

McMaster University

Advanced Optimization Laboratory



Title:

A computational framework for determining run-maximal strings

Authors:

Andrew Baker, Antoine Deza, and Frantisek Franek

AdvOL-Report No. 2011/6

December 2011, Hamilton, Ontario, Canada

A computational framework for determining run-maximal strings *

Andrew Baker, Antoine Deza, and Frantisek Franek

December 10, 2011

Abstract

We investigate the function $\rho_d(n) = \max\{ \mathbf{r}(x) \mid x \text{ is a } (d, n)\text{-string} \}$, where $\mathbf{r}(x)$ denotes the number of runs in a string x and (d, n) -string denotes a string of length n with exactly d distinct symbols. The notion of r-cover is presented and discussed with emphasis on recursive computational determination of $\rho_d(n)$. Thus, r-covers can be used as a computational framework for an efficient computation of the maximum number of runs.

1 Introduction

In [2] 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 run-maximal strings. The straightforward assertion from [2] that a run-maximal string has an r-cover – except possibly a single weak point – holds only when the size of the alphabet is not kept fixed. However, the approach can be adapted inductively to handle situations with fixed alphabets and can be used to speed up computations of the maximum number of runs.

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 $e = s + 2p - 1$. Similarly, we encode a run as a triple (s, e, p) . Note that the exponent of such a run equals $\lfloor \frac{e-s+1}{p} \rfloor$ and the tail of the run equals the remainder of the division of $(e-s+1)$ by p . The *leading square* of a run (s, e, p) refers to the square $(s, s+2p-1, p)$. It is always clear from the context if a triple (s, e, p) encodes a square or a run.

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 $i_1 < j_1$ and $j_1 \leq i_k + 1$ and then $x[i_1 .. i_k] \cup x[j_1 .. j_m] = x[i_1 .. j_m]$, or if $j_1 < i_1$ and $i_1 \leq j_m + 1$ and then $x[i_1 .. i_k] \cup x[j_1 .. j_m] = x[j_1 .. i_k]$. Simply put, the join is defined when the two substrings either are adjacent or overlap. The alphabet of x is denoted by $\mathcal{A}(x)$, (d, n) -string refers to a string of length n with exactly d distinct symbols, $\mathbf{r}(x)$ denotes the number of runs in a string x , and $\rho_d(n)$ refers to the maximum number of runs over all (d, n) -strings, i.e. $\rho_d(n) = \max\{ \mathbf{r}(x) \mid x \text{ is a } (d, n)\text{-string} \}$. $d(x)$ denotes the number

*Supported by the Natural Sciences & Engineering Research Council of Canada and by the Canada Research Chair program.

of distinct symbols of a string x . A singleton is a symbol which occurs exactly once in the string under consideration.

A square (s, e, p) is *left-shiftable* if $x[s-1]$ is defined ($s > 1$), and $x[s-1] = x[s+p-1]$. Similarly, a square is *right-shiftable* if $x[e+1]$ is defined ($e < n$) and $x[e+1] = x[s]$. In other words, a square (s, e, p) is left-shiftable exactly when $(s-1, e-1, p)$ is also a square, and is right-shiftable exactly when $(s+1, e+1, p)$ is also a square.

2 Computational approach to runs

First we provide a definition of an r -cover, and then use it to significantly reduce the search space containing all run-maximal strings.

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

- (1) none of the S_i 's, $2 \leq i \leq m$ are left-shiftable;
- (2) for any $1 \leq i < m$, $s_i < s_{i+1}$ and $d_i \leq s_{i+1}+1$, i.e. two consecutive squares are either adjacent or overlap;
- (3) $\bigcup_{1 \leq i \leq m} S_i = x$;
- (4) for any run (s, e, p) of x there is $1 \leq i \leq m$ so that S , the leading square of the run is a substring of S_i , denoted by $S \subseteq S_i$.

A string which has an r -cover is referred to as r -covered.

Lemma 2. If a string x is r -covered, its r -cover is unique.

Proof. Let us assume that we have two different r -covers of x , $\{ S_i \mid 1 \leq i \leq m \}$ and $\{ S'_j \mid 1 \leq j \leq k \}$. We shall prove by induction that they are identical.

By Definition 1 (4), $S_1 \subseteq S'_1$ 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 $1 \leq i \leq t$. If $\bigcup_{1 \leq i \leq t} S_i = x$, we have $t = m = k$ and we are done. Otherwise consider S_{t+1} . By Definition 1 (4), 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 (4). Therefore $S_{t+1} \subseteq S'_{t+1}$. By the same argument, $S'_{t+1} \subseteq S_{t+1}$ and so $S_{t+1} = S'_{t+1}$. \square

Lemma 3. If a string x has an r -cover, then it is singleton free.

Proof. Let $\{ S_j \mid 1 \leq j \leq m \}$ be the r -cover of $x = x[1 .. n]$. For any $1 \leq i \leq n$, $x[i] \in S_t$ for some t by Definition 1 (3). Since S_t is a square, the symbol $x[i]$ occurs in x at least twice. \square

A notion of density of runs can be informally viewed as the property that a removal of any symbol from a string destroys certain number of runs. However, the removal of a symbol from a string and the concatenation of the two parts may cause some runs to merge together. Thus we need a more precise notion to describe the notion of a symbol $x[i]$ *destroying* a certain number of runs.

Definition 4. We say that the symbol $x[i]$ destroys k runs in x , if

- (a) $i = 1$ and $\mathbf{r}(x) - \mathbf{r}(x[2 \dots n]) = k$, or
- (b) $i = n$ and $\mathbf{r}(x) - \mathbf{r}(x[1 \dots n-1]) = k$, or
- (c) $1 < i < n$ and $\mathbf{r}(x) - \mathbf{r}(x[i \dots i-1]) - \mathbf{r}(x[i+1 \dots n]) = k$.

A singleton-free (d, n) -string x is t -dense, $t \geq 1$, if

- (a) $x[1]$ destroys strictly more than $t - \rho_d(n-1)$ runs in x ;
- (b) $x[i]$, $1 < i < n$, destroys strictly more than $t - \mathbf{r}(x[1 \dots i-1]) - \rho_{d'}(n-i)$ runs in x , where $d' = |\mathcal{A}(x[1 \dots n]) - \mathcal{A}(x[1 \dots i])| \leq d$;
- (c) $x[n]$ destroys strictly more than $t - \mathbf{r}(x[1 \dots n-1])$ runs in x .

Lemma 5. If a singleton-free (d, n) -string is not t -dense, then $\mathbf{r}(x) \leq t$.

Proof. Since x is not t -dense, either $x[1]$ destroys $k \leq t - \rho_d(n-1)$ runs in x , in which case $\mathbf{r}(x) \leq k + \mathbf{r}(x[2 \dots n]) \leq k + \rho_d(n-1) \leq t$. Or $x[i]$, for some $1 < i < n$, destroys $k \leq t - \mathbf{r}(x[1 \dots i-1]) - \rho_{d'}(n-i)$ runs in x , where $d' = |\mathcal{A}(x[1 \dots n]) - \mathcal{A}(x[1 \dots i])| \leq d$, in which case $\mathbf{r}(x[1 \dots n]) \leq \mathbf{r}(x[1 \dots i-1]) + k + \mathbf{r}(x[i+1 \dots n]) \leq \mathbf{r}(x[1 \dots i-1]) + k + \rho_{d'}(n-i) \leq t$. Or $x[n]$ destroys $k \leq t - \mathbf{r}(x[1 \dots n-1])$ runs in x , in which case $\mathbf{r}(x[1 \dots n]) \leq k + \mathbf{r}(x[1 \dots n-1]) \leq t$. \square

Lemma 6. Let x be a (d, n) -string. If any $x[i]$ destroys at least one run in x , then x has an r -cover.

Proof. We build an r -cover by induction:

Since the $x[1]$ destroys at least one run in x , there must be at least one run starting at position 1. Among all runs starting at position 1, set S_1 to the leading square of the one with the largest period.

Suppose that we have built the r -cover up to $i \leq t$. If $\bigcup_{1 \leq i \leq t} S_i = x$, we are done. Otherwise $\bigcup_{1 \leq i \leq m} S_i = x[1 \dots v]$ for some $v < n$. If $v = n-1$, then $x[v+1] = x[n]$ destroys at least one run in x . Therefore, there is at least one run (s, e, p) in x so that $s \leq v+1 \leq s+2p-1$. From all such runs chose the leftmost ones, and set S_{t+1} to the leading square of the one among them with the largest period.

It is straightforward to verify that all the conditions of Definition 1 are satisfied and that we have build the r -cover of x . \square

Note that for an (d, n) -string having an r -cover implies being singleton free, however it does not imply that every $x[i]$ destroys at least one run, even though it is very close to it. Consider the r -cover $\{ S_i = (s_i, e_i, p_i) \mid 1 \leq i \leq m \}$ of x . If S_1 is right-shiftable and there is no other run in x starting at position 1, then $x[1]$ does not destroy any run. A similar situation occurs for two adjacent non-overlapping squares S_i and S_{i+1} in the r -cover: if S_{i+1} is right-shiftable and there is no other run in x starting at position s_{i+1} , then $x[s_{i+1}]$ does not destroy any run. In this sense, r -cover is a computationally efficient structural generalization of the property that every $x[i]$ destroys at least one run.

The notion of density was designed so it can be checked using only knowledge of an initial segment of a string and use the knowledge of the $\rho_d(n)$ function for the previous values of n . The following lemma shows how the density of an r -covered string x can be verified incrementally using the members of the r -cover.

Lemma 7. *Let $\{ S_i = (s_i, e_i, p_i) \mid 1 \leq i \leq m \}$ be the r -cover of an (d, n) -string x . Then x is t -dense if and only if*

$$(a) \text{ for any } 1 \leq i < m, \text{ for any } s_i \leq j < s_{i+1}, x[j] \text{ destroys strictly more than } k \text{ runs in } x, \\ \text{where } k = \begin{cases} t - \rho_d(e_i - 1) & \text{if } j = 0 \\ t - \mathbf{r}(x[1 \dots j-1]) - \rho_{d'}(e_i - j) & \text{if } 0 < j \text{ and } d' = |\mathcal{A}(x[1 \dots j]) - \mathcal{A}(x[1 \dots n])| \end{cases}$$

$$(b) \text{ for any } s_m \leq j \leq e_m, x[j] \text{ destroys strictly more than } k \text{ runs in } x, \text{ where} \\ k = \begin{cases} t - \mathbf{r}(x[1 \dots j-1]) - \rho_{d'}(n - j) & \text{if } j < n \text{ and } d' = |\mathcal{A}(x[1 \dots j]) - \mathcal{A}(x[1 \dots n])| \\ t - \mathbf{r}(x[1 \dots n-1]) & \text{if } j = n \end{cases}$$

Proof. Straightforward from item (4) of Definition 1 and Definition 4. \square

Lemma 8. *Let $\rho_d(n_1) + \rho_d(n_2) \leq \rho_d(n_1 + n_2)$ for any $n_1 + n_2 = n - 1$. If a singleton-free run-maximal (d, n) -string x does not have an r -cover, then $\rho_d(n) = \rho_d(n - 1)$.*

Proof. Since there is no r -cover of x , there is an $x[i]$ that is not in any run and not a singleton. If $i = 1$, then $\rho_d(n) = \mathbf{r}(x[1 \dots n]) = \mathbf{r}(x[2 \dots n]) \leq \rho_d(n - 1) \leq \rho_d(n)$. If $i = n$, then $\rho_d(n) = \mathbf{r}(x[1 \dots n]) = \mathbf{r}(x[1 \dots n - 1]) \leq \rho_d(n - 1) \leq \rho_d(n)$. Otherwise consider substrings $x[1 \dots i - 1]$ and $x[i + 1 \dots n]$. If $\mathcal{A}(x[1 \dots i - 1]) = \mathcal{A}(x[i + 1 \dots n])$, then $\rho_d(n) = \mathbf{r}(x[1 \dots n]) = \mathbf{r}(x[1 \dots i - 1]) + \mathbf{r}(x[i + 1 \dots n]) \leq \rho_d(i - 1) + \rho_d(n - i - 1) \leq \rho_d(n - 1) \leq \rho_d(n)$.

If $\mathcal{A}(x[1 \dots i - 1]) \neq \mathcal{A}(x[i + 1 \dots n])$ there are two cases:

There is $\mathbf{c} \in \mathcal{A}(x[1 \dots i - 1]) - \mathcal{A}(x[i + 1 \dots n])$. In this case, permute the alphabet of $x[1 \dots i - 1]$ so that $x[i - 1] = \mathbf{c}$. Then $x[1 \dots i - 1]$ and $x[i + 1 \dots n]$ can be concatenated without merging any runs. Therefore, $\rho_d(n) = \mathbf{r}(x[1 \dots n]) = \mathbf{r}(x[1 \dots i - 1]) + \mathbf{r}(x[i + 1 \dots n]) = \mathbf{r}(x[1 \dots i - 1]x[i + 1 \dots n]) \leq \rho_d(n - 1) \leq \rho_d(n)$.

There is $\mathbf{c} \in \mathcal{A}(x[i + 1 \dots n]) - \mathcal{A}(x[1 \dots i - 1])$. In this case, permute the alphabet of $x[i + 1 \dots n]$ so that $x[i + 1] = \mathbf{c}$. Then $x[1 \dots i - 1]$ and $x[i + 1 \dots n]$ can be concatenated without merging any runs. As in the previous case, $\rho_d(n) = \rho_d(n - 1)$. \square

Lemma 9. *If a run-maximal (d, n) -string has a singleton, then either $\rho_d(n) = \rho_d(n - 1)$ or $\rho_d(n) = \rho_{d-1}(n - 1)$.*

Proof. For a given run-maximal (d, n) -string x there are three cases. The first case is when x has a singleton at the end or the beginning. If it is at the end, then $\rho_d(n) = \mathbf{r}(x) = \mathbf{r}(x[1 .. n-1]) \leq \rho_{d-1}(n-1)$ as $x[1 .. n-1]$ is a $(d-1, n-1)$ -string. It follows that $\rho_d(n) = \rho_{d-1}(n-1)$. For a singleton at the beginning the proof is identical.

The second case is when x has a singleton in the middle at a position j and the alphabets of the two parts are different, i.e. there is \mathbf{c} so that either $\mathbf{c} \in \mathcal{A}(x[1 .. j-1]) - \mathcal{A}(x[j+1 .. n])$ or $\mathbf{c} \in \mathcal{A}(x[j+1 .. n]) - \mathcal{A}(x[1 .. j-1])$. If $\mathbf{c} \in \mathcal{A}(x[j+1 .. n]) - \mathcal{A}(x[1 .. j-1])$, then create x_1 by permuting the alphabet of $x[j+1 .. n]$ so that \mathbf{c} moves to the position $j+1$. This will not affect the number of runs and so $\mathbf{r}(x) = \mathbf{r}(x_1)$. Create x_2 by moving the singleton $x[j]$ to the end. Again, the number of runs will not be affected and so $\mathbf{r}(x_1) = \mathbf{r}(x_2)$. Then $y = x_2[1 .. n-1]$ is a $(d-1, n-1)$ -string and $\rho_d(n) = \mathbf{r}(x) = \mathbf{r}(x_2) = \mathbf{r}(y) \leq \rho_{d-1}(n-1) \leq \rho_d(n)$. The argument is similar if $\mathbf{c} \in \mathcal{A}(x[1 .. j-1]) - \mathcal{A}(x[j+1 .. n])$.

The third case is when x has a singleton \mathbf{c} in the middle at a position j and the alphabets of the two parts are the same, i.e. $\mathcal{A}(x[j+1 .. n]) = \mathcal{A}(x[1 .. j-1])$. It follows that $\mathcal{A}(x[j+1 .. n]) = \mathcal{A}(x[1 .. j-1]) = \mathcal{A}(x) - \{\mathbf{c}\}$. Replace all occurrences of $x[j+1]$ in $x[j+1 .. n]$ with the singleton \mathbf{c} , producing x_1 . This will not affect any runs and so $\mathbf{r}(x) = \mathbf{r}(x_1)$. Moreover, $\mathcal{A}(x) = \mathcal{A}(x_1)$. Remove $x_1[j]$ producing a string x_2 . Since no runs are merged, $\mathbf{r}(x_1) = \mathbf{r}(x_2)$. Since $\mathcal{A}(x) = \mathcal{A}(x_2)$, x_2 is a $(d, n-1)$ -string and thus $\rho_d(n) = \mathbf{r}(x) = \mathbf{r}(x_2) \leq \rho_d(n-1)$. \square

Corollary 10. *Let x be a run-maximal $(d, 2d)$ -string, $d \geq 2$. If x has any singletons, they can all be moved to the beginning or the end of the string without affecting the number of runs in x .*

Proof. Let x have a singleton in the middle. If it is case (b) from the proof of Lemma 9, then the singleton can be moved to the beginning or the end of x without changing the number of runs. If it is case (c) from the proof of Lemma 9, then $\rho_d(2d) = \rho_d(2d-1)$ by Lemma 9, which is impossible as $\rho_d(2d) > \rho_{d-1}(2d-2)$. \square

3 Heuristic for a lower bound $\rho_d^-(n)$

Let $\rho_d^-(n)$ denote a lower bound for $\rho_d(n)$. The higher the value of $\rho_d^-(n)$ that we can compute easily, the less computational effort must be spent on determining $\rho_d(n)$. For $d = 2$, generate $\mathcal{L}_2(n)$, the set of (d, n) -strings which are: r-covered, balanced over every prefix (the frequencies of a 's and b 's differ by at most a predefined constant), and contain no triples (aaa or bbb). Then

$$\rho_d^-(n) = \max \{ \rho_2(n-1), \rho_2(n-2)+1, \max_{x \in \mathcal{L}_2(n)} \mathbf{r}(x) \}.$$

This heuristic was found to be very good when tested against the known run-maximal strings for $\rho_2(n)$: Franek & Smyth up to 34, and Kolpakov & Kucherov up to 60. Note that $\rho_2^-(25) < \rho_2(25)$ since the only run-maximal $(2, 25)$ -string contains a triple. For $d \geq 3$, we simply set $\rho_d^-(n) = \rho_{d-1}(n)+1$.

4 Generating r-covered (d, n) -strings

Rather than generating strings, we generate their r-covers. The generation proceeds by extending the partially built r-cover in all possible ways. Every time a potential square of the r-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 r-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 r-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 lexicographic order. The generation of the r-cover follows these principles:

The generator for the first square and any other square in the r-cover that is adjacent to the previous square is generated by iterative calls to `Next()` producing all the possible generators. Each generator is checked for the additional properties (must be primitive, is not left-shiftable, did not create an intermediate square in the intermediate string, etc.) before it is accepted. For subsequent square, if the new square is overlapping the previous 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, Lemma 7 is used.

5 Recursive computation of $\rho_d(n)$

First it is verified that $\rho_d(n_1) + \rho_d(n_2) \leq \rho_d(n-1)$ for any $n_1 + n_2 = n-1$. Then $\rho_d^-(n)$ is computed by the heuristic of Section 3. Then \mathcal{R} , the set of all $\rho_d^-(d)$ -dense r-covered (d, n) -strings is generated as described in Section 4.

Consider the existence of a run-maximal (d, n) -string with singletons. By Lemma 9, $\rho_d(n) = \rho_d(n-1)$ or $\rho_{d-1}(n-1)$.

Consider the existence of a singleton-free run-maximal string $x \notin \mathcal{R}$. Either x does not have an r-cover, in which case by Lemma 8, $\rho_d(n) = \rho_d(n-1)$, or x has an r-cover and is not $\rho_d^-(n)$ -dense, in which case by Lemma 5, $\rho_d(n) \leq \rho_d^-(n)$. Therefore

$$\rho_2(n) = \max \{ \rho_d^-(n), \rho_d(n-1), \rho_{d-1}(n-1), \max_{x \in \mathcal{R}_k} \mathbf{r}(x) \}.$$

Note that $\mathcal{R} \subseteq \{ x \mid x \text{ is a } (d, n)\text{-string} \}$ and from the computational results, the decrease in size is very significant, in many cases \mathcal{R} is empty.

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

For computation of values on the main diagonal we can use r-covers satisfying additional conditions.

Definition 11. The r -cover $\{ S_i = (s_i, e_i, p_i) \mid 1 \leq i \leq m \}$ of $x = x[1 .. n]$ satisfies the parity condition if for any $1 \leq i < m$, $\mathcal{A}(x[1 .. e_i-1]) \cap \mathcal{A}(x[s_{i+1}+1 .. n]) \subseteq \mathcal{A}x[s_{i+1} .. e_i]$.

Lemma 12. The singleton-free part of a run-maximal $(d, 2d)$ -string x with all its singletons at the end has an r -cover satisfying the parity condition.

Proof. Let x have $v \leq d-2$ singletons, all at the end. Let us assume that $x[i]$ for some $1 \leq i \leq 2d-v$ does not destroy a run. If $i = 1$ or $2d-v$, then $\rho_d(2d) = \mathbf{r}(x) = \rho_d(2d-1) = \rho_{d-1}(2d-2)$, a contradiction. Therefore $1 < i < 2d-v$. If $\mathcal{A}(x[1 .. i-1]) = \mathcal{A}(x[i+1 .. 2d-v])$, then $x[1 .. i-1]$ must have d distinct symbols and no singletons, hence $i-1 \geq 2d$, a contradiction. Therefore there is \mathbf{c} so that either $\mathbf{c} \in \mathcal{A}(x[1 .. i-1]) - \mathcal{A}(x[i+1 .. 2d-v])$ or $\mathbf{c} \in \mathcal{A}(x[i+1 .. 2d-v]) - \mathcal{A}(x[1 .. i-1])$. Similarly as in the proof of Lemma 8, $\rho_d(2d) \leq \rho_d(2d-1) = \rho_{d-1}(2d-2)$, a contradiction.

So every $x[i]$ destroys at least one run and thus x has an r -cover $\{ S_i \mid 1 \leq i \leq m \}$ by Lemma 6. Let us assume that the r -cover does not satisfy the parity condition. There are two cases and both yield a contradiction:

(i) $\bigcup_{1 \leq i \leq t} S_i$ and $\bigcup_{t+1 \leq j \leq m} S_j$ for some $1 \leq t \leq m$ are disjoint and have at least one symbol in common, say \mathbf{c} . If we replace all \mathbf{c} 's in $\bigcup_{1 \leq i \leq t} S_i$ by a new symbol $\hat{\mathbf{c}} \notin \mathcal{A}(x)$, we get a new $(d+1, n)$ -string y so that $\mathbf{r}(y) = \mathbf{r}(x)$. Thus $\rho_{d-1}(2d-2) = \rho_d(2d-1) = \rho_{d+1}(2d) \geq \mathbf{r}(y) = \mathbf{r}(x) = \rho_d(2d)$, a contradiction.

(ii) $\bigcup_{1 \leq i \leq t} S_i$ and $\bigcup_{t+1 \leq j \leq m} S_j$ for some $1 \leq t \leq m$ are overlapping, and there is a symbol \mathbf{c} occurring in $\bigcup_{1 \leq i \leq t} S_i$ and in $\bigcup_{t+1 \leq j \leq m} S_j$, but not in the overlap $S_t \cap S_{t+1}$. If we replace all \mathbf{c} 's in $\bigcup_{1 \leq i \leq t} S_i$ by a new symbol $\hat{\mathbf{c}} \notin \mathcal{A}(x)$, we get a new $(d+1, n)$ -string y so that $\mathbf{r}(y) = \mathbf{r}(x)$. Thus $\rho_{d-1}(2d-2) = \rho_d(2d-1) = \rho_{d+1}(2d) \geq \mathbf{r}(y) = \mathbf{r}(x) = \rho_d(2d)$, a contradiction. \square

With additional assumptions, the previous lemma can be strengthened to exclude non-overlapping squares from the r -cover.

Lemma 13. Let $\rho_{d'}(2d') = d'$ for any $d' < d$. Either $\rho_d(2d) = d$ or for every run-maximal $(d, 2d)$ -string x with v singletons all at the end, $v \leq d-2$, its singleton-free part $x[1 .. 2d-v]$ has an r -cover satisfying the parity condition and which has no adjacent non-overlapping squares.

Proof. The existence of the r -cover $\{ S_i \mid 1 \leq i \leq m \}$ of $x[1 .. 2d-v]$ satisfying the parity condition follow from Lemma 12. We need to prove that either $\rho_d(2d) = d$ or there are no adjacent squares in the r -cover. Since $\rho_{d'}(2d') = d'$ for any $d' < d$, $\rho_{d'}(n') \leq n' - d'$ for any $n' - d' < d$. Let us assume that the r -cover of x has two adjacent squares S_t and S_{t+1} . Let $x_1 = \bigcup_{1 \leq i \leq t} S_i$ and let $x_2 = \bigcup_{t < i \leq m} S_i$. Then $\mathbf{r}(x) = \mathbf{r}(x_1) + \mathbf{r}(x_2)$ and x_1 is a (d_1, n_1) -string for some d_1 and n_1 , and x_2 is a (d_2, n_2) -string for some d_2 and n_2 , where $n_1 + n_2 = 2d$ and $d_1 + d_2 \geq d$. Since the r -cover satisfies the parity condition, $\mathcal{A}(x_1)$ and $\mathcal{A}(x_2)$ are disjoint and hence $d_1 + d_2 = d$. Therefore $(n_1 - d_1) + (n_2 - d_2) = d$. Since $x[1 .. 2d-v]$ is singleton free, x_1 is singleton free, and thus $n_1 - d_1 > 0$ and x_2 has some non-singletons,

and thus $n_2 - d_2 > 0$. It follows that both $n_1 - d_1$ and $n_2 - d_2$ are smaller than d . Therefore $\rho_d(2d) = \mathbf{r}(x) = \mathbf{r}(x_1) + \mathbf{r}(x_2) \leq \rho_{d_1}(n_1) + \rho_{d_2}(n_2) \leq (n_1 - d_1) + (n_2 - d_2) = d$. \square

Since the number of runs in a singleton-free $(d, 2d)$ -string is at most d , we do not need to consider the singleton-free strings. By Corollary 10, we can consider only $(d, 2d)$ -strings that have singletons at the end. Since $\rho_d(2d) > \rho_{d-1}(2d-2)$, we can set $\rho_d^-(2d) = \rho_{d-1}(2d-2) + 1$ and thus consider only the strings that have the non-singleton part $\rho_d^-(2d)$ -dense. By Lemma 13 we can only consider strings whose \mathbf{r} -covers of the non-singleton part satisfy the parity condition with no adjacent non-overlapping squares. Moreover, by the result in [1], we know that the number of singletons must be at least $\lceil \frac{7d}{8} \rceil$. For every $\lceil \frac{7d}{8} \rceil \leq v \leq d-2$, let \mathcal{T}_v denote the set of all singleton-free $\rho_d^-(2d)$ -dense \mathbf{r} -covered $(d, 2d-v)$ -strings whose \mathbf{r} -covers satisfy the parity condition and have no adjacent non-overlapping squares. Then

$$\rho_d(2d) = \max \left(d, \max \left\{ \max_{x \in \mathcal{T}_v} \mathbf{r}(x) : \lceil \frac{7d}{8} \rceil \leq v \leq d-2 \right\} \right)$$

7 Conclusion

We first presented the notion of \mathbf{r} -covers as a structural generalization of a uniform distribution of runs in a string. Then we showed that only for \mathbf{r} -covered strings we do not know the exact value of the maximum number of runs and hence only \mathbf{r} -covered strings must be examined to determine $\rho_d(n)$. Based on these observations, we presented a fast and efficient computational framework with significantly reduced search space for computations of $\rho_d(n)$ based on the notion of density. Let us remark that whenever the computation required determining the number of runs in a concrete string, the C++ implementation of the Franek, Jiang, and Weng's algorithm [3] was used.

References

- [1] A. Baker, A. Deza, and F. Franek. A parameterized formulation for the maximum number of runs problem. *Journal of Discrete Algorithms* (submitted) and AdvOL-Report 2011/02, McMaster University, 2011.
- [2] A. Baker, A. Deza, and F. Franek. On the structure of run-maximal strings. *Journal of Discrete Algorithms*, 14:10–14, 2012.
- [3] 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, 2011.