Computing (Sub-)Cadences with Three Elements by Using Polygonal Convolutions

Mitsuru Funakoshi[1,2,a)] Julian Pape-Lange[3]

Abstract: The discrete acyclic convolution computes the $2n + 1$ sums

$$\sum_{\substack{i+j=k \\ (i,j)\in[0,1,2,\ldots,n]^2}} a_i b_j$$

in $O(n \log n)$ time. By using suitable offsets and setting some of the variables to zero, this method provides a tool to calculate all non-zero sums

$$\sum_{\substack{i+j=k \\ (i,j)\in P\cap\mathbb{Z}^2}} a_i b_j$$

in a rectangle P with perimeter p in $O(p \log p)$ time.
This paper extends this geometric interpretation in order to allow arbitrary convex polygons P with k vertices and perimeter p. Also, this extended algorithm only needs $O\left(k + p(\log p)^2 \log k\right)$ time.
Additionally, this paper presents fast algorithms for counting sub-cadences and cadences with 3 elements using this extended method.

Keywords: discrete acyclic convolutions, string-cadences, geometric algorithms, number theoretic transforms

1. Introduction

The convolution is a well-known and very useful method, which is not only closely linked to signal processing (e.g. [10]) but is also used to multiply polynomials (see [2], p. 905) and large numbers (e.g. [9] (written in German)) in quasi-linear time. The convolution can be efficiently computed with the fast Fourier transform or its counterpart in residue class rings, the number theoretic transform:

Theorem 1. *Let $a = (a_0, a_1, a_2, \ldots, a_n)$ and $b = (b_0, b_1, b_2, \ldots, b_n)$ be two sequences. The sequence $c = (c_0, c_1, c_2, \ldots, c_{2n})$ with $c_k = \sum_{i+j=k} \left(a_i b_j\right)$ can be computed in $O(n \log n)$ operations.*

The most well-known proofs use additions and multiplications of arbitrary complex numbers. However, with the finite register lengths of real-world computers, one must either cope with the roundoff errors or do all calculations in a different ring. In Appendix of a full version of this paper [4], we show that a suitable ring can be found deterministically in $O\left(n(\log n)^2(\log\log n)\right)$ time if the generalized Riemann hypothesis is true.

The convolution can also be interpreted geometrically: Let $a = (a_0, a_1, a_2, \ldots, a_n)$ and $b = (b_0, b_1, b_2, \ldots, b_n)$ be sequences. Then the convolution calculates the partial sums

$$\sum_{\substack{i+j=k \\ (i,j)\in P\cap\mathbb{Z}^2}} a_i b_j,$$

where P is the square given by $\{(x, y) : 0 \le x, y \le n\}$.

This paper extends this geometric interpretation and shows that if P is an arbitrary convex polygon with k vertices and perimeter p, the partial sums can be calculated in $O\left(k + p(\log p)^2 \log k\right)$ time.

We also use this extended method to solve an open problem of a string pattern called cadence. A cadence is given by an arithmetic progression of occurrences of the same character in a string such that the progression cannot be extended to either side without extending the string as well. For example, in the string 001001001 the indices $(3, 6, 9)$ corresponding to the "1"s form a 3-cadence. On the other hand, in the string 001010100 the indices $(3, 5, 7)$ corresponding to the "1"s do not form a 3-cadence since, for example, the index 1 is still inside of the string.

3-cadences can be found naïvely in quadratic time. In the paper [1], a quasi-linear time algorithm for detecting the existence of 3-cadences was proposed, but this algorithm also detects false positives as the aforementioned string 001010100.

This paper fixes this issue and also extends the algorithm to the slightly more general notion of (a, b, c)-partial-k-cadences. The resulting extended algorithm also allows counting those partial-cadences with a given character of an alphabet Σ of a string with length n and only needs $O\left(n(\log n)^2\right)$ time. Using a method presented by Amir et al. in [1], this implies that all (a, b, c)-partial-k-cadences can be counted in $O\left(\min(|\Sigma| n(\log n)^2, n^{3/2} \log n)\right)$ time.

[1] Department of Informatics, Kyushu University
[2] Japan Society for the Promotion of Science
[3] Technische Universität Chemnitz, Straße der Nationen 62, 09111 Chemnitz, Germany
[a)] mitsuru.funakoshi@inf.kyushu-u.ac.jp

Furthermore, we show that the output of the counting algorithm also allows finding x partial-cadences in $O(xn)$ time.

This paper also gives similar results for 3-sub-cadences.

For the time complexity, we assume that arithmetic operations with $O(\log n)$ bits can be done in constant time. In particular, we want to be able to get the remainder of a division by a prime $p < 2(2n\log(2n))^2$ in constant time.

Also, in this paper, we assume a suitable alphabet. I.e. the characters are given by sufficiently small integers in order to allow constant time reading of a given character in the string and in order to allow sorting the characters.

Recent Work on (Sub-)Cadences

Recently, Funakoshi et al. [3] presented counting / locating algorithms for k-(sub-)cadences for arbitrary k. Furthermore, Pape-Lange [8] very recently proposed detecting algorithms for 3-cadences in grammar-compressed / uncompressed binary strings. In the same work [8], he proved that the problems of detecting several variants of cadences in grammar-compressed strings are NP-complete.

2. (Sub-)Cadences and Their Definitions

While the concept of cadences in the context of strings was already considered in [11] by Van der Waerden, the term cadence dates back to 1964 and was first introduced by Gardelle and Guilbaud in [5] (written in French). Since then, there were at least two other, slightly different and non-equivalent definitions given by Lothaire in [7], Chapter 3.3 and Amir et al. in [1].

This paper uses the most restrictive definition of the cadence, which was introduced by Amir et al. in [1], and also uses their definition of the sub-cadence, which is equivalent to Gardelle's cadence in [5] and Lothaire's arithmetic cadence in [7], Chapter 3.3.

A string S of length n is the concatenation $S = S[1..n] = S[1]S[2]S[3]\dots S[n]$ of characters from an alphabet Σ.

Definition 1. *A* k-sub-cadence *is a triple* (i,d,k) *of positive integers such that*

$$S[i] = S[i+d] = S[i+2d] = \cdots = S[i+(k-1)d]$$

holds.

In this paper, cadences are additionally required to start and end close to the boundaries of the string:

Definition 2. *A* k-cadence *is a* k*-sub-cadence* (i,d,k) *such that the inequalities* $i-d \le 0$ *and* $n < i+kd$ *hold.*

Since for any k-sub-cadence the inequality $i+(k-1)d \le n$ holds, for any k-cadence $i+(k-1)d \le n < i+kd$ holds. This implies $k-1 \le \frac{n-i}{d} < k$ and thereby $k = \left\lfloor \frac{n-i}{d} \right\rfloor + 1$. It is therefore sufficient to omit the variable k of a k-cadence (i,d,k) and just denote this k-cadence by the pair (i,d).

Remark 1 (Comparison of the Definitions)**.**

- *The cadence as defined by Lothaire is just an ordered sequence of unequal indices such that the corresponding characters are equal.*
- *The cadence as defined by Gardelle and Guilbaud additionally requires the sequence to be an arithmetic sequence.*
- *The cadence as defined by Amir et al. and as used in this paper additionally requires that the cadence cannot be extended in any direction without extending the string as well.*

For the analysis of cadences with errors, we need two more definitions:

Definition 3. *A* k-cadence with at most m errors *is a tuple* (i,d,k,m) *of integers such that* $i,d,k \ge 1$ *and* $i-d \le 0$ *and* $n < i+kd$ *hold and such that there are* $k-m$ *different integers* $\pi_j \in \{0,1,2,\dots,k-1\}$ *with* $j = 1,2,3,\dots,k-m$ *and*

$$S[i+\pi_1 d] = S[i+\pi_2 d] = S[i+\pi_3 d] \cdots = S[i+\pi_{k-m} d].$$

A particularly interesting case of cadences with errors is given by the partial-cadences in which we know all positions where an error is allowed:

Definition 4. *For some different integers* $\pi_j \in \{0,1,2,\dots,k-1\}$ *with* $j = 1,2,3,\dots,p$, *a* $(\pi_1,\pi_2,\pi_3,\dots,\pi_p)$-partial-$k$-cadence *is a triple* (i,d,k) *of positive integers with* $i-d \le 0$ *and* $n < i+kd$ *such that*

$$S[i+\pi_1 d] = S[i+\pi_2 d] = S[i+\pi_3 d] \cdots = S[i+\pi_p d]$$

hold.

In this paper, we will only consider the case of $k-3$ errors. I.e. *k-cadences with at most $k-3$ errors* and *(a,b,c)-partial-k-cadences* for three different integer $a,b,c \in \{0,1,...,k-1\}$.

3. 3-Sub-Cadences and Rectangular Convolutions

It is a direct consequence of van der Waerden's theorem that sufficiently large strings are guaranteed to have sub-cadences of a given length:

Theorem 2 (Existence of sub-cadences (Van der Waerden [11] (written in German), see Lothaire [7], Chapter 3.3))**.**

Let Σ *be an alphabet and* k *an integer. There exists an integer* $N = N(|\Sigma|,k)$ *such that every string containing at least* N *characters has at least one* k*-sub-cadence*

However, this theorem does not provide the number of k-sub-cadences of a given string.

In this section, we will show that 3-sub-cadences with a given character of a string of length n can be efficiently counted in $O(n\log n)$ time. We will also show that arbitrary 3-sub-cadences of a string of length n can be counted in $O\left(n^{3/2}(\log n)^{1/2}\right)$ time and that both counting algorithms allow us to output x different 3-sub-cadences in $O(xn)$ additional time if at least x different 3-sub-cadences exist.

Let $\sigma \in \Sigma$ be a character. We will now count all 3-sub-cadences with character σ.

Let (i,d) be a 3-sub-cadence. Since $i+d = \frac{i+(i+2d)}{2}$ holds, the position $i+d$ of the middle occurrence of σ only depends on the sum of the index i of first occurrence and the index $i+2d$ of the third occurrence but does not depend on the individual indices of those two positions. Therefore, it is possible to determine the candidates for the middle occurrences with the convolution of the candidates of the first occurrence and the candidates of the third occurrence.

Let the sequence $\delta = (\delta_0,\delta_1,\delta_2,\dots,\delta_n)$ be given by the indicator function for σ in S:

$$\delta_i := \begin{cases} 1 & \text{if } S[i] = \sigma \\ 0 & \text{if } S[i] \neq \sigma \text{ (this includes } i = 0) \end{cases}$$

With this definition, the product $\delta_i\delta_j$ is 1 if and only if $S[i] = S[j] = \sigma$ and otherwise is 0. Therefore $c_k = \sum_{i+j=k} \left(\delta_i\delta_j\right) = \#\{i : S[i] = S[k-i] = \sigma\}$ counts in how many ways the index $\frac{k}{2}$ lies in the middle of two σ. These partial sums can be calculated in $O(n \log n)$ time by convolution.

If k is odd or $S\left[\frac{k}{2}\right] \neq \sigma$ holds, the index $\frac{k}{2}$ cannot be the middle index of a 3-sub-cadence. If $S\left[\frac{k}{2}\right] = \sigma$ holds, the indicator function $\delta_{\frac{k}{2}}$ is 1, and therefore $\delta_{\frac{k}{2}}\delta_{\frac{k}{2}} = 1$ holds as well. Since the arithmetic progression $(\delta_{\frac{k}{2}}, 0, 3)$ consisting of three times the number $\delta_{\frac{k}{2}}$ is not a 3-sub-cadence, the output element c_k contains one false positive. Additionally, for $i + j = k$ with $i \neq j$ and $S[i] = S[j] = \sigma$, the output element c_k counts the combination $\delta_i\delta_j$ as well as $\delta_j\delta_i$.

Therefore,

$$s_k := \begin{cases} \frac{c_{2k}-1}{2} & \text{if } S[k] = \sigma \\ 0 & \text{if } S[k] \neq \sigma \end{cases}$$

counts exactly the number of 3-sub-cadences with character σ such that the second occurrence of σ has index k. The sum of the s_k is the number of total 3-sub-cadences with character σ.

Also, for each $s_k \neq 0$, all those s_k 3-sub-cadences can be found in $O(k) \subseteq O(n)$ time by checking for each index $i < k$ whether $S[i] = S[k] = S[2k - i] = \sigma$ holds.

If the character σ is rare, we can also follow the idea of Amir et al. in [1] for detecting 3-cadences with rare characters: If all n_σ occurrences of the character are known, the c_k can be computed in $O(n_\sigma^2)$ time by computing every pair of those occurrences. Therefore:

Theorem 3. *For every character $\sigma \in \Sigma$, the 3-sub-cadences with σ can be counted in $O(n \log n)$ time. Also, if all n_σ occurrences of σ are known, the 3-sub-cadences with σ can be counted in $O(n_\sigma^2)$ time.*

Following the proof in [1], we can get all occurrences of every character by sorting the input string in $O(n \log n)$ time. This implies that the algorithm needs at most $O\left(\sum_{\sigma\in\Sigma} \min(n_\sigma^2, n \log n)\right) \subseteq O\left(\frac{n}{(n \log n)^{1/2}} n \log n\right) = O(n^{3/2}(\log n)^{1/2})$ time.

Theorem 4. *The number of all 3-sub-cadences can be counted in*

$$O\left(\min(|\Sigma| n \log n, n^{3/2}(\log n)^{1/2})\right) \text{ time.}$$

Theorem 5. *After counting at least x 3-sub-cadences, it is possible to output x 3-sub-cadences in $O(xn)$ time.*

4. Non-Rectangular Convolutions

In this section, we will extend the geometric interpretation of the convolution and show that for convex polygons P with k vertices and perimeter p it is possible to calculate the partial sums

$$c_k = \sum_{\substack{i+j=k \\ (i,j)\in P\cap\mathbb{Z}^2}} a_i b_j$$

in $O\left(k + p(\log p)^2 \log k\right)$ time.

Let us imagine a graph where all integer coordinates (i, j) have the value $f(i, j) := a_i b_j$. We do not need the convolution in order to determine the sum of the function values in a given rectangle since we can use the simple factorization $\sum_{i=0}^{n} \sum_{j=0}^{m} \left(a_i b_j\right) = \left(\sum_{i=0}^{n} a_i\right)\left(\sum_{j=0}^{m} b_j\right)$ in $O(n + m)$ time. However, the convolution provides the $2n$ partial sums on the 45°-diagonals in almost the same time of $O((n + m) \log(n + m))$.

We will now extend this geometric interpretation firstly to triangles with a vertical cathetus and a horizontal cathetus, then to arbitrary triangles and lastly to convex polygons. In order to do this, we will cover the given polygon P in polygons P_p^+ and P_m^- such that for each integer point (i, j) the equality

$$\#\{P_p^+|(i, j) \in P_p^+\} - \#\{P_m^-|(i, j) \in P_m^-\} = \begin{cases} 1 & \text{if } (i, j) \in P \\ 0 & \text{if } (i, j) \notin P \end{cases}$$

holds, and we define

$$(c_p)_k := \sum_{\substack{i+j=k \\ (i,j)\in P_p^+\cap\mathbb{Z}^2}} a_i b_j \text{ and } (c_m)_k := -\sum_{\substack{i+j=k \\ (i,j)\in P_m^-\cap\mathbb{Z}^2}} a_i b_j.$$

By construction, $c_k = \left(\sum (c_p)_k\right) + \left(\sum (c_m)_k\right)$ holds. However, if the edges and vertices of the polygons P_p^+ and P_m^- contain integer points, we need to carefully decide for every of these polygons, which edges and vertices are supposed to be included in the polygons and which are excluded from the polygons.

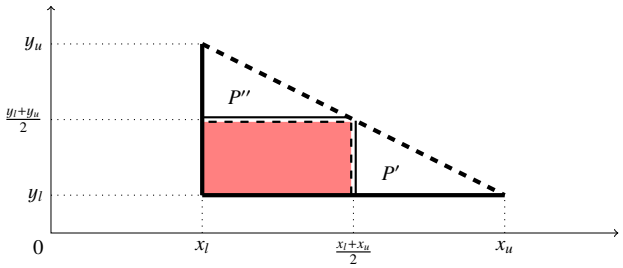

Fig. 1 The right-angled triangle P in Lemma 1.

Lemma 1. *Let P be a triangle with a vertical cathetus and a horizontal cathetus and perimeter p. Let also the sequences $a = (a_0, a_1, a_2, \ldots, a_n)$ and $b = (b_0, b_1, b_2, \ldots, b_n)$ be given.*

Then the partial sums

$$c_k = \sum_{\substack{i+j=k \\ (i,j)\in P\cap\mathbb{Z}^2}} a_i b_j$$

can be calculated in $O\left(p(\log p)^2\right)$ time.

Proof. The proof will be symmetrical with regard to horizontal and vertical mirroring. Therefore, without loss of generality, we will assume that P is oriented as in Figure 1.

We first initialize the output vector $c = (c_{x_l+y_l}, c_{x_l+y_l+1}, c_{x_l+y_l+2}, \ldots, c_{x_u+y_u})$ with zero. This takes $O(p)$ time.

In the following proof, we assume that both catheti are included in the polygon and that the hypotenuse as well as its endpoints are excluded. If this is not the expected behavior, we can traverse the edges in $O(p)$ time and for each integer point (i, j) on the edge, we can decrease/increase the corresponding c_{i+j} by $a_i b_j$ if necessary.

If p is at most one, there is at most one integer point (i, j) in the

triangle, and this point can be found in constant time. In this case, we only have to increase c_{i+j} by $a_i b_j$.

If p is bigger than one, we will separate the triangle P into three disjoint parts as seen in Figure 1.

- The triangle P' of points with x-coordinate of at least $\left\lceil \frac{x_l+x_u}{2} \right\rceil$,
- the triangle P'' of points with y-coordinate of at least $\left\lceil \frac{y_l+y_u}{2} \right\rceil$ and
- the red rectangle of points with x-coordinate of at most $\left\lceil \frac{x_l+x_u}{2} \right\rceil - 1$ and y-coordinate of at most $\left\lceil \frac{y_l+y_u}{2} \right\rceil - 1$.

There are no integers bigger than $\left\lceil \frac{x_l+x_u}{2} \right\rceil - 1$ but smaller than $\left\lceil \frac{x_l+x_u}{2} \right\rceil$ nor integers bigger than $\left\lceil \frac{y_l+y_u}{2} \right\rceil - 1$ but smaller than $\left\lceil \frac{y_l+y_u}{2} \right\rceil - 1$. Therefore, each integer point in P is in exactly one of the three parts.

For the red rectangle, we can calculate the convolution and thereby get the corresponding partial sums in $O(p \log p)$ time. The partial sums corresponding to the sub-triangles are calculated recursively. Increasing the c_k by the partial results leads to the final result.

Hence, the algorithm takes

$$O\left(p + \left(\sum_{i=0}^{\log_2 p} 2^i \left(\frac{p}{2^i} \log \frac{p}{2^i}\right)\right) + 2^{\log_2 p}\right) \subseteq O\left(\sum_{i=0}^{\log p} p \log p\right)$$
$$= O\left(p(\log p)^2\right)$$

time. □

We will now further extend this result to arbitrary triangles:

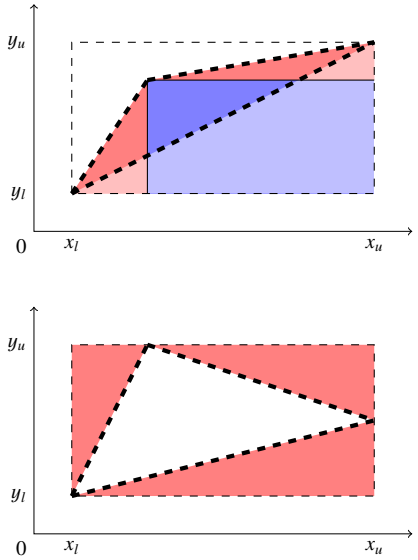

Fig. 2 The two possible triangles P in Lemma 2.

Lemma 2. *Let a triangle P with perimeter p and sequences $a = (a_0, a_1, a_2, \ldots, a_n)$ and $b = (b_0, b_1, b_2, \ldots, b_n)$ be given.*

Then the partial sums

$$c_k = \sum_{\substack{i+j=k \\ (i,j) \in P \cap \mathbb{Z}^2}} a_i b_j$$

can be calculated in $O\left(p(\log p)^2\right)$ time.

Proof. Let x_l, y_l, x_u, y_u be the minimal and maximal x-coordinates and y-coordinates of the three vertices of the polygon P. As in the last lemma, we first initialize the output vector $c = (c_{x_l+y_l}, c_{x_l+y_l+1}, c_{x_l+y_l+2}, \ldots, c_{x_u+y_u})$.

Similarly to the last lemma, we can remove/add edges and vertices in linear time with respect to p. Since the number of edges and vertices is constant, we ignore them for the sake of simplicity.

Let R be the rectangle $\{(x, y) | x_l < x < x_u \wedge y_l < y < y_u\}$. Since R has four edges but P only has three vertices, at least one of the vertices of P is also a vertex of R. Without loss of generality, this vertex is (x_l, y_l).

Case 1: The opposing vertex (x_u, y_u) in R also coincides with a vertex of P (as in the upper part of Figure 2):
Without loss of generality, we can assume that the third vertex of P is above the diagonal from (x_l, y_l) to (x_u, y_u). In this case, the partial sums corresponding to P are given by the sum of the partial sums of the red triangles and the partial sums of the blue rectangle minus the partial sums of the lighter triangle. There are only three triangles and one rectangle involved, and each of those polygons has perimeter $O(p)$. Furthermore, all triangles have a vertical cathetus and a horizontal cathetus. Therefore, using Lemma 1, we can calculate all partial sums in $O\left(p(\log p)^2\right)$ time.

Case 2: The opposing vertex (x_u, y_u) in R does not coincide with a vertex of P (as in the lower part of Figure 2):
In this case, one vertex of P lies on the right edge of R and one vertex of P lies on the upper edge of R.
The wanted partial sums are in this case the difference of the partial sums of the rectangle and of the partial sums of the three red triangles. Again, we can calculate all partial sums in $O\left(p(\log p)^2\right)$ time.

Since both cases require $O\left(p(\log p)^2\right)$ time, this concludes the proof. □

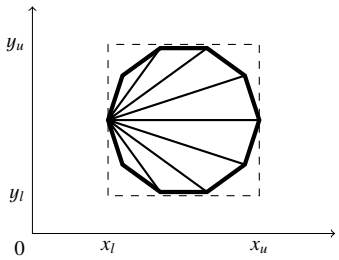

Fig. 3 A regular k-gon. All chords from the leftmost vertex to the vertices on the right-hand side of the k-gon are at least $\frac{p}{4}$ long. The sum of all chords' lengths is therefore $\Theta(kp)$

Now we will extend this algorithm to convex polygons with k vertices by dissecting them into $k-2$ triangles by adding $k-3$ chords. Since the time complexity of the triangular convolution given by Lemma 2 depends on the sum of the triangles' perimeters, it is not sufficient to just select one vertex and connect it with every other vertex in the polygon (see Figure 3). On the other hand, the triangulation algorithm itself should not take longer than the convolutions. Additionally, the order in which the chords are added does not matter for the convolutions. We will show that for convex polygons there is a triangulation which can be computed in linear time and only increases the perimeter by the factor $O(\log k)$.

Theorem 6. *Let P be a convex polygon with k vertices and*

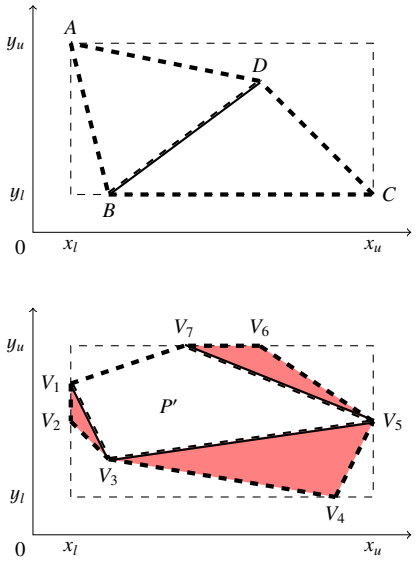

Fig. 4 Two possible convex polygons P with more than 3 vertices in Lemma 6.

perimeter p. Let also the sequences $a = (a_0, a_1, a_2, \ldots, a_n)$ and $b = (b_0, b_1, b_2, \ldots, b_n)$ be given.

Then the partial sums

$$c_k = \sum_{\substack{i+j=k \\ (i,j)\in P\cap\mathbb{Z}^2}} a_i b_j$$

can be calculated in $O\left(k + p(\log p)^2 \log k\right)$ time.

Proof. As in the last two Lemmata, we define x_l, y_l, x_u, y_u to be the minimal and maximal x-coordinates and y-coordinates of the k vertices of P. Also, we first initialize the output vector $c = (c_{x_l+y_l}, c_{x_l+y_l+1}, c_{x_l+y_l+2}, \ldots, c_{x_u+y_u})$. We further assume that none of the edges and vertices of P is included in P.

If P is a triangle, then this Lemma simplifies to Lemma 2 and there is nothing left to prove.

If P is a quadrilateral $ABCD$, as in the upper part of Figure 4, then it can be partitioned into the triangles ABD and CDB where the edge BD is included in exactly one triangle and all other edges are excluded. The triangle inequality proves that $|BD| \le |DA|+|AB|$ and $|BD| \le |BC| + |CD|$ hold. Therefore, both triangles have a perimeter of at most p. This implies that the partial sums can be calculated in $O\left(p(\log p)^2\right)$

If P is a polygon $V_1V_2V_3\ldots V_k$ with more than four vertices, as in the lower part of Figure 4, it can be partitioned into

- the polygon $P' = V_1V_3V_5\ldots V_{2\lceil\frac{k}{2}\rceil-1}$, which is given by the odd vertices without its edges,
- the red triangles $V_iV_{i+1}V_{i+2}$ with $i = 1, 3, 5, \ldots, 2\left\lceil\frac{k}{2}\right\rceil - 3$ including the edge V_iV_{i+2} but excluding the other edges and the vertices,
- if k is even, the triangle $V_{k-1}V_kV_1$ including the edge $V_{k-1}V_1$ but excluding the other edges and the vertices.

By construction and triangle inequality, the perimeter p' of P' is at most p. This, however, also implies that the total perimeter $\sum p_i$ of the triangles is at most $2p$. The inequality

$$\sum \min\left(1, p_i(\log p_i)^2\right) \le k + \sum\left(p_i(\log p)^2\right) \le k + p(\log p)^2$$

implies that the algorithm needs $O\left(k + p(\log p)^2\right)$ time plus the time we need for processing P'. Since each step almost halves the number of vertices, we need $O(\log k)$ steps. This results in a total time complexity of $O\left(k + p(\log p)^2 \log k\right)$. □

5. (a,b,c)-Partial-k-Cadences

In this section, we will show how the non-rectangular convolution helps counting the (a, b, c)-partial-k-cadences as defined in Definition 4.

In particular, we will show that (a, b, c)-partial-k-cadences with a given character σ can be counted in $O\left(n(\log n)^2\right)$ time. We will further show that all (a, b, c)-partial-k-cadences can be counted in $O\left(\min(|\Sigma|n(\log n)^2, n^{3/2}\log n)\right)$ time and that both counting algorithms allow us to output x of those partial-cadences in $O(xn)$ time.

As a special case, these results also hold for 3-cadences.

We further conclude from these results that the existence of k-cadences with at most $k-3$ errors as defined in Definition 3 can be detected in $O\left(\min(|\Sigma|k^3n(\log n)^2, k^3n^{3/2}\log n)\right)$ time.

Without loss of generality, we will only deal with the case $a < b$ in this section.

Lemma 3. *Three positions x, y and z form a (a, b, c)-partial-k-cadence if and only if*

- *the equation $\frac{y-x}{b-a} = \frac{z-y}{c-b} \in \mathbb{Z}$ holds,*
- *the equation $S[x] = S[y] = S[z]$ holds and*
- *the inequalities*

$$0 \ge \frac{(b+1)x - (a+1)y}{b-a}, \tag{1}$$

$$0 < \frac{bx - ay}{b-a}, \tag{2}$$

$$n \ge \frac{(b-k+1)x - (a-k+1)y}{b-a} \text{ and} \tag{3}$$

$$n < i + kd = \frac{(b-k)x - (a-k)y}{b-a} \text{ hold.} \tag{4}$$

Proof. Define $d := \frac{y-x}{b-a}$ and $i := x - ad$. Then $x = i + ad$ and $y = i + bd$. Furthermore, the equation $\frac{y-x}{b-a} = \frac{z-y}{c-b}$ holds if and only if $z = i + cd$ and $\frac{y-x}{b-a} \in \mathbb{Z}$ holds if and only if d is an integer.

Additionally, using $x = i + ad$ and $y = i + bd$, the four inequalities can be simplified to $0 \ge i - d$, $0 < i$, $n \ge i + (k-1)d$ and $n < i + kd$.

Therefore, the lemma follows from the definition of the partial-cadence. □

The four inequalities hold if the points (x, y) lie inside the convex quadrilateral given, as shown in Figure 5, by the corners

$$A = \left(\frac{an}{k}, \frac{bn}{k}\right)$$

$$B = \left(\frac{(a+1)n}{k+1}, \frac{(b+1)n}{k+1}\right)$$

$$C = \left(\frac{(a+1)n}{k}, \frac{(b+1)n}{k}\right)$$

$$D = \left(\frac{an}{k-1}, \frac{bn}{k-1}\right)$$

including the vertex C and the edges between B and C as well as

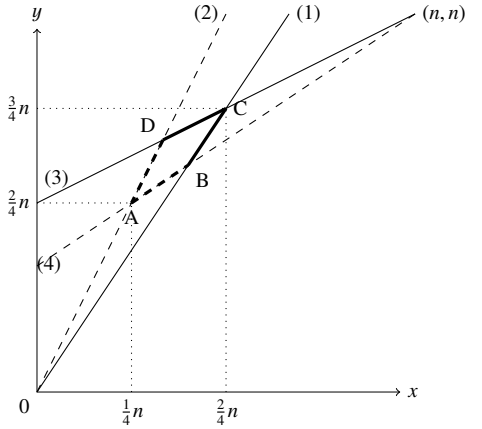

Fig. 5 The four inequalities of Lemma 3 for $(1, 2, 3)$-partial-4-cadences.

between C and D but excluding all other vertices and the edges between A and B as well as between D and A.

For given $x = i+ad$ and $y = i+bd$, the third occurrence $z = i+cd$ can be calculated with the equation $i + cd = \frac{(b-c)(i+ad)+(c-a)(i+bd)}{b-a}$ directly without calculating i and d first. The corresponding partial sums

$$c_k = \sum_{\substack{i+j=k \\ (i,j)\in P\cap\mathbb{Z}^2}} a_{\frac{i}{(b-c)}} b_{\frac{j}{(c-a)}}$$

can be calculated by using the partial sums

$$c_k = \sum_{\substack{i+j=k \\ (i,j)\in P'\cap\mathbb{Z}^2}} a'_i b'_j$$

with $a'_i := \begin{cases} a_{\frac{i}{b-c}} & \text{if } i \equiv 0 \pmod{b-c} \\ 0 & \text{otherwise} \end{cases}$ and $b'_j := \begin{cases} b_{\frac{j}{c-a}} & \text{if } j \equiv 0 \pmod{c-a} \\ 0 & \text{otherwise} \end{cases}$ and a polygon P', which is derived from P by stretching the first coordinate by $(b-c)$ and the second coordinate by $(c-a)$. The perimeter of P' is at most $\max(|b-c|, |c-a|)$ times the perimeter of P. Using the quadrilateral $P = ABCD$ with perimeter

$$\begin{aligned} p &\le 2|C_x - A_x| + 2|C_y - A_y| \\ &= 2\left(\frac{(a+1)n}{k} - \frac{an}{k}\right) + 2\left(\frac{(b+1)n}{k} - \frac{bn}{k}\right) = \frac{4n}{k} \in O\left(\frac{n}{k}\right), \end{aligned}$$

the polygon P' has perimeter $p' \in O(n)$. This proves the following three theorems.

Theorem 7. *For every character $\sigma \in \Sigma$, the (a, b, c)-partial-k-cadences with σ can be counted in $O(n(\log n)^2)$ time. Also, if all n_σ occurrences of σ are known, the (a, b, c)-partial-k-cadences with σ can be counted in $O(n_\sigma^2)$ time.*

Theorem 8. *The number of all (a, b, c)-partial-k-cadences can be counted in*

$$O\left(\min(|\Sigma|n(\log n)^2, n^{3/2}\log n)\right) \text{ time.}$$

Theorem 9. *After counting at least x (a, b, c)-partial-k-cadences, it is possible to output x (a, b, c)-partial-k-cadences in $O(xn)$ time.*

Since every 3-cadence is an $(0, 1, 2)$-partial-3-cadence, we also obtain the special case:

Corollary 1. *For every character $\sigma \in \Sigma$, the 3-cadences with σ can be counted in $O(n(\log n)^2)$ time. Also, if all n_σ occurrences of σ are known, the 3-cadences with σ can be counted in $O(n_\sigma^2)$ time.*

Therefore, the number of all 3-cadences can be counted in

$$O\left(\min(|\Sigma|n(\log n)^2, n^{3/2}\log n)\right) \text{ time.}$$

Also, after counting at least x 3-cadences, it is possible to output x 3-cadences in $O(xn)$ time.

Taking the sum over all possible triples (a, b, c), we can also search for k-cadences with at most $k-3$ errors. It can be checked in

$$O\left(\min(|\Sigma|k^3 n(\log n)^2, k^3 n^{3/2}\log n)\right)$$

time whether the given string has a k-cadence with at most $k-3$ errors. However, since k-cadences with less than $k-3$ errors are counted more than once, it seems to be difficult to determine the exact number of k-cadences with at most $k-3$ errors.

6. Conclusion

This paper extends convolutions to arbitrary convex polygons. One might wonder whether these convolutions could be sped up or be further extended to non-convex polynomials.

Instead of just partitioning the interior of the polygon into triangles, it is also possible to identify polygons by the difference of a slightly bigger but less complex polygon and a triangle. However, if the algorithm presented in this paper is adapted to non-convex polygons, it can generate self-intersecting polygons. While the time-complexity stays the same for these polygons, it becomes hard to ensure that every vertex and every edge of the polygon is counted exactly once.

Another approach is given by Levcopoulos and Lingas in [6]. This paper shows that any simple polygon can be decomposed into convex components in $O(k \log k)$ time while only increasing the total perimeter by the factor $O(\log k)$. This paper also shows that if the input polygon is rectilinear, this partition only contains axis-aligned rectangles. Since the convolution handles rectangles quicker and more easily than triangles, this saves a logarithm. However, in general, it is not obvious how to transform arbitrary polygons into equivalent simple rectilinear polygons in quasilinear time without blowing-up the number of vertices too much.

The non-rectangular convolution, unlike the usual convolution, allows us to define a dependence between the indices of the convoluted sequences. This dependence is not usable in applications like the multiplication of polynomials, and for many signal processing applications this extended method does not seem to bring any benefits either. However, in order to count the partial-cadences this dependence was essential. The non-rectangular convolution may also have future applications in image processing and convolutional neural networks.

In terms of cadences, this paper presents algorithms to count and find sub-cadences, cadences and partial-cadences with three elements. However, if there are linearly many c-positions of (a, b, c)-partial-k-cadences, the knowledge of those partial-cadences does not lead to a sub-quadratic-time-algorithm for determining the existence 4-cadences. On the other hand, it is also not shown that this problem needs quadratic time.

Also, the time-complexity $O(xn)$ for finding x 3-cadences is quite pessimistic. If there are many 3-cadences, it is very likely that quite a few of these 3-cadences share one of their occurrences. These occurrences can be found in $O(n)$ time. On the other hand, in the string $10^{n-1}1^{2n}$, for example, there are linearly many 3-cadences but every second occurrence and every third occurrence only occurs in at most one of those 3-cadences.

References

[1] Amir, A., Apostolico, A., Gagie, T. and Landau, G. M.: String cadences, *Theoretical Computer Science*, Vol. 698, pp. 4 – 8 (2017). Algorithms, Strings and Theoretical Approaches in the Big Data Era (In Honor of the 60th Birthday of Professor Raffaele Giancarlo).

[2] Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C.: *Introduction to Algorithms, 3rd Edition*, MIT Press (2009).

[3] Funakoshi, M., Nakashima, Y., Inenaga, S., Bannai, H., Takeda, M. and Shinohara, A.: Detecting k-(Sub-)Cadences and Equidistant Subsequence Occurrences, *31st Annual Symposium on Combinatorial Pattern Matching, CPM 2020, June 17-19, 2020, Copenhagen, Denmark*, LIPIcs, Vol. 161, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 12:1–12:11 (2020).

[4] Funakoshi, M. and Pape-Lange, J.: Non-Rectangular Convolutions and (Sub-)Cadences with Three Elements, *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France*, LIPIcs, Vol. 154, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 30:1–30:16 (2020).

[5] Gardelle, J.: Cadences, *Mathématiques et Sciences humaines*, Vol. 9, pp. 31–38 (1964).

[6] Levcopoulos, C. and Lingas, A.: Bounds on the length of convex partitions of polygons, *Foundations of Software Technology and Theoretical Computer Science* (Joseph, M. and Shyamasundar, R., eds.), Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 279–295 (1984).

[7] Lothaire, M.: *Combinatorics on Words*, Cambridge Mathematical Library, Cambridge University Press (1997).

[8] Pape-Lange, J.: Cadences in Grammar-Compressed Strings, *CoRR*, Vol. abs/2008.05594 (online), available from ⟨https://arxiv.org/abs/2008.05594⟩ (2020).

[9] Schönhage, A. and Strassen, V.: Schnelle Multiplikation großer Zahlen, *Computing*, Vol. 7, No. 3, pp. 281–292 (1971).

[10] Thompson, W. B., Shirley, P. and Ferwerda, J. A.: A Spatial Post-Processing Algorithm for Images of Night Scenes, *Journal of Graphics Tools*, Vol. 7, No. 1, pp. 1–12 (2002).

[11] Waerden, B. L. v. d.: Beweis einer Baudet'schen Vermutung, *Nieuw Archief voor Wiskunde*, Vol. 15, pp. 212–216 (1927).