McMaster University

Advanced Optimization Laboratory

Title:

A computational framework for determining square-maximal strings

Authors:

Antoine Deza, Frantisek Franek, and Mei Jiang

AdvOL-Report No. 2011/5

December 2011, Hamilton, Ontario, Canada

A computational framework for determining square-maximal strings [*]

Antoine Deza, Frantisek Franek, and Mei Jiang

Abstract

We investigate the function $\sigma_d(n) = \max\{\boldsymbol{s}(x) \mid x \text{ is a } (d,n)\text{-string}\}$, where $\boldsymbol{s}(x)$ denotes the number of distinct primitively rooted squares in a string x and (d,n)-string denotes a string of length n with exactly d distinct symbols. New properties of the $\sigma_d(n)$ function are presented. The notion of s-cover is presented and discussed with emphasis on the recursive computational determination of $\sigma_d(n)$. In particular, we were able to determine all values of $\sigma_2(n)$ for $n \leq 53$ and $\sigma_3(n)$ for $n \leq 41$ and to point out that $\sigma_2(33) < \sigma_3(33)$; that is, among all strings of length 33, no binary string achieves the maximum number of distinct primitively rooted squares. Noticeably, these computations reveal the unexpected existence of pairs (d,n) satisfying $\sigma_{d+1}(n+2) - \sigma_d(n) > 1$ such as (2,33) and (2,34), and of three consecutive equal values: $\sigma_2(31) = \sigma_2(32) = \sigma_2(33)$. In addition we show that $\sigma_2(n) \leq 2n - 66$ for $n \geq 53$.

Keywords: *string, square, primitively rooted square, maximum number of distinct primitively rooted squares, parameterized approach, $(d, n-d)$ table*

1 Introduction

In [1] the notion of an r-cover was introduced as a means to represent the distribution of the runs in a string and thus describe the structure of the run-maximal strings. Ignoring the number of distinct symbols d, a key assertion from [1] states that essentially any run-maximal string has an r-cover. A similar approach was adapted for run-maximal (d,n)-strings in [2] and we show in Section 2 how this approach can be adapted for square-maximal (d,n)-strings. This notion is used to speed up computations of the maximum number of distinct primitively rooted squares.

We encode a square as a triple (s,e,p) where s is the starting position of the square, e is the ending position of the square, and p is its period. Note that $e = s + 2p - 1$. The *join* $x[i_1 \,..\, i_k] \cup x[j_1 \,..\, j_m]$ of two substrings of a string $x = x[1 \,..\, n]$ is defined if $i_1 \leq j_1 \leq i_k + 1$ and then $x[i_1 \,..\, i_k] \cup x[j_1 \,..\, j_m] = x[i_1 \,..\, max\{i_k, j_m\}]$, or if $j_1 \leq i_1 \leq j_m + 1$ and then $x[i_1 \,..\, i_k] \cup x[j_1 \,..\, j_m] = x[j_1 \,..\, max\{i_k, j_m\}]$. Simply put, the join is defined when the two substrings either are adjacent or overlapping. The alphabet of x is denoted

[*] This work was supported by the *Natural Sciences and Engineering Research Council of Canada* and *MITACS*, and by the *Canada Research Chair program*, and made possible by the facilities of the *Shared Hierarchical Academic Research Computing Network* (`http://www.sharcnet.ca/`).

by $\mathcal{A}(x)$, (d,n)-string refers to a string of length n with exactly d distinct symbols, $\boldsymbol{s}(x)$ denotes the number of distinct primitively rooted squares in a string x, and $\sigma_d(n)$ refers to the maximum number of distinct primitively rooted squares over all (d,n)-strings, i.e. $\sigma_d(n) = \max\{\boldsymbol{s}(x) \mid x \text{ is a } (d,n)\text{-string}\}$. A singleton is a symbol which occurs exactly once in the string under consideration. To simplify the notation, for an empty string ε we set $\boldsymbol{s}(\varepsilon) = 0$ and $\sigma_d(0) = 0$.

2 Computational approach to distinct primitively rooted squares

In the computational framework for determining $\sigma_d(n)$ we will be discussing later, we first compute a lower bound of $\sigma_d(n)$ denoted as $\sigma_d^-(n)$. It is enough to consider (d,n)-strings x that could achieve $\boldsymbol{s}(x) > \sigma_d^-(n)$ for determining $\sigma_d(n)$, thus significantly reducing the search space. The purpose of this section is to introduce the necessary conditions that guarantee that for such an x, $\boldsymbol{s}(x) > \sigma_d^-(n)$ for a given $\sigma_d^-(n)$. The necessary conditions are the existence of an *s-cover* and a sufficient *density* of the string, see Lemmas 5, 9, 10. The s-cover is guaranteed through generation, while the density is verified incrementally during the generation at the earliest possible stages. Note that the notion of s-cover, though similar to r-cover for runs, see [2, 1], is slightly different.

Definition 1. *An* s-cover *of a string $x = x[1 \mathinner{..} n]$ is a sequence of primitively rooted squares $\{S_i = (s_i, e_i, p_i) \mid 1 \le i \le m\}$ so that*

(1) *for any $1 \le i < m$, $s_i < s_{i+1} \le e_i + 1$ and $e_i < e_{i+1}$, i.e. two consecutive squares are either adjacent or overlapping;*

(2) $\displaystyle\bigcup_{1 \le i \le m} S_i = x$*;*

(3) *for any occurrence of square S in x, there is $1 \le i \le m$ so that S is a substring of S_i, denoted by $S \subseteq S_i$.*

Lemma 2. *An s-cover of a string is unique.*

Proof. Let us assume that we have two different s-covers of x, $\{S_i \mid 1 \le i \le m\}$ and $\{S'_j \mid 1 \le j \le k\}$. We shall prove by induction that they are identical. By Definition 1 (3), $S_1 \subseteq S'_1$ and, by the same argument, $S'_1 \subseteq S_1$, and thus $S_1 = S'_1$. Let the induction hypothesis be $S_i = S'_i$ for $1 \le i \le t$. If $\bigcup_{1 \le i \le t} S_i = x$, we have $t = m = k$ and we are done. Otherwise consider S_{t+1}. By Definition 1 (3), there is S'_v so that $S_{t+1} \subseteq S'_v$ and $v > t$. We need to show that $v = t+1$. If not, then S_{t+1} would neither be a substring of S'_t nor of S'_{t+1} contradicting Definition 1 (3). Therefore $S_{t+1} \subseteq S'_{t+1}$. By the same argument, $S'_{t+1} \subseteq S_{t+1}$ and so $S_{t+1} = S'_{t+1}$. □

Lemma 3. *If a string admits an s-cover, then it is singleton free.*

Proof. Let $\{S_j \mid 1 \leq j \leq m\}$ be the s-cover of $x = x[1 \mathinner{..} n]$. For any $1 \leq i \leq n$, $x[i] \in S_t$ for some t by Definition 1 (2). Since S_t is a square, the symbol $x[i]$ occurs in x at least twice. □

Before we can define what a *dense string* is, we must first define the notion of a *core* of a square, similarly to the core of a run, see [2, 6]. For a square, its core is the set of indices formed by the intersection of the indices of all its occurrences in the string.

Definition 4. *The* core vector $k(x)$ *of a* (d,n)*-string* x *is defined by* $k_i(x) =$ *the number of cores of squares of* x *containing* i *for* $i = 1, \dots, n$. *A singleton-free* (d,n)*-string* x *is* dense, *if its core vector* $k(x)$ *satisfies* $k_i(x) > \sigma_d^-(n) - \boldsymbol{s}(x[1 \mathinner{..} i-1]) - m_i$ *for* $i = 1, \dots, n$, *where* $m_i = \max \{\sigma_{d'}(n-i) \ : \ d - |\mathcal{A}(x[1 \mathinner{..} i-1])| \leq d' \leq \min(n-i, d)\}$.

Lemma 5. *If a* (d,n)*-string* x *is not dense, then* $\boldsymbol{s}(x) \leq \sigma_d^-(n)$.

Proof. The proof follows from the basic observation that for any string x, $\boldsymbol{s}(x) \leq \boldsymbol{s}(x[1 \mathinner{..} i-1]) + \boldsymbol{s}(x[i+1 \mathinner{..} n]) + k_i(x)$ for any i. Note that the inequality occurs when there are the same type of squares in both $x[1 \mathinner{..} i-1]$ and $x[i+1 \mathinner{..} n]$. If x is not dense, then for some i_0, $k_{i_0}(x) \leq \sigma_d^-(n) - \boldsymbol{s}(x[1 \mathinner{..} i_0 - 1]) - m_{i_0}$. Then $\boldsymbol{s}(x) \leq \boldsymbol{s}(x[1 \mathinner{..} i_0 - 1]) + \boldsymbol{s}(x[i_0 + 1 \mathinner{..} n]) + k_{i_0}(x) \leq \boldsymbol{s}(x[1 \mathinner{..} i_0 - 1]) + m_{i_0} + k_{i_0}(x) \leq \boldsymbol{s}(x[1 \mathinner{..} i_0 - 1]) + m_{i_0} + \sigma_d^-(n) - \boldsymbol{s}(x[1 \mathinner{..} i_0 - 1]) - m_{i_0} = \sigma_d^-(n)$. □

Lemma 6. *If the core vector* $k(x)$ *of a* (d,n)*-string* x *satisfies* $k_i(x) > 0$ *for* $i = 1, \dots, n$, *then* x *has an s-cover.*

Proof. We build an s-cover by induction: Since the $k_1(x) \geq 1$, 1 is in at least one core, hence there must be at least one square starting at position 1. Among all squares starting at position 1, set the one with the largest period to be S_1. Suppose that we have built the s-cover $\{S_i = (s_i, e_i, p_i) \ : \ i \leq t\}$. If $\bigcup_{1 \leq i \leq t} S_i = x$, we are done. Otherwise $\bigcup_{1 \leq i \leq t} S_i = x[1 \mathinner{..} e_t]$ where $e_t < n$. Since $k_{e_t+1}(x) \geq 1$, there is at least one square (s, e, p) in x so that $s \leq e_t + 1 \leq s + 2p - 1$. From all such squares choose the leftmost ones, and among them choose the one with the largest period and set it as S_{t+1}. It is straightforward to verify that all the conditions of Definition 1 are satisfied and that we have built the s-cover of x. □

Note that for a (d,n)-string, having an s-cover implies being singleton free. However it does not imply that every $k_i(x) \geq 1$, even though it is quite close to it. Consider the s-cover $\{S_j = (s_j, e_j, p_j) : 1 \leq j \leq m\}$ of x. If S_1 has another occurrence in x and there is no other square in x starting at position 1, then 1 is not in any core and $k_1(x) = 0$. Similarly, if the s-cover has two consecutive adjacent squares S_j and S_{j+1}, if there is no other square occurring at position s_{j+1}, and if the square S_{j+1} has some other occurrence, then $k_{s_{j+1}}(x) = 0$. In this sense, the s-cover is a computationally efficient structural generalization of the property that every $k_i(x) \geq 1$.

Lemma 7. *Let $\{S_i = (s_i, e_i, p_i) \mid 1 \le i \le m\}$ be an s-cover of x. Let $k = k(x)$ be the core vector of x. Then for any $1 \le i < m$ and the core vector $k' = k(x[1 \mathinner{..} e_i])$, $(\forall\ 1 \le j < s_{i+1})(k'_j \ge k_j)$.*

Proof. Let us assume that for some $i = 1, 2, \ldots m-1$ there is a j so that $k_j > k'_j$. Then there must exist a square (s, e, p) in $x = x[1 \mathinner{..} e_m]$ that is not a square of $x[1 \mathinner{..} e_i]$, i.e. $e > e_i$ and $s < s_{i+1}$, so it is an intermediate square violating the definition of s-cover, see Definition 1 (3). □

Lemma 8. *If a square-maximal (d, n)-string x has an s-cover with two consecutive adjacent squares, then $\sigma_d(n) \le \sigma_{d_1}(n_1) + \sigma_{d_2}(n_2)$ for some $2 \le d_1, d_2 \le d \le d_1 + d_2$ and some n_1, n_2, possibly equal to zero, such that $n_1 + n_2 = n$.*

Proof. Let $\{S_i \ :\ 1 \le i \le m\}$ be the s-cover of x and let $S_j \cap S_{j+1} = \emptyset$. Then $\boldsymbol{s}(x) \le \boldsymbol{s}(x_1) + \boldsymbol{s}(x_2)$, where $x_1 = \bigcup_{1 \le i \le j} S_i$ and $x_2 = \bigcup_{j < i \le m} S_i$. Therefore $\sigma_d(n) = \boldsymbol{s}(x) \le \boldsymbol{s}(x_1) + \boldsymbol{s}(x_2) \le \sigma_{d_1}(n_1) + \sigma_{d_2}(n_2)$ where x_1 and x_2 are, respectively, a (d_1, n_1)- and a (d_2, n_2)-string. □

Lemma 9. *If a singleton-free square-maximal (d, n)-string x does not have an s-cover, then $\sigma_d(n) = \sigma_d(n-1)$.*

Proof. Since x does not have an s-cover, there exist some i_0 such that $k_{i_0} = 0$ by Lemma 6. Remove $x[i_0]$ to form a $(d, n-1)$-string y. This will not decrease the number of distinct squares in x since there is no core of any square contains i_0. Then $\sigma_d(n) = \boldsymbol{s}(x) \le \boldsymbol{s}(y) \le \sigma_d(n-1)$. Since $\sigma_d(n) \ge \sigma_d(n-1)$ (see [4]), therefore $\sigma_d(n) = \sigma_d(n-1)$. □

Lemma 10. *If a square-maximal (d, n)-string has a singleton, then $\sigma_d(n) = \sigma_{d-1}(n-1)$.*

Proof. Remove the singleton to form a $(d-1, n-1)$-string y with $\sigma_d(n) = \boldsymbol{s}(x) \le \boldsymbol{s}(y) \le \sigma_{d-1}(n-1)$. Since $\sigma_d(n) \ge \sigma_{d-1}(n-1)$ (see [4]), therefore $\sigma_d(n) = \sigma_{d-1}(n-1)$. □

3 Heuristics for a lower bound $\boldsymbol{\sigma_d^-(n)}$

Recall that $\sigma_d^-(n)$ denotes the best available lower bound for $\sigma_d(n)$. The higher the value of $\sigma_d^-(n)$, the less computational effort must be spent on determining $\sigma_d(n)$. For $d = 2$, generate $\mathcal{L}_2(n)$, the set of $(2, n)$-strings which admit an s-cover and are balanced over every prefix (the frequencies of a's and b's differ by at most a predefined constant), have a maximum period bounded by at most a predefined constant, and contain no triples (aaa or bbb). Then we set

$$\sigma_2^-(n) = \max\ \{\sigma_2(n-1),\ \max_{x \in \mathcal{L}_2(n)} \boldsymbol{s}(x)\}.$$

For $d \ge 3$, we simply set $\sigma_d^-(n) = \max\ \{\sigma_{d-1}(n-1),\ \ \sigma_{d-1}(n-2) + 1,\ \ \sigma_d(n-1)\}$. The heuristic was found to be quite efficient.

4 Generating special (d, n)-strings admitting an s-cover

Rather than generating strings, we generate their s-covers. By *special* we mean only s-covers that have no consecutive adjacent squares. The generation proceeds by extending the partially built s-cover in all possible ways. Every time a potential square of the s-cover is to be extended by one position, all previously used symbols and the first unused symbol are tried. For each symbol, the frequency counter is checked that the symbol does not exceed $n + 2 - 2d$. Once a symbol is used, the frequency counter is updated. When the whole s-cover is generated, the counter is checked whether all d symbols occurred in the resulting string; if not, the string is rejected. A typical implementation of the generation of the s-cover would be through recursion as backtracking is needed. For computational efficiency reasons we opted instead for a user-stack controlled backtracking implemented as a co-routine `Next()` allowing us to call the co-routine repeatedly to produce the next string. Note that the strings are generated in a lexico-graphic order. The generation of the s-cover follows these principles: The generator for the first square is created by iterative calls to `Next()` producing all the possible generators. Each generator is checked for the additional properties (must be primitive, did not create an intermediate square in the partial string, etc.) before it is accepted. For each subsequent square, its generator may be partially or fully determined. If it is partially determined, iterative calls to `Next()` are used to generate all possible completions of the generator. The complete generator is checked and accepted or rejected. In addition, if the density of the string being generated is to be checked, we use Lemma 7 and the core vector of the partially generated string to reject the string or allow it to be extended further.

5 Recursive computation of $\sigma_d(n)$

First, $\sigma_d^-(n)$ is computed by the heuristic of Section 3. Then it is verified that $\sigma_{d_1}(n_1) + \sigma_{d_2}(n_2) \leq \sigma_d^-(n)$ for any $2 \leq d_1, d_2 \leq d \leq d_1 + d_2$ and any $n_1 + n_2 = n$. Then $\mathcal{U}_d(n)$, the set of all dense special (d, n)-strings admitting an s-cover is generated as described in Section 4. It follows that

$$\sigma_d(n) = \max\,\{\sigma_d^-(n),\ \max_{x \in \mathcal{U}_d(n)} \boldsymbol{s}(x)\}.$$

To see that, first consider the existence of a square-maximal (d, n)-string with singletons: by Lemma 10, $\sigma_d(n) = \sigma_{d-1}(n - 1)$. Then consider the existence of a singleton-free square-maximal string x not in $\mathcal{U}_d(n)$:
(i) either x does not have an s-cover, in which case by Lemma 9, $\sigma_d(n) = \sigma_d(n - 1)$;
(ii) or x has an s-cover with two consecutive adjacent squares and by Lemma 8, $\sigma_d(n) \leq \sigma_{d_1}(n_1) + \sigma_{d_2}(n_2)$ for some $2 \leq d_1, d_2 \leq d$ and some $n_1 + n_2 = n$, and so $\sigma_d(n) \leq \sigma_d^-(n)$;
(iii) or x has a special s-cover, but is not dense and by Lemma 5, $\sigma_d(n) \leq \sigma_d^-(n)$.

6 Recursive computation of $\sigma_d(2d)$

To compute the values on the main diagonal we can use s-covers satisfying additional necessary parity condition. The s-cover $\{S_i = (s_i, e_i, p_i) : 1 \le i \le m\}$ of $x = x[1 \,..\, n]$ satisfies the *parity condition* if for any $1 \le i < m$, $\mathcal{A}(x[1 \,..\, e_i]) \cap \mathcal{A}(x[s_{i+1} \,..\, n]) \subseteq \mathcal{A}(x[s_{i+1} \,..\, e_i])$.

Lemma 11. *The singleton-free part of a square-maximal $(d, 2d)$-string x with all its singletons at the end has an s-cover satisfying the parity condition.*

Proof. We can assume that x has $0 \le v \le d-2$ singletons, all at the end. Let $k(x)$ be the core vector of x. Suppose the singleton-free part $x[1 \,..\, 2d-v]$ does not have an s-cover, then there exist some $1 \le i_0 \le 2d - v$ such that $k_{i_0}(x) = 0$. Remove $x[i_0]$ to form a $(d, 2d-1)$-string y. Therefore, $\sigma_d(2d) = \boldsymbol{s}(x) \le \boldsymbol{s}(y) \le \sigma_d(2d-1) = \sigma_{d-1}(2d-2)$, a contradiction. So $x[1 \,..\, 2d-v]$ has an s-cover $\{S_i : 1 \le i \le m\}$. Let us assume that the s-cover does not satisfy the parity condition. Then either
(i) $\bigcup_{1 \le i \le t} S_i$ and $\bigcup_{t < i \le m} S_i$ for some $1 \le t \le m$ are adjacent and their respective alphabets have at least one symbol in common, say c. If we replace c in $\bigcup_{1 \le i \le t} S_i$ by a new symbol $\hat{\boldsymbol{c}} \notin \mathcal{A}(x)$, we get a new $(d+1, 2d)$-string y so that $\boldsymbol{s}(y) \ge \boldsymbol{s}(x)$. Thus $\sigma_d(2d) = \boldsymbol{s}(x) \le \boldsymbol{s}(y) \le \sigma_{d+1}(2d) = \sigma_d(2d-1) = \sigma_{d-1}(2d-2)$, a contradiction, or
(ii) $\bigcup_{1 \le i \le t} S_i$ and $\bigcup_{t < i \le m} S_i$ for some $1 \le t \le m$ are overlapping, and there is a symbol c occurring in $\bigcup_{1 \le i \le t} S_i$ and in $\bigcup_{t < i \le m} S_i$, but not in the overlap $S_t \cap S_{t+1}$. If we replace c in $\bigcup_{1 \le i \le t} S_i$ by a new symbol $\hat{\boldsymbol{c}} \notin \mathcal{A}(x)$, we get a new $(d+1, 2d)$-string y so that $\boldsymbol{s}(y) \ge \boldsymbol{s}(x)$. Thus $\sigma_d(2d) = \boldsymbol{s}(x) \le \boldsymbol{s}(y) \le \sigma_{d+1}(2d) = \sigma_d(2d-1) = \sigma_{d-1}(2d-2)$, a contradiction. □

With additional assumptions, Lemma 11 can be strengthen to exclude consecutive adjacent squares from the s-cover of a square-maximal $(d, 2d)$-string.

Lemma 12. *Let $\sigma_{d'}(2d') = d'$ for any $d' < d$. Either $\sigma_d(2d) = d$ or for every square-maximal $(d, 2d)$-string x with v singletons all at the end, $0 \le v \le d-2$, its singleton-free part $x[1 \,..\, 2d-v]$ has an s-cover satisfying the parity condition and which has no consecutive adjacent squares.*

Proof. The existence of the s-cover $\{S_i \mid 1 \le i \le m\}$ of $x[1 \,..\, 2d-v]$ satisfying the parity condition follow from Lemma 11. We need to prove that either $\sigma_d(2d) = d$ or there are no adjacent squares in the s-cover. Since $\sigma_{d'}(2d') = d'$ for any $d' < d$, $\sigma_{d'}(n') \le n' - d'$ for any $n' - d' < d$. Let us assume that the s-cover of x has two adjacent squares S_t and S_{t+1}. Let $x_1 = \bigcup_{1 \le i \le t} S_i$ and let $x_2 = \bigcup_{t < i \le m} S_i$. Then $\boldsymbol{s}(x) \le \boldsymbol{s}(x_1) + \boldsymbol{s}(x_2)$ whre x_1 and x_2 are, respectively, a (d_1, n_1)- and a (d_2, n_2)-string with $n_1 + n_2 = 2d - v$ and $d_1 + d_2 \ge d - v$. Since the s-cover satisfies the parity condition, $\mathcal{A}(x_1)$ and $\mathcal{A}(x_2)$ are disjoint and hence $d_1 + d_2 = d - v$. Therefore $(n_1 - d_1) + (n_2 - d_2) = d$. Since both x_1 and x_2 are singleton-free, $n_1 - d_1 > 0$ and $n_2 - d_2 > 0$. Hence $n_1 - d_1 < d$ and $n_2 - d_2 < d$, and therefore $\sigma_d(2d) = \boldsymbol{s}(x) \le \boldsymbol{s}(x_1) + \boldsymbol{s}(x_2) \le \sigma_{d_1}(n_1) + \sigma_{d_2}(n_2) \le (n_1 - d_1) + (n_2 - d_2) = d$. □

Since the number of distinct squares in a singleton-free $(d, 2d)$-string is at most d, we do not need to consider singleton-free strings. Moving a singleton to the end of a string does not decrease the number of distinct squares, therefore we shall only consider $(d, 2d)$-strings that have singletons at the end. We can set $\sigma_d^-(2d) = \sigma_{d-1}(2d-2)+1$ and thus consider only the strings that have the non-singleton part dense. By Lemma 12 we need only to consider strings whose s-covers of the non-singleton part satisfy the parity condition with no consecutive adjacent squares. Moreover, the number of singletons must be at least $\lceil \frac{2d}{3} \rceil$, see [4]. Let $\mathcal{T}_v$ denote the set of all singleton-free $\sigma_d^-(2d)$-dense $(d - \lceil \frac{2d}{3} \rceil, 2d - \lceil \frac{2d}{3} \rceil)$-strings admitting an s-cover satisfying the parity condition with no consecutive adjacent squares. Then we set

$$\sigma_d(2d) = \max\,\{d,\ \max_{x \in \mathcal{T}_v} \boldsymbol{s}(x)\}.$$

7 Additional properties of $\boldsymbol{\sigma_d(n)}$

Though fundamental properties of $\sigma_d(n)$ were presented previously in [4], here we present some additional properties concerning the gaps between consecutive values in the $(d, n-d)$ table where the value of $\sigma_d(n)$ is the entry on the d-th row and the $(n-d)$-th column. Lemma 13, respectively Lemma 14, shows that the difference between any two consecutive entries along a row, respectively between any two consecutive entries on the main diagonal, in the $(d, n-d)$ table is bounded by 2.

Lemma 13. *For any $2 \leq d \leq n$, $\sigma_d(n+1) - \sigma_d(n) \leq 2$.*

Proof. Let $(d, n+1)$-string $x = x[1 \mathinner{\ldotp\ldotp} n+1]$ be square-maximal, then $\boldsymbol{s}(x) = \sigma_d(n+1)$. Without a loss of generality we can assume that the first symbol of x is not a singleton – otherwise we can move all singletons from the beginning of x to the end of x without destroying any square type. Let $y = x[2 \mathinner{\ldotp\ldotp} n+1]$. Then y is a (d, n)-string and $\boldsymbol{s}(y) \leq \sigma_d(n)$. By Fraenkel-Simpson [5], there are at most two rightmost occurrences of squares starting at the same position in a string. In other words, the removal of $x[1]$ destroyed at most two square types. That is, $\boldsymbol{s}(x) - 2 \leq \boldsymbol{s}(y)$. Therefore, $\sigma_d(n+1) - 2 \leq \boldsymbol{s}(y) \leq \sigma_d(n)$, implying $\sigma_d(n+1) - \sigma_d(n) \leq 2$. □

Lemma 14. *For any $2 \leq d$, $\sigma_{d+1}(2d+2) - \sigma_d(2d) \leq 2$.*

Proof. By Lemma 13, $\sigma_{d+1}(2d+2) - \sigma_{d+1}(2d+1) \leq 2$. By the results from [4], the entries under and on the main diagonal along a column are constant; that is, $\sigma_{d+1}(2d+1) = \sigma_d(2d)$. Therefore, $\sigma_{d+1}(2d+2) - \sigma_d(2d) \leq 2$. □

Lemma 15. *For any $d \geq 2$, if there is a square-maximal singleton free $(d, 2d+1)$-string x, then there exists a square-maximal $(d, 2d+1)$-string y of the form $y = aaabbccdd....$*

Proof. Since x contains no singletons, then x contains exactly $d-1$ pairs and 1 triple. To prove there exists a square-maximal string in the form that all pairs consist of adjacent symbols and the triple also consists of adjacent symbols, we need to show the non-adjacent symbols can be moved together without reducing the number of distinct squares. Let us suppose that there is a non-adjacent pair of $\boldsymbol{c}$'s in x.

(i) If the $\boldsymbol{c}$'s did not occur in any square, then we could move both $\boldsymbol{c}$'s to the end of the string without destroying an square type. Moreover, we would gain a new square $\boldsymbol{cc}$, contradicting the square-maximality of x.
(ii) If the $\boldsymbol{c}$'s occur in exactly one square $u\boldsymbol{c}vu\boldsymbol{c}v$, where u and v are some strings, we can move both $\boldsymbol{c}$'s to the end of x to form a new string y. The new squares created by this move are $uvuv$ and $\boldsymbol{cc}$ while the old square $u\boldsymbol{c}vu\boldsymbol{c}v$ was destroyed. If $uvuv$ did not exist in any other part of x, then $\boldsymbol{s}(y) > \boldsymbol{s}(x)$ which contradicts the square-maximality of x; thus $uvuv$ already existed in some other part of x, so we lost the square $u\boldsymbol{c}vu\boldsymbol{c}v$, but gained $\boldsymbol{cc}$, so $\boldsymbol{s}(y) = \boldsymbol{s}(x)$.
(iii) If the $\boldsymbol{c}$'s occur in more than one square, these squares must form a non-trivial run, i.e. a run with a non-empty tail. Since there is only one symbol $\boldsymbol{t}$ occurring in x 3 times, the only form of such a non-trivial run can be $\boldsymbol{t}u\boldsymbol{c}v\boldsymbol{t}u\boldsymbol{c}v\boldsymbol{t}$. If $u = v = \varepsilon$, then the run is $\boldsymbol{tctct}$ containing two distinct squares $\boldsymbol{tctc}$ and $\boldsymbol{ctct}$. We can change it to $\boldsymbol{tttcc}$, destroying the two squares $\boldsymbol{tctc}$ and $\boldsymbol{ctct}$, but gaining two new squares $\boldsymbol{tt}$ and $\boldsymbol{cc}$. If either $u \neq \varepsilon$ or $v \neq \varepsilon$, then by moving both $\boldsymbol{c}$'s to the end of x, we destroy the two distinct squares $\boldsymbol{t}u\boldsymbol{c}v\boldsymbol{t}u\boldsymbol{c}v$ and $u\boldsymbol{c}v\boldsymbol{t}u\boldsymbol{c}v\boldsymbol{t}$, but gain three new squares $\boldsymbol{t}uv\boldsymbol{t}uv$, $uv\boldsymbol{t}uv\boldsymbol{t}$, and $\boldsymbol{cc}$. Note that neither $\boldsymbol{t}uv\boldsymbol{t}uv$ nor $uv\boldsymbol{t}uv\boldsymbol{t}$ can exist anywhere else in x for the lack of $\boldsymbol{t}$'s. Thus we have more distinct squares than x, which contradicts the maximality of x.
Since we can move safely all pairs together to the end of x, the symbols of the triple will end up also adjacent at the beginning of the string. □

Lemma 16 shows that the two entries of the $(d, n-d)$ table in the same column just above the main diagonal must be identical.

Lemma 16. *For any* $3 \leq d$, $\sigma_d(2d+1) = \sigma_{d-1}(2d)$.

Proof. We prove it by induction. Let (H_d) be the statement that $\sigma_d(2d+1) = \sigma_{d-1}(2d)$. (H_d) for $2 \leq d \leq 10$ is true from the values in the $(d, n-d)$ table computed so far, see [4]. This takes care of the base case of the induction. Thus let us assume that H_{d-1} is true, and let us prove that (H_d) is true. Let $(d, 2d+1)$-string x be square-maximal. If x contains a singleton, remove it to form a new $(d-1, 2d)$-string y. Then $\sigma_d(2d+1) = \boldsymbol{s}(x) \leq \boldsymbol{s}(y) \leq \sigma_{d-1}(2d)$ and since $\sigma_d(2d+1) \geq \sigma_{d-1}(2d)$, see [4], thus $\sigma_d(2d+1) = \sigma_{d-1}(2d)$. If x contains no singletons, by Lemma 15 we can assume that it has the form $aaabbccdd$.... Remove a pair from z forming a new $(d-1, 2d-1)$-string y. Then $\sigma_d(2d+1) - 1 = \boldsymbol{s}(x) - 1 = \boldsymbol{s}(y) \leq \sigma_{d-1}(2d-1)$ and since $\sigma_d(2d+1) - 1 \geq \sigma_{d-1}(2d-1)$ by [4], therefore $\sigma_d(2d+1) = \sigma_{d-1}(2d-1) + 1$. Since H_{d-1}, $\sigma_{d-1}(2d) \geq \sigma_{d-2}(2d-2) + 1$ and $\sigma_d(2d+1) \geq \sigma_{d-1}(2d)$ according to [4], hence $\sigma_d(2d+1) = \sigma_{d-1}(2d)$. □

Corollary 17 demonstrates the fact that the difference between any two consecutive entries on the two diagonals immediately above the main diagonal of the $(d, n-d)$ table is also bounded by 2.

Corollary 17. *For any* $3 \leq d$, $\sigma_d(2d+1) - \sigma_{d-1}(2d-1) \leq 2$ *and* $\sigma_d(2d+2) - \sigma_{d-1}(2d) \leq 2$.

Proof. By Lemma 13, $\sigma_{d-1}(2d) - \sigma_{d-1}(2d-1) \leq 2$, and by Lemma 16, $\sigma_{d-1}(2d) = \sigma_d(2d+1)$. Therefore, $\sigma_d(2d+1) - \sigma_{d-1}(2d-1) \leq 2$. Similarly, $\sigma_d(2d+2) - \sigma_d(2d+1) \leq 2$ by Lemma 13, and $\sigma_d(2d+1) = \sigma_{d-1}(2d)$ by Lemma 16. Therefore, $\sigma_d(2d+2) - \sigma_{d-1}(2d) \leq 2$. □

Remark 18. Fraenkel-Simpson [5] gave the upper bound of $2n - 8$ for $n \geq 5$ and any d, and $\sigma_2(n) \leq 2n - 29$ for $n \geq 22$. Ilie [8] provided an asymptomatic bound of $2n - \Theta(\log n)$. We slightly improve Fraenkel-Simpson's bounds with: for any $2 \leq d \leq n$ and $n \geq d_0 + 2$, $\sigma_d(n) \leq 2n - d_0 - 2d$, where d_0 is the maximum d such that $\sigma_d(2d) = d$ is known. Currently, $d_0 = 20$. In addition, since $\sigma_2(53) = 40$ we get $\sigma_2(n) \leq 2n - 66$ for $n \geq 53$. Similarly, since $\sigma_3(41) = 30$, $\sigma_4(31) = 22$, $\sigma_5(33) = 23$, $\sigma_6(28) = 17$, $\sigma_7(27) = 17$, $\sigma_8(23) = 13$, $\sigma_9(23) = 12$ and $\sigma_{10}(23) = 11$, we get $\sigma_3(n) \leq 2n - 52$ for $n \geq 41$, $\sigma_4(n) \leq 2n - 40$ for $n \geq 31$, $\sigma_5(n) \leq 2n - 43$ for $n \geq 33$, $\sigma_6(n) \leq 2n - 39$ for $n \geq 28$, $\sigma_7(n) \leq 2n - 37$ for $n \geq 27$, $\sigma_8(n) \leq 2n - 33$ for $n \geq 23$, $\sigma_9(n) \leq 2n - 34$ for $n \geq 23$ and $\sigma_{10}(n) \leq 2n - 35$ for $n \geq 23$.

Proof. By Lemma 14, $\sigma_d(n) \leq d_0 + 2k$, where $n - d = d_0 + k$ and $k \geq 1$. Thus $\sigma_d(d_0 + k + d) \leq d_0 + 2k = 2(d_0 + k + d) - d_0 - 2d$. Therefore, $\sigma_d(n) \leq 2n - d_0 - 2d$ for $n \geq d_0 + 2$. □

8 Computational Results

We implemented the described algorithms in C++, and ran the programs in parallel on the SHARCNET computer cluster. We were able to compute all $\sigma_2(n)$ values for $n \leq 53$ in a matter of hours. The 10 largest new values are: $\sigma_2(44) = 33$, $\sigma_2(45) = 34$, $\sigma_2(46) = 35$, $\sigma_2(47) = 36$, $\sigma_2(48) = 36$, $\sigma_2(49) = 37$, $\sigma_2(50) = 37$, $\sigma_2(51) = 38$, $\sigma_2(52) = 39$ and $\sigma_2(53) = 40$. The results and sample square-maximal strings may be found at [3]. Whenever the computation required determining the number of distinct primitively rooted squares in a concrete string, a C++ implementation of the Franek, Jiang, and Weng's algorithm [7] was used. The values of interest include: three consecutive equal values: $\sigma_2(31) = \sigma_2(32) = \sigma_2(33)$, the unexpected existence of pairs (d, n) satisfying $\sigma_{d+1}(n + 2) - \sigma_d(n) > 1$ such as (2,33) and (2,34), and $\sigma_2(33) < \sigma_3(33)$; that is, among all strings of length 33, no binary string achieves the maximum number of distinct primitively rooted squares.

9 Conclusion

We presented the notion of s-cover as a structural generalization of a uniform distribution of squares in a string. We showed that it is sufficient to consider special strings admitting an s-cover in order to recursively determine the maximum number of distinct primitively rooted squares $\sigma_d(n)$. Based on these observations, we presented an efficient computational framework with significantly reduced search space for computations of $\sigma_d(n)$ based on the notion of density and exploiting the tightness of the available lower bound. We used an implementation of this algorithm to obtain the previously unknown values of $\sigma_d(n)$, and in particular $\sigma_2(n)$ up to $n = 53$.

Acknowledgments

This work was supported by the Natural Sciences and Engineering Research Council of Canada and by the Canada Research Chair program, and made possible by the facilities of the Shared Hierarchical Academic Research Computing Network (`http://www.sharcnet.ca/`).

References

[1] A. Baker, A. Deza, and F. Franek. On the structure of run-maximal strings. *Journal of Discrete Algorithms* (to appear).

[2] A. Baker, A. Deza, and F. Franek. Computational framework for determining run-maximal strings. AdvOL-Report 2011/06, McMaster University, 2011.

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

[4] A. Deza, F. Franek, and M. Jiang. A *d*-step approach for distinct squares in strings. *Lecture Notes in Computer Science*, 5029:77–89, 2011.

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

[6] F. Franek and J. Holub. A different proof of Crochemore-Ilie lemma concerning microruns. In *London Algorithmics 2008: Theory and Practice*, pages 1–9. College Publications, London, UK, 2009.

[7] F. Franek, M. Jiang, and C. Weng. An improved version of the runs algorithm based on crochemore's partitioning algorithm. In *Proceedings of Prague Stringology Conference 2011*, pages 98–105. Prague, Czech Republic, August 2011.

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