Annales Univ. Sci. Budapest., Sect. Comp. 46 (2017) 81–96

SOME IMPROVEMENTS ON NUMBER EXPANSION COMPUTATIONS

Péter Hudoba and Attila Kovács

(Budapest, Hungary)

Communicated by Imre Kátai
(Received July 2, 2017; accepted September 18, 2017)

Im Memoriam Professor Antal Iványi

He was that rare mathematician who could effectively communicate at all levels, imparting his love for the mathematical algorithms with the same ease to undergraduates, graduates and all his colleagues at the University of ELTE, Faculty of Informatics. Tóni was my friend, my department colleague, my tennis partner, whom we could always consult for advice. We keep him in our souls. – Attila Kovács

Abstract. In this paper we present some algorithmic problems and their analysis regarding number expansions in lattices. We show how to compute more efficiently the discrete dynamics of the expansions. We implemented our solutions in the computer algebra system Sage and we measured and analysed the improvements.

1. Introduction

Let Λ be a lattice in $\mathbb{R}^n$ and let $M : \Lambda \to \Lambda$ be a linear operator such that $\det(M) \neq 0$. Let furthermore $0 \in D \subseteq \Lambda$ be a finite subset.

Lattices can be seen as finitely generated free abelian groups. They have many significant applications in pure mathematics (Lie algebras, number theory and group theory), in applied mathematics (coding theory, cryptography) because of conjectured computational hardness of several lattice problems, and are used in various ways in the physical sciences.

In this paper we consider number expansions in lattices.

Key words and phrases: Number expansion, expansive operator.
2010 Mathematics Subject Classification: 11Y55.
The project has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00001)

Definition 1.1. The triple (Λ, M, D) is called a *number system* (GNS) if every element x of Λ has a unique, finite representation in the form

$$x = \sum_{i=0}^{L} M^i d_i \ ,$$

where $d_i \in D$, $L \in \mathbb{N}$, and $d_L \neq 0$.

Here M is called the *base* (or *radix*) and D is the *digit set*. The *length of expansion* of x in Definition 1.1 is $L + 1$.

It is easy to see that similarity preserves the number system property, i.e., if M_1 and M_2 are similar via the matrix Q then the systems (Λ, M_1, D) and $(Q\Lambda, M_2, QD)$ are number systems at the same time. If we change the basis in Λ a similar integer matrix can be obtained, hence, no loss of generality in assuming that M is integral acting on the lattice $\mathbb{Z}^n$.

If two elements of Λ are in the same coset of the factor group $\Lambda/M\Lambda$ then they are said to be *congruent* modulo M. The following theorem shows some necessary conditions for the number system property.

Theorem 1.1. ([6, 3]) *If (Λ, M, D) is a number system then*
(1) *D must be a full residue system modulo M,*
(2) *M must be expansive,*
(3) $\det(I_n - M) \neq \pm 1$ *(unit condition).*

If a system fulfills the first two conditions then it is called a radix system. *It is known that these conditions are not always sufficient* (see [1]).

Let $\varphi : \Lambda \to \Lambda$, $x \overset{\varphi}{\mapsto} M^{-1}(x - d)$ for the unique $d \in D$ satisfying $x \equiv d \pmod{M}$. Since M^{-1} is contractive and D is finite, there exists a norm $\|.\|$ on Λ and a constant C such that the orbit of every $x \in \Lambda$ eventually enters the finite set $S = \{x \in \Lambda \mid \|x\| < C\}$ for the repeated application of φ. This means that the sequence $x, \varphi(x), \varphi^2(x), \ldots$ is eventually periodic for all $x \in \Lambda$. Clearly, (Λ, M, D) is a number system iff for every $x \in \Lambda$ the orbit of x eventually reaches 0.

A point p is called *periodic* if $\varphi^k(p) = p$ for some $k > 0$. The orbit of a periodic point p is a *cycle*. The set of all periodic points is denoted by $\mathcal{P}$. A *signature* $(s_1, s_2, \ldots, s_\omega)$ of a radix system (Λ, M, D) is a finite sequence of nonnegative integers in which the periodic structure $\mathcal{P}$ consists of $\#s_i$ cycles with period length i $(1 \leq i \leq \omega)$.

The following problem classes are in the mainstream of the research: for a given (Λ, M, D)

(1) the *decision problem* asks if the triple form a number system or not;
(2) the *classification problem* means finding all cycles (*witnesses*);

(3) the *parametrization problem* means finding parametrized families of number systems;

(4) the *construction problem* aims at constructing a digit set D to M for which (Λ, M, D) is a number system. In general, it aims at constructing a digit set D to M such that (Λ, M, D) satisfies a given signature.

At the time of writing this paper the algorithmic complexities of the decision and classification problems are unknown. Computer experiments show that the worst cases are exponential. In this paper we suggest new methods improving the running time of the computations. The improved methods still do not have polynomial runtime. The measurements are performed using the computer algebra system Sage. The experiments based on the systems, which are operators created by the companion of monic, integer polynomials with constant terms ± 2,± 3,± 5, or ± 7.

2. Preliminaries

There are various deterministic methods for solving the decision and classification problems. In [4, 2] the authors proposed the method `Decide` (see Algorithm 1 below) that solves the decision problem.

Algorithm 1 `Decide`(Λ, M, D)

```
1:  finished := {}
2:  Compute K(M, D)
3:  for z ∈ K(M, D) ∩ Λ do
4:    if z ∉ finished then
5:      orbit := {}
6:      repeat
7:        orbit := orbit ∪ {z}
8:        finished := finished ∪ {z}
9:        z := φ(z)
10:     until z ∉ finished
11:     if z ≠ 0 and z ∈ orbit then
12:       return FALSE
13:     end if
14:   end if
15: end for
16: return TRUE
```

The algorithm computes the orbit of each integer point in a bounded set $K(M, D) \subset \mathbb{R}^n$ which contain all the periodic points. If an orbit ends up in

a nonzero cycle, a witness is found, therefore it cannot be a number system. Otherwise the system is a number system.

Example 2.1. Let $M = \begin{pmatrix} 1 & -2 \\ 1 & 1 \end{pmatrix}$, $D = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix} \right\}$. The system $(\mathbb{Z}^2, M, D)$ is a number system, Figure 1 shows the orbits of the integers in $K(M, D) = [-1, 1]^2$.

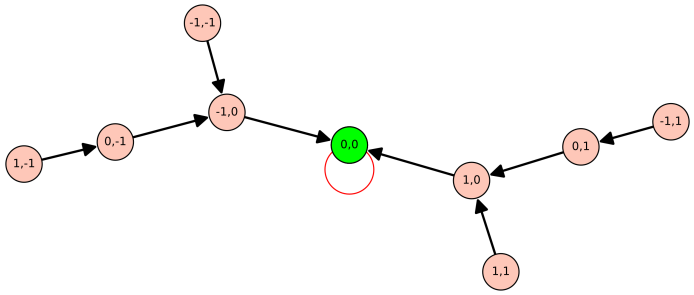

Figure 1: The integer points in $K(M, D)$ and their orbits.

2.1. Fraction set

In order to compute the set $K(M, D)$ various methods are available. Instead of constructing an appropriate norm and determining the set S a computationally simpler model was suggested in [4]. Let us consider the set

$$H = \left\{ \sum_{i=1}^{\infty} M^{-i} d_i \ : \ d_i \in D \right\} \subset \mathbb{R}^n ,$$

the *fundamental set*, or *set of fractions*. It is known that $\mathcal{P} \subset -H$, hence it is enough to calculate a cover of $-H$. Since H is compact in $\mathbb{R}^n$, its cover can easily be calculated. There are various cover set computing procedures known, see [2, 5]. Clearly, the cover set $K(M, D)$ can be any set of $\mathbb{R}^n$ which contain the periodic elements and the integers inside are easily enumerable.

Example 2.2. Let $M = \begin{pmatrix} 2 & -1 \\ 1 & 2 \end{pmatrix}$, $D = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 3 \\ 0 \end{bmatrix}, \begin{bmatrix} 4 \\ 0 \end{bmatrix} \right\}$. Figure 2 reveals the set $-H$, Figure 3 the orbits of the integers in $K(M, D) = [-3, 1] \times \times [-1, 3]$ showing that there are three different loops in the system.

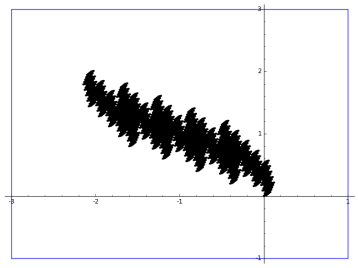

Figure 2: The set $-H$ in $(\mathbb{Z}^2, M, D)$.

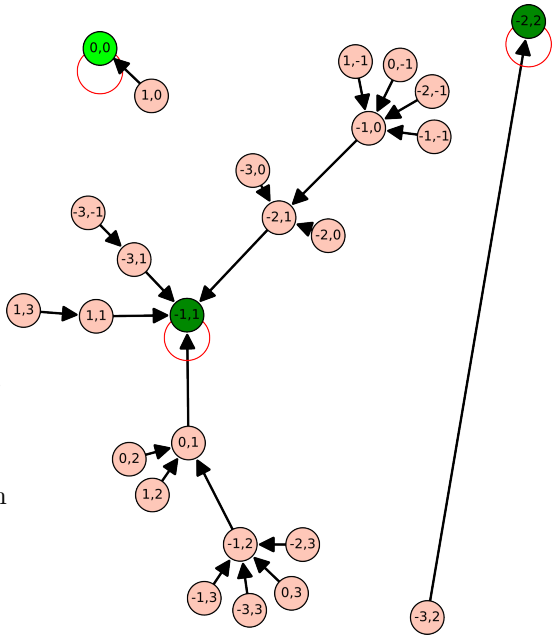

Figure 3: The integer points in $K(M, D)$ and their orbits.

2.2. Volume optimization

In Algorithm 1 the pairs $(x, \varphi(x))$ are computed for all $x \in K(M, D) \cap \Lambda$, hence the running time T of the decision problem can roughly be estimated by

$$T(decide) \sim VolumeOfK \cdot T(computingPhi) \ . \tag{2.1}$$

We can reduce the computation time if we are able to reduce the volume of the set $K(M, D)$ by finding a better basis, or we can improve the efficiency of computing the function φ.

In order to get a better basis in [2] the authors focused on the *VolumeOfK* part of (2.1), i.e, they aimed at finding an unimodular integer matrix Q by which the algorithm `Decide`$(\mathbb{Z}^n, Q^{-1}MQ, QD)$ has smaller running time. They suggested to perform the following steps:

(1) Start with a diagonal matrix Q with ± 1 entries;
(2) Choose a position (i, j) randomly from the upper triangle of Q;
(3) Modify the matrix at position (i, j) and compute the new volume;
(4) If improvements were found then Goto (3) otherwise Goto (2).

The process can be performed until a time limit is reached or there are no improvements in a limited number of steps. For this task in [2] AI algorithms were suggested, like simulated annealing or genetic algorithm. The following algorithm realizes the steps (3)-(4), i.e., it searches a basis with smaller covering by modifying the position (i, j) of Q randomly using a direction value ± 1.

Algorithm 2 `VolumeOptSearchAtPosition`$(M, D, Q, i, j, direction)$

$vol \leftarrow VolumeOfK(Q^{-1}MQ, QD)$
$Q_1 \leftarrow Q$
$improved \leftarrow$ TRUE
while *improved* **do**
  $oldVol \leftarrow vol$
  $Q_1[i, j] \leftarrow Q_1[i, j] + direction$
  $vol \leftarrow VolumeOfK(Q_1^{-1}MQ_1, Q_1D)$
  $improved \leftarrow vol < oldVol$
end while
$Q_1[i, j] \leftarrow Q_1[i, j] - direction$
return $(Q_1, oldVol)$

After executing the algorithm `VolumeOptSearchAtPosition` repeatedly at different positions as we described above, we can find a transformation matrix Q (denoted by arrows on Figure 4) to achieve a smaller volume for the covering of $K(Q^{-1}MQ, QD)$.

Example 2. (contd.) Figure 4 and 5 visualize the fraction sets in different bases. The second case can be obtained by using the transformation matrix $Q = \left(\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right)$. The cover in the optimized base is smaller, and still has every periodic point in it.

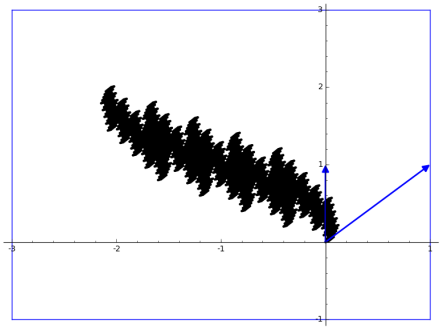

Figure 4: $|K(M, D) \cap \mathbb{Z}^2| = 9$.

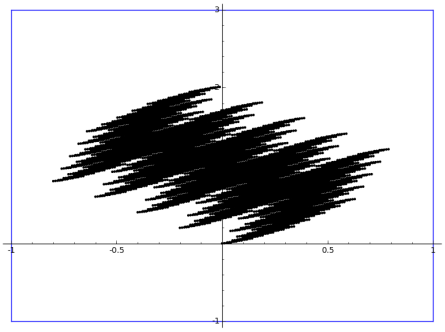

Figure 5: $|K(M, D) \cap \mathbb{Z}^2| = 3$.

In Garsia systems (companions of monic integer polynomials with constant term ± 2) up to dimension 7 the *VolumeOfK* optimization resulted in 75% smaller cover on average. For more details see [2].

3. Our contribution

The `Decide` algorithm (Algorithm 1) is deterministic, but the running time may grow exponentially with the dimension. To decrease the computation time let us investigate the algorithm in detail. It can be divided into three steps:

(1) Determining an appropriate set $K(M, D)$ (optimized in [2]),
(2) Calculating the graph G with edges $(x, \varphi(x))$ for all $x \in K(M, D) \cap \Lambda$,
(3) Finding the cycles in G.

In this paper, we are focusing on minimizing the runtime of Step (2). For searching the cycles (Step 3) there are many known algorithms, in this paper we do not discuss that part.

3.1. Estimating the φ computation

Next we concentrate on the $T\,(computingPhi)$ part of (2.1). Recall that $\varphi : \Lambda \to \Lambda$, $x \overset{\varphi}{\mapsto} M^{-1}(x - d)$ for the unique $d \in D$ satisfying $x \equiv d \pmod{M}$. The computation of φ is performed in the following way:

(1) Find the congruent $d \in D$ digit;
(2) Subtract two vectors;
(3) Multiply by M^{-1} (shift).

Clearly, it is enough to consider (1) and (3). If we store the inverse matrix in sparse representation then we do not need to calculate multiplications for zero elements in the inverse matrix, hence we get the estimation

$$T(computingPhi) \sim T_m(inverseWeight) + T(findCongrElement)\,, \tag{3.1}$$

where *inverseWeight* of M denotes the number of nonzeros in M^{-1}, and $T_m(x)$ means the time of multiplications of x objects. In order to find the congruent element normal forms can be used (see[4]). Let $UMV = G$ be the Smith normal form of M where U and V are unimodular. Let furthermore the diagonal of G be $g_1, ..., g_n$.

Theorem 3.1 (Congruence with Smith normal form [4]). *Let $z_1, z_2 \in \mathbb{Z}^n$. For a radix M let the numbers $u_1, u_2, \ldots, u_n$ and $\hat{u_1}, \hat{u_2}, \ldots, \hat{u_n}$ denote the coordinates of Uz_1 and Uz_2, respectively. Then $z_1 \equiv z_2$ modulo M if and only if $u_i \equiv \hat{u_i}$ modulo g_i for all $i = 1, 2, \ldots, n$.*

Clearly, we need to compute the Smith normal form only once before the `Decide` algorithm, so the computational overhead is negligible. To determine the congruent element for a given $z \in \mathbb{Z}^n$ we must find $Uz \pmod G$ in the set $\{Ud \pmod G, d \in D\}$ which can be performed by hashing (as was suggested in [4]). Having the sparse representation of U one needs to consider only its nonzero elements. Moreover, we have to perform the multiplications only in those rows where $|g_i| > 1$. Suppose that we have $s \leq n$ such rows. Let us denote by *smithWeight* of M the number of nonzero values in those the rows of U where the corresponding g_i absolute values are bigger than one. Then we get that

$$T(findCongrElement) \sim T_m(smithWeight + s)\,,$$

so our final estimation for the time of computing φ is

$$T(computingPhi) \sim T_m(inverseWeight + smithWeight + s)\,. \tag{3.2}$$

In 3.2 the value s counts the number of divisions, but we consider them equivalent to multiplications. We start our investigations based on this estimation and present some improvements of the `Decide` algorithm. In the rest of this paper we always assume that all the matrices are stored in sparse form in the computing environment.

Our aim is to show how to find an unimodular basis transformation Q in which

(1) the *volume* of $K(Q^{-1}MQ, QD)$,
(2) the *inverseWeight* of $Q^{-1}MQ$, and
(3) the *smithWeight* of $Q^{-1}MQ$.

are as small as possible.

3.2. Generic optimization

The following generic optimization algorithm can be used with a custom *targetFunction*. The method is a slight modification of Algorithm 2, making it more generic.

Algorithm 3 `GenericOptSearchAtPosition`$(M, D, Q, i, j, direction,$ $targetFunction)$

1: $value \leftarrow targetFunction(M, D, Q)$
2: $Q_1 \leftarrow Q$
3: $improved \leftarrow$ TRUE
4: **while** *improved* **do**
5: $\quad oldValue \leftarrow value$
6: $\quad Q_1[i, j] \leftarrow Q_1[i, j] + direction$
7: $\quad value \leftarrow targetFunction(M, D, Q)$
8: $\quad improved \leftarrow value < oldValue$
9: **end while**
10: $Q_1[i, j] \leftarrow Q_1[i, j] - direction$
11: **return** $(Q_1, oldVol)$

Observe that using $VolumeOfK(Q^{-1}MQ, QD)$ as the *targetFunction* we got Algorithm 2. Algorithm 3 can be used to achieve the optimal basis searching in the following straightforward way:

Algorithm 4 `SimpleGenericOptSearch`$(M, D, Q, targetFunction,$ $iterateNum, time)$

1: $Q \leftarrow I$
2: **for** i **to** $iterateNum$ **do**
3: $\quad i \leftarrow$ choose randomly an integer from $[1, dim\,(M)]$
4: $\quad j \leftarrow$ choose randomly an integer from $[1, dim\,(M)]$
5: $\quad direction \leftarrow$ choose randomly an integer from $\{-1, 1\}$
6: $\quad (Q, value) \leftarrow GenericOptSearchAtPosition(M, D, Q, i, j, direction,$ $targetFunction)$
7: **end for**
8: **return** Q

In practice we applied simulated annealing with 2 cooldown. Algorithm 5 describes the generic optimization.

Algorithm 5 `GenericOptSearch`$(M, D, targetFunction, candNum, iterateNum, time)$

```
1: Q_best ← I
2: improvementFound ← TRUE
3: while improvementFound do
4:   Candidates ← [Q_best]
5:   while running time ≤ time do
6:     for c to candNum do
7:       for i to iterateNum do
8:         i ← choose randomly an integer from [1, dim(M)]
9:         j ← choose randomly an integer from [1, dim(M)]
10:        direction ← choose randomly an integer from {−1, 1}
11:        (Q_1, value) ←
12:          GenericOptSearchAtPosition(M, D, Candidates[c], i, j,
                                        direction, targetFunction)
13:        Append Q_1 to Candidates
14:      end for
15:    end for
16:    Candidates ← choose the best candNum number of candidates from
       Candidates
17:  end while
18:  Q_best ← pick the best candidate from Candidates
19:  if no improvements in Q_best then
20:    improvementFound ← FALSE
21:  end if
22: end while
23: return Q_best
```

3.3. One-step optimization

In the following we define a new target function for the optimization. Clearly, the optimization process is unnecessary if the volume of $K(M, D)$ is small, i.e., it has less than a few thousand integer points inside which are easily enumerable. Otherwise for a given transformation Q we estimate $T(Decide) \sim volumeOfK \cdot T(computingPhi)$ based on (3.2). Then, our *targetFunction* changes in the following way:

Algorithm 6 `OneStepTargetFunction`(M, D, Q)

```
1: vol ← VolumeOfK(Q^{-1}MQ, QD)
2: inverseWeight ← number of nonzeros in Q^{-1}M^{-1}Q
3: smithWeight ← number of nonzeros in smith U of Q^{-1}MQ
4: return vol · (inverseWeight + smithWeight)
```

The constant s from (3.2) does not appear in the algorithm, because the integer basis transformation with Q does not change the value of s. With this method we try to find a better basis starting from a given position by optimizing the values of the *volume*, the *inverseWeight* and *smithWeight* at the same time.

3.4. Two-step optimization

In this approach the volume of $K(M, D)$ and the factors *inverseWeight* and *smithWeight* are minimized *separately* by transformations Q_{vol} and Q_{phi}, respectively. It means that in order to decide the GNS property we have to

1. iterate through the integer points $z \in K(Q_{vol}^{-1} M Q_{vol}, Q_{vol} D)$ in a way that
2. transform each point z to the system $(\mathbb{Z}^n, Q_{phi}^{-1} M Q_{phi}, Q_{phi} D)$ with $Q_{phi} Q_{vol}^{-1}$ and
3. calculate $\varphi(z)$ in this system.

After determining the transformation Q_{vol} by `VolumeOptSearchAtPosition` we traverse the integer points in the Q_{vol} transformed covering, transforming again each point by $Q_{phi} Q_{vol}^{-1}$ to the φ computation optimized system and solve the decision problem there.

With this approach, compared to the original optimized algorithm, the number of necessary φ computations are the same but their execution is faster. The efficiency of the optimization for Q_{phi} can be measured by the sum

$$inverseWeight + smithWeight + transformationWeight,$$

where *transformationWeight* is the number of nonzeros in the matrix $Q_{phi} Q_{vol}^{-1}$.

Algorithm 7 `PhiOptimizedTargetFunction`(M, D, Q)

1: $inverseWeight \leftarrow$ number of nonzeros in $Q^{-1} M^{-1} Q$
2: $smithWeight \leftarrow$ number of nonzeros in smith U of $Q^{-1} M Q$
3: $tansfWeight \leftarrow$ number of nonzeros in $Q Q_{vol}^{-1}$ (Q_{vol} is the volume optimized transformation)
4: **return** $inverseWeight + smithWeight + transfWeight$

Algorithm 8 `DecideWithTransform`(Λ, M, D, T)

```
finished := {}
K ← K(M, D)
for z_1 ∈ K ∩ Λ do
    z ← T z_1
    if z ∉ finished then
        orbit := {}
        repeat
            orbit := orbit ∪ {z}
            finished := finished ∪ {z}
            z := φ(z)
        until z ∉ finished
        if z ≠ 0 and z ∈ orbit then
            return FALSE
        end if
    end if
end for
return TRUE
```

The two-step approach calls the $DecideWithTransform(\Lambda, Q_{vol}^{-1} M Q_{vol}, Q_{vol} D, Q_{phi} Q_{vol}^{-1})$ function, where Q_{vol} and Q_{phi} are the results of the volume and the φ optimization process.

3.5. Experimental results

We implemented the algorithms in Sage and ran it for roughly two weeks on two Amazon EC2 c4.xlarge and three Amazon EC2 c4.large servers. In order to estimate the runtime we calculated 10 000 steps of φ for each randomly chosen radix systems. As we mentioned earlier, we used companion matrices as operators generating from monic integer polynomials with constant terms $\pm 2, \pm 3, \pm 5$, or ± 7 for the measurements.

First of all we measured the runtime of the volume optimized decision algorithm (denote below with "original") and our two new approaches. As you can see in the Figure 6, the two-step (two-transform) solution was more efficient in most cases.

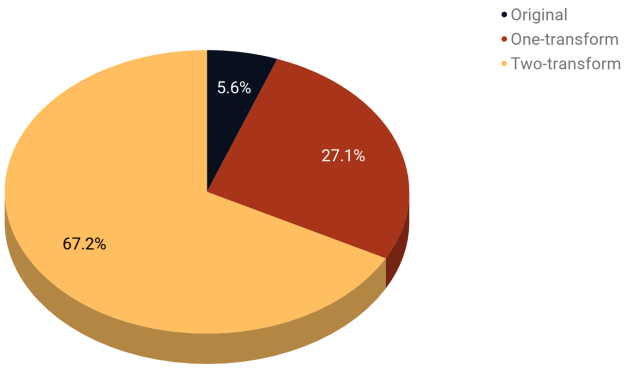

Figure 6: Ratio of efficiency between various approaches for GNS Garsia operators.

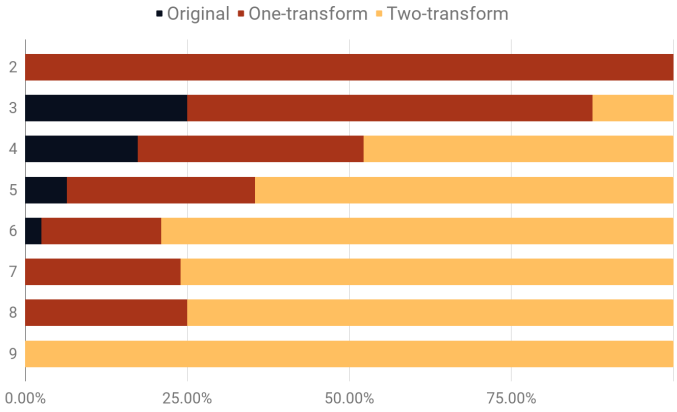

Figure 7: Comparing the efficiency of the approaches for arbitrary Garsia operators.

Moreover, our experiments show that in the GNS cases with higher dimensions the two-transform approach is always superior (Figure 8).

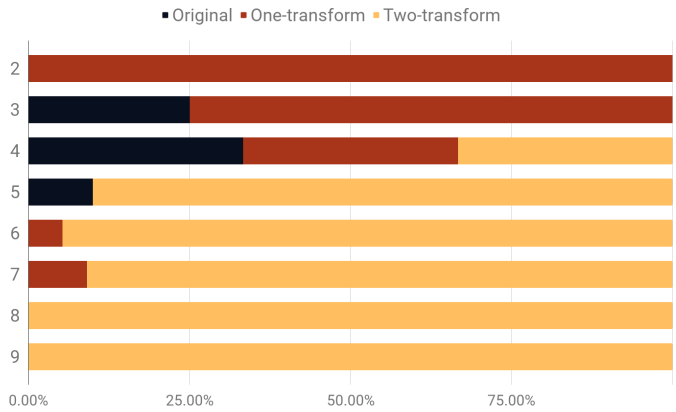

Figure 8: Comparing the efficiency of the approaches for GNS Garsia operators.

Finally, Figure 9 shows the overall runtime improvements in different dimensions (approximated with splines).

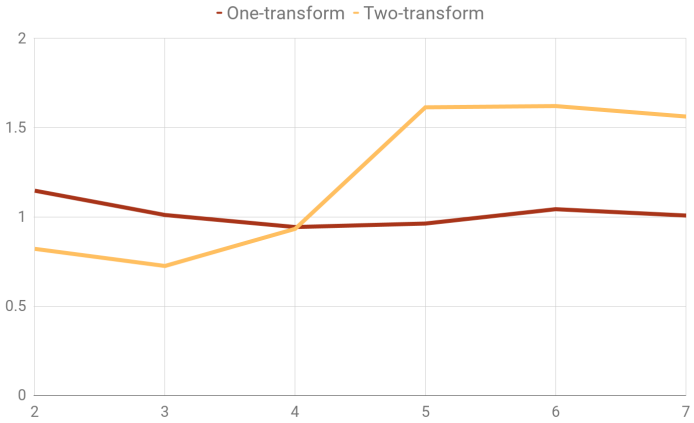

Figure 9: Runtime improvements in different dimensions for Garsia systems – the speedup is compared to the simple optimized original version.

3.6. Further analysis

Our approaches resulted in reasonable improvements in Garsia cases, but somewhat less in systems with higher constant terms. Therefore we combined our algorithms and measured the speedup. Table 1 shows the result.

1. Two-step, version 2
   (a) Run the volume optimization on (Λ, M, D) to get Q_{vol}
   (b) Run the φ optimization on $(\Lambda, Q_{vol}^{-1} M Q_{vol}, Q_{vol} D)$ to get Q_{phi}
   (c) Run `DecideWithTransform` with $T = Q_{phi} Q_{vol}$
2. Combined
   (a) Run the One-step optimization to get Q_{os}
   (b) Run the φ optimization on (Λ, M, D) to get Q_{phi}
   (c) Run `DecideWithTransform` with $T = Q_{phi} Q_{os}^{-1}$
3. Combined, version 2
   (a) Run the One-step optimization to get Q_{os}
   (b) Run the φ optimization on $(\Lambda, Q_{os}^{-1} M Q_{os}, Q_{os} D)$ to get Q_{phi}
   (c) Run `DecideWithTransform` with $T = Q_{phi} Q_{os}$

	One-step	Two-step	Two-step 2
Garsia	92.14%	204.55%	209.41%
With ±3 constant term	99.66%	117.82%	257.58%
With ±5 constant term	84.09%	111.40%	242.11%
With ±7 constant term	96.40%	151.34%	293.81%
Overall	95.23%	125.12%	255.63%

	Combined	Combined 2
Garsia	71.48%	178.45%
With ±3 constant term	115.09%	253.64%
With ±5 constant term	92.92%	163.19%
With ±7 constant term	145.00%	265.66%
Overall	109.20%	225.04%

Table 1: Average speedup compared to the original volume optimized approach.

4. Summary

The proposed new algorithms in Garsia systems show better performance with increasing dimensions, and the two-transform approach looks generally

better than the others. Random experiments in dimension 14 resulted in more than 145% speedup. Experimental results show also that the algorithms may get stuck in a local optimum, but if we combine the one-step and two-step approaches and execute them one after another, a globally better result can be achieved.

References

[1] **Barbé, A. and F. von Haeseler,** Binary number systems for $\mathbb{Z}^k$, *J. Number Theory*, **117(1)**, (2006), 14–30.

[2] **Burcsi, P., A. Kovács and Zs. Papp-Varga,** Decision and classification algorithms for generalized number systems, *Annales Univ. Sci. Budapest., Sect. Comp.*, **28**, (2008), 141–156.

[3] **Kovács, A.,** *Radix Expansion in Lattices*, PhD dissertation, Eötvös Loránd University, Budapest, 2001.

[4] **Kovács, A.,** On computation of attractors for invertible expanding linear operators in $\mathbb{Z}^k$, *Publ. Math. Debrecen*, **56(1-2)**, (2000), 97–120.

[5] **Kovács, A.,** On number expansions in lattices, *Math. and Comp. Modelling*, **8**, (2003), 909–915.

[6] **Vince, A.,** Radix representation and rep-tiling, *Congressus Numerantium*, **98**, (1993), 199–212.

P. Hudoba and A. Kovács
Department of Computer Algebra
Eötvös Loránd University
H-1117 Budapest
Pázmány Péter Sétány 1/C
Hungary
hudoba.peter@inf.elte.hu
attila.kovacs@inf.elte.hu