible combinations of primitively-rooted (a_{k+1}, b_{k+1}). From the definition of S-cover, $a_{k+1} < a_k$ and $b_{k+1} > b_k$. We compute all rightmost squares of $\boldsymbol{x}[1 .. b_{k+1}]$, $B(1, a_{k+1}-1)$ and $E(1, a_{k+1}-1)$. For each partial S-cover $(a_1, b_1), ..., (a_{k+1}, b_{k+1})$, we perform the following checks, and if they are all successful, the partial S-cover $(a_1, b_1), ..., (a_{k+1}, b_{k+1})$ is passed to the next recursive call. When the loop is over, we return to the caller.
   - If $d = 2$ and $B(1, j) \leq j$ for some $2 \leq j < a_{k+1}$, then by Corollary 4, it cannot be a beginning of a counterexample, so we abandon it and jump to the top of the loop to try another (a_{k+1}, b_{k+1}).
   - If $d = 2$ and $B(1, j) - E(1, j) \leq 2$ for some $2 \leq j < a_{k+1}$, then by Corollary 4, it cannot be a beginning of a counterexample, so we abandon it and jump to the top of the loop to try another (a_{k+1}, b_{k+1}).
   - If $d > 2$, by Lemma 2, if $B(1, j) \leq j - d + 2$ fro some $2 \leq j < a_{k+1}$, it cannot be a beginning of a counterexample, so we abandon it and jump to the top of the loop to try another (a_{k+1}, b_{k+1}).
   - If $d > 2$ and $B(1, j) - E(1, j) = 0$ for some $2 \leq j < a_2$, then by Lemma 2, it cannot be a beginning of a counterexample, so we abandon it and jump to the top of the loop to try another (a_{k+1}, b_{k+1}).
   - If $a_2 > 2$ and there is no rightmost square at the position 2 (there was one in $\boldsymbol{x}[1 .. b_1]$ but in the extension $\boldsymbol{x}[1 .. b_{k+1}]$ there is another occurrence, so it is no longer rightmost), we abandon further extension and jump to the top of the loop to try another (a_{k+1}, b_{k+1}).
   - If $b_{k+1} = n$, we check if the string $\boldsymbol{x}[1 .. n]$ has d distinct symbols. If not, we jump to the top of the loop to try another (a_{k+1}, b_{k+1}).
   - If $b_2 = n$ and the number of the rightmost squares is $\leq n - d$ we jump to the top of the loop to try another (a_2, b_2). On the other hand, if it is $> n - d$, **we stop the execution and announce and display the found counterexample**.
   - If $b_2 < n$ and (a_1, b_1) is no longer a rightmost square (there is a further occurrence of $\boldsymbol{x}[a_1 .. b_1]$ in $\boldsymbol{x}[a_1 .. b_2]$), we abandon the generation of the rest as we are no longer building a viable S-cover and jump to the top of the loop to try another (a_{k+1}, b_{k+1}).
   - If there is a rightmost square that is not completely covered by $(a_1, b_1), ..., (a_{k+1}, b_{k+1})$, (property (iii) of Definition 6), again we are not building a viable S-cover and so further generation is abandoned and we jump to the top of the loop to try another (a_{k+1}, b_{k+1}).
   - Otherwise all checks were successful, and so we pass the partial S-cover $(a_1, b_1), ..., (a_{k+1}, b_{k+1})$ to the next recursive call for further extension. When the call returns, we jump to the top of the loop to try another (a_{k+1}, b_{k+1}).

5 Conclusion and Future Work

In conclusion, we present a framework for computer search for counterexamples to the d-step conjecture, i.e., strings of length n with d distinct symbols that admit strictly more than $n-d$ rightmost squares. The significant restriction on a form of a first counterexample presented in a series of lemmas and corollaries, allows for an early abandonment of partially generated strings that could not possibly be counterexamples, thus, dramatically reducing the search space.

6 Acknowledgment

The research of the first author was supported by the National Sciences and Research Council of Canada (NSERC) grant RGPIN/5504-2018. The research of the second author was supported by the University of Toronto Mississauga's Office of the Vice-Principal, Research (OVPR) Fund.

References

[1] F. Blanchet-Sadri, M. Bodnar, J. Nikkel, J.D. Quigley, and X. Zhang. Squares and primitivity in partial words. *Discrete Applied Mathematics*, 185:26–37, 2015.

[2] F. Blanchet-Sadri and S. Osborne. Constructing words with high distinct square densities. *Electronic Proceedings in Theoretical Computer Science*, 252:71–85, 08 2017.

[3] A. Deza and F. Franek. A d-step approach to the maximum number of distinct squares and runs in strings. *Discrete Applied Mathematics*, 163:268–274, 2014.

[4] A. Deza, F. Franek, and M. Jiang. Square-maximal strings. `http://optlab.mcmaster.ca/~jiangm5/research/square.html`.

[5] A. Deza, F. Franek, and M. Jiang. A d-step approach for distinct squares in strings. In *Proceedings of 22nd Annual Symposium on Combinatorial Pattern Matching - CPM 2011*, pages 11–89, 2011.

[6] A. Deza, F. Franek, and M. Jiang. A computational framework for determining square-maximal strings. In J. Holub and J. Žd'árek, editors, *Proceedings of Prague Stringology Conference 2012*, pages 112–119. Czech Technical University, Prague, Czech Republic, 2012.

[7] A. Deza, F. Franek, and M. Jiang. A computational substantiation of the d-step approach to the number of distinct squares problem. *Discrete Applied Mathematics*, pages 81–87, 2016.

[8] A. Deza, F. Franek, and A. Thierry. How many double squares can a string contain? *Discrete Applied Mathematics*, 180:52–69, 2015.

[9] A. S. Fraenkel and J. Simpson. How many squares can a string contain? *Journal of Combinatorial Theory Series A*, 82:112–120, 1998.

[10] L. Ilie. A simple proof that a word of length n has at most $2n$ distinct squares. *Journal of Combinatorial Theory Series A*, 112:163–164, 2005.

[11] L. Ilie. A note on the number of squares in a word. *Theoretical Computer Science*, 380:373–376, 2007.

[12] N. Jonoska, , F. Manea, and S. Seki. A Stronger Square Conjecture on Binary Words. In *SOFSEM 2014: Theory and Practice of Computer Science*, pages 339–350. Springer International Publishing, 2014.

[13] N.H. Lam. On the number of squares in a string. AdvOL-Report 2013/2, McMaster University, 2013.

[14] F. Manea and S. Seki. Square-Density Increasing Mappings. In *Combinatorics on Words*, pages 160–169. Springer International Publishing, 2015.

[15] A. Thierry. A proof that a word of length n has less than $1.5n$ distinct squares. *arXiv:2001.02996*, 2020.