Lattice-Based Memory Allocation

Alain Darte
CNRS, LIP, ENS-Lyon,
46, Allée d'Italie,
Lyon, France
Alain.Darte@ens-lyon.fr

Rob Schreiber
Hewlett Packard Laboratories,
1501 Page Mill Road,
Palo Alto, USA
Rob.Schreiber@hp.com

Gilles Villard
CNRS, LIP, ENS-Lyon,
46, Allée d'Italie,
Lyon, France
Gilles.Villard@ens-lyon.fr

ABSTRACT

We investigate the problem of memory reuse, for reducing the necessary memory size, in the context of compilation of dedicated processors. Memory reuse is a well-known concept when allocating registers (i.e., scalar variables). Its (recent) extension to arrays was studied mainly by Lefebvre and Feautrier (for loop parallelization) and by Quilleré and Rajopadhye (for circuit synthesis based on recurrence equations). Both consider affine mappings of indices to data, with modulo expressions in the first and (mainly) projections in the second. We develop a mathematical framework based on (integral) critical lattices that subsumes all previous approaches and gives new insights into the problem. Our technique consists first in building an abstract representation of conflicting indices (equivalent in a multi-dimensional space to the interference graph for register allocation), then in defining an integral lattice, admissible for the set of differences of conflicting indices, used to build a valid modular allocation. We also show the link with critical lattices, successive minima, and basis reduction, and we analyze various strategies for lattice-based memory allocation.

Categories and Subject Descriptors

B.6.1 [**Logic Design**]: Design Styles—*logic arrays, memory control and access*; B.6.3 [**Logic Design**]: Design Aids—*automatic synthesis, optimization*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures, geometrical problems and computations*; G.2.1 [**Discrete Mathematics**]: Combinatorics—*combinatorial algorithms.*

General Terms

Algorithms, Design, Theory.

Keywords

Program transformation, memory allocation, admissible lattice, critical determinant, successive minima.

1. INTRODUCTION

In the context of program optimizations, and especially loop transformations, scheduling and mapping computations are two major problems that compilers-parallelizers for parallel machines or embedded systems, and compilers for automatic generation of hardware accelerators have to face. In this last context, a particularly important problem is the design of the buffers needed to store data inside one application-specific circuit as well as between communicating hardware processes. All projects that try to automatically design hardware accelerators from high-level programs have to face it (see for example the PICO project [12] that was developed at HP Labs, the Atomium project [6] at IMEC, the Compaan project [13] developed in Leiden, the Alpha project [20] first developed in Rennes, etc.). As noticed in [22], a theoretical framework is still lacking to describe and analyze the interplay between scheduling constraints (i.e., *when* statements can be executed) and storage constraints (i.e., *where* their results can be stored). The theory is quite mature now for the scheduling problem itself (see for example [4] for a survey), when communication costs or storage optimizations are ignored, but this is not the case for the storage allocation problem, even for a fixed schedule of loop iterations. Several heuristics have been proposed to reduce the memory size required for a given program, but there is no mathematical framework available in which to compare heuristics and analyze them with respect to some optimal theoretical solution.

De Greef, Catthoor, and De Man [8] proposed a memory reduction technique based on canonical linearizations of an array, followed by a wrapping with a modulo operation. For example, for a 2-dimensional square array of size N, indexed with i and j, they consider four mappings $Ni + j \bmod b$, $-Ni + j \bmod b$, $Ni - j \bmod b$, and $-Ni - j \bmod b$, where b is a large enough integer, to be defined. Lefebvre and Feautrier [15] proposed a more general technique based on a multi-projection (with a folding thanks to a modulo operation in each dimension) of the original loop indices (not the array indices). For a 2-dimensional loop, the result of iteration (i, j) is mapped in memory location $(i \bmod b_i, j \bmod b_j)$ where b_i and b_j are well-chosen integers. Quilleré and Rajopadhye [19] proposed to project loop indices, not in the original basis, but along some well-chosen vectors, depending on the schedule (they also mentioned the use of modulos, but only with a brief analysis). They showed that, under some hypotheses (mainly that iteration domains have large sizes in all dimensions), this strategy leads to a memory size that is optimal in order of magnitude. More recently, Thies, Vivien, Sheldon, and Amarasinghe [22], extending the work of Strout, Carter, Ferrante, and Simon [21], considered the problem of reusing storage along only one vector (simple projection + one modulo operation). Although part of their technique is more limited than previous work, they also consider memory reductions valid for *any* schedule. They conclude by mentioning the need for a technique able "to consider more general storage mappings" and that "would allow variations in the number of array dimensions, while still capturing the directional and modular reuse of the occupancy vector". This is exactly our goal here; to propose and study such a mathematical framework, which allows us to capture all linear modular storage allocations, to define optimality, and to discuss the quality of heuristics with respect to this optimal.

In Section 2, we introduce modular allocations and the basic object that we need to build from the program representation, the set DS of differences of conflicting indices (a set symmetric with respect to 0). We show the equivalence between valid modular allocations and strictly admissible integral lattices for DS. In Section 3, we introduce approximation mechanisms, using techniques from the theory of successive minima and reduction theory. This study shows how the optimal memory size achievable by a modular allocation is linked to the volume of DS; for a n-dimensional polytope DS (this is the most common case in practice), any modular allocation corresponds to a memory size larger than $\text{Vol}(DS)/2^n$ and we develop heuristics that lead to a memory size $c_n \text{Vol}(DS)$ where c_n is a constant that depends on the dimension n only, thus optimal up to a multiplicative factor. As a particular case of our heuristics, we retrieve the mechanism used by Lefebvre and Feautrier, which, as we show, should be – in general – applied in a well-chosen basis. In Section 4, we present a case study of a modestly sized application, which illustrates the concepts and techniques we discussed. We conclude in Section 5.

2. MODULAR ALLOCATIONS AND THE SET OF CONFLICTING INDICES

In this paper, we are not going to focus too much on the first steps of the compilation of loops, i.e., how to analyze the program, how to represent it, how loops can be scheduled while respecting dependences. etc. On the contrary, we will keep the discussion as general as possible so as to develop a framework suitable for memory allocation whatever the program analysis and program representation.

We use the classical representation of loops (possibly not perfectly nested) with iteration vectors. A n-dimensional iteration vector $\vec{i} = (i_1, \ldots, i_n)$ represents the values of the counters of n nested loops. Each statement S has an associated iteration domain $\mathcal{D}_S$, a closed body in $\mathbb{R}^n$, such that $\forall \vec{i} \in \mathcal{D}_S \cap \mathbb{Z}^n$, there is a dynamic instance (or operation) $u = (S, \vec{i})$ of statement S at iteration $\vec{i}$ during the execution of the program. We assume that a scheduling function θ (that may or may not express some parallelism) is given, which assigns to each operation u a "virtual" execution time, i.e., an element of a partially ordered set $(\mathcal{T}, \preceq)$; the operation u will be computed strictly before the operation v iff $\theta(u) \prec \theta(v)$. A schedule respects dependences, especially flow dependences, i.e., if v uses a value computed by u, then $\theta(u) \prec \theta(v)$. Classically, $(\mathcal{T}, \preceq)$ is the set of integers with the order $\leq$, or a discrete d-dimensional space with the lexicographic order, and the mapping θ, when restricted to operations $u = (S, \vec{i})$, is an affine function of $\vec{i}$.

2.1 Modular Allocations

We seek a storage allocation σ that specifies, for each operation $u = (S, \vec{i})$, where its result will be stored; $\sigma(u)$ gives an array name and an access function to it. Furthermore, we would like, if possible, to reuse memory when a value that has been stored is no longer used. To simplify the discussion, we first make two assumptions: a) the n loops surrounding S have a unitary step and each operation $(S, \vec{i})$ creates a value that needs to be stored for a future use; b) all values created by an operation of a given statement S are stored in a unique dedicated array A_S. In other words, there is no memory sharing between different statements, but only between different operations of the same statement (and all share the same array). With these hypotheses, the set of operations that share the array A_S can be represented as the set of all integer points within the body $\mathcal{D}_S$. [1]

To simplify the search space as well as the resulting code generation, we restrict our study to **modular allocations**: in a modular allocation, the value computed by $u = (S, \vec{i})$ is stored in the array A_S in the location with indices $\sigma(S, \vec{i}) = \sigma_S(\vec{i}) = (M_S \, \vec{i}) \bmod \vec{b}_S$, where M_S is an $n \times n$ integral matrix and $\vec{b}_S$ is n-dimensional integral vector (the modulo operation is applied component-wise). The array A_S will then be implemented as an array with multi-dimensional extent $\vec{b}_S$, and its size (that we want to minimize) is the product of the components of $\vec{b}_S$. All previous techniques use restricted forms of the modular allocations we just defined. For example, a projection corresponds to the case where one or more components of $\vec{b}_S$ are equal to 1 (in which case, we can forget about the corresponding row(s) of M_S). We could also look for matrices with more rows than columns, but as we will see, this will not help, as far as memory size is concerned. Note also that, here, we represent the allocation with respect to the loop iteration vectors. We could also make the same reasoning with respect to the original array indices being accessed. Many variants are possible, but one should not forget that the solutions we can capture are only those that are affine with respect to the representation (i.e., indices) we are working with.

2.2 Conflicting Indices

We need to be able to determine valid modular allocations, i.e., allocations for which the semantics of the program is preserved. For each value created by an operation $u = (S, \vec{i})$, with $\vec{i} \in \mathcal{D}_S \cap \mathbb{Z}^n$, we denote by $\text{Last}(u)$ the last operation $v = (T, \vec{j})$ that uses it (remember that we assumed that all iteration vectors in $\mathcal{D}_S$ corresponds to operations that create a useful value). Consider two iteration vectors $\vec{i} \neq \vec{i'}$ and assume without loss of generality that $\theta(S, \vec{i}) \preceq \theta(S, \vec{i'})$.

[1]To generalize our technique to more complex cases and to not loose opportunities, we would need to identify the set $\mathcal{O}(A)$ of all operations that are going to store a useful value in the same shared array A (these could be only part of the operations corresponding to a given statement or these could even be operations corresponding to different statements) and to represent this set, thanks to some one-to-one mapping, as the set of all integer points within a particular body $\mathcal{D}(A)$. Then the mapping for memory allocation will be expressed in terms of the integer points in $\mathcal{D}(A)$.

Then $\theta(S,\vec{i}) \preceq \theta(S,\vec{i'}) \prec \theta(\text{Last}(S,\vec{i'}))$ since θ is a valid schedule. Two cases can occur:

- $\theta(S,\vec{i'}) \preceq \theta(\text{Last}(S,\vec{i}))$: the result of operation $(S,\vec{i'})$ cannot be stored at the same place as the result of operation $(S,\vec{i})$ since it is not "dead" yet [2]. We say that $\vec{i}$ and $\vec{i'}$ are **conflicting indices**. (In the general case, we should speak about conflicting *operations*.)

- $\theta(\text{Last}(S,\vec{i})) \prec \theta(S,\vec{i'})$: the memory location used for operation $(S,\vec{i})$ can be safely reused for operation $(S,\vec{i'})$.

Thanks to this analysis, we define the symmetric and reflexive relation $\bowtie$ of conflicting indices:

$$\vec{i} \bowtie \vec{i'} \Leftrightarrow \begin{cases} \theta(S,\vec{i}) \preceq \theta(\text{Last}(S,\vec{i'})) \\ \theta(S,\vec{i'}) \preceq \theta(\text{Last}(S,\vec{i})) \end{cases}$$

and we define $CS = \{(\vec{i},\vec{i'}) \in (\mathcal{D}_S \cap \mathbb{Z}^n)^2 \mid \vec{i} \bowtie \vec{i'}\}$, the set of all pairs of conflicting indices. Note that because $\bowtie$ is symmetric, CS is symmetric with respect to the plane of pairs $(\vec{i},\vec{i})$ and, since $\bowtie$ is reflexive, $(\vec{i},\vec{i}) \in CS$ when $\vec{i} \in \mathcal{D}_S \cap \mathbb{Z}^n$. Finally, we define DS, the set of all differences of conflicting indices, $DS = \{\vec{d} = (\vec{i} - \vec{i'}) \mid (\vec{i},\vec{i'}) \in CS\}$. Because of the properties of CS, $0 \in DS$ and DS is 0-symmetric (i.e., symmetric with respect to 0). Note that CS is nothing but a multi-dimensional generalization of the interference graph for register allocation. Also, the well-known quantity MAXLIVE (the maximal number of values simultaneously live) is equal to the maximal cardinality of a set S such that $S \times S \subset CS$ (all indices in S are conflicting with each other).

We can now determine when a memory allocation σ is valid. In the general case, when several statements can share the same memory locations, we would have to generalize the relation $\bowtie$ to conflicting operations (instead of conflicting iteration vectors, i.e., operations of a given statement). Then an allocation σ would be valid iff $(S,\vec{i}) \bowtie (T,\vec{j})$ and $(S,\vec{i}) \neq (T,\vec{j})$ imply $\sigma(S,\vec{i}) \neq \sigma(T,\vec{j})$. In our restricted case, a memory allocation σ_S is valid iff $\vec{i} \bowtie \vec{i'}$, $\vec{i} \neq \vec{i'} \Rightarrow \sigma_S(\vec{i}) \neq \sigma_S(\vec{i'})$, i.e., $(\vec{i},\vec{i'}) \in CS, \vec{i} \neq \vec{i'} \Rightarrow M_S\,\vec{i} \bmod \vec{b}_S \neq M_S\,\vec{i'} \bmod \vec{b}_S$, or equivalently, iff $M_S\,\vec{d} \bmod \vec{b}_S = 0, \vec{d} \neq 0 \Rightarrow \vec{d} \notin DS$. In other words, 0 should be the only integer point in DS whose image is 0. We will illustrate the different concepts and the different traps in which a modular memory allocator can fall through the following ad-hoc example (and some variants).

Example 1. Consider the code fragment in Figure 1 and suppose that we can reuse the memory for A, i.e., A is not "live-out" from the code fragment. Suppose that the two loops are scheduled sequentially as they are written but that, for some reason, the second loop is pipelined, with respect to the first one, one "clock cycle" later. In other words, $\theta(S,(i,j)) = Ni + j$ and $\theta(T,(i,j)) = Ni + j + 1$. This kind of situation occurs typically when compiling communicating processes, the values of A being placed in an intermediate buffer whose size is to be optimized. Here, MAXLIVE $= 2$ (we can use alternatively two registers for example). But how can a compiler build such a solution? And can it be achieved by a modular allocation? We will see later how this can be

[2]We could also say that there is a conflict only if $\theta(S,\vec{i'}) \prec \theta(\text{Last}(S,\vec{i}))$ instead of $\theta(S,\vec{i'}) \preceq \theta(\text{Last}(S,\vec{i}))$. This definition depends on when we consider that a value is dead, but this does not change the underlying concepts.

```
DO $i = 0,\ N-1$
  DO $j = 0,\ N-1$
    S: $A(i,j) = \ldots$
  ENDDO
ENDDO

DO $i = 0,\ N-1$
  DO $j = 0,\ N-1$
    T: $B(i,j) = A(i,j) + \ldots$
  ENDDO
ENDDO
```

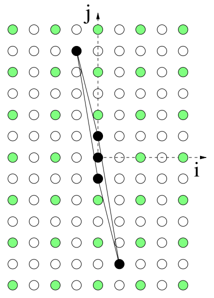

Figure 1: Code for Example 1, the set DS, and a strictly admissible integral lattice.

done. Let us first compute the (exact) relation $\bowtie$, and the sets CS and DS: $(i,j) \bowtie (i,j+1)$ if $0 \leq j < N-1$ and $(i,N-1) \bowtie (i+1,0)$ (two successive iterations cannot share the same memory location). DS is the set of all integer points in the polytope (with volume 2) with vertices $(0,1)$, $(0,-1)$, $(1,1-N)$, and $(-1,N-1)$ depicted in Figure 1 for $N = 6$ (black points). Here, the set DS is the same whether we reason with loop indices or array indices since both are aligned, but this is a particular case. In the rest of the paper, when we will play with this example, we will define DS with respect to loop indices (and not array indices), so that DS does not depend on how the array A is defined by the user (e.g., as a 1D array or a 2D array). For example, here, if DS is defined with respect to loop indices, replacing $A(i,j)$ by $A(Ni+j)$ will not change it.

The mapping $Ni + j \bmod 2$ is valid: $N \times 0 + 1 \bmod 2 = 1$ and $N \times 1 + 1 \times (1-N) \bmod 2 = 1$. When N is even (resp. odd), the mapping is nothing but $j \bmod 2$ (resp. $i + j \bmod 2$). Note that in this particular case, MAXLIVE is achievable with a modular allocation (which is not the case in general). An example of an invalid mapping is $j \bmod 2$ when N is odd since the conflicting iterations $(i,N-1)$ and $(i+1,0)$ would store their results at the same place (the value computed at time $Ni + N - 1$ would be overwritten at time $N(i+1) = Ni+N$ at the same time it is read by operation $(T,(i,N-1))$. An example of a suboptimal but valid mapping is $(i \bmod 2, j \bmod 2)$, which implies a memory of size 4. The set of vectors $\vec{d}$ whose image is 0 for this mapping is also depicted in Figure 1 (grey points), its intersection with DS is indeed reduced to 0. We will come back to this mapping later. $\square$

So far, the definitions of the sets CS and DS are purely formal. We did not explain how to compute them. We will not do so here, since this is not the goal of this paper and we want to keep the discussion as general as possible. We just point out that when θ is defined by affine schedules (even multi-dimensional), these sets can be computed by looking for lexicographic minima or maxima in polytopes, as done in [7] for dependence analysis, in [15, 19] for memory reduction, or [23] for storage management. In this case, CS (resp. DS) can be represented as all integer points within a polytope (or a union of polytopes) in $\mathbb{R}^{2n}$ (resp. $\mathbb{R}^n$). Also, any super-approximation of CS (and DS) can be used to derive valid memory allocations. The situation is the same as for the scheduling problem: super-approximations can be needed for practical reasons, and the quality of the result depends on the accuracy of the super-approximation. A complete study remains to be done to determine the impact of an approximation scheme on the quality of the allocation, but this goes out of the scope of this paper.

To conclude this discussion, and to make the rest of the paper simpler, we will now forget about the way CS and DS are computed or super-approximated, and we will consider the "simplest" case, the case where CS (resp. DS) is the set of *all integer points within a polytope.* [3] This is not the general case, but it is important to understand at least this case, since it is very common in practice, and it is also more or less the case in [15, 19] where, basically, the optimizations rely on the set $\Delta = \{(\vec{i}, \vec{i}') \in (\mathcal{D}_S)^2 \mid \theta(S, \vec{i}) \preceq \theta(S, \vec{i}') \preceq \theta(S, \vec{i}) + \vec{t}_S\}$ where $\vec{t}_S$ is the maximal (w.r.t. $\preceq$) "time" difference $\theta(\text{Last}(S, \vec{i})) - \theta(S, \vec{i})$. Using $\vec{t}_S$ instead of $\vec{t}_{S,\vec{i}} = \theta(\text{Last}(S, \vec{i})) - \theta(S, \vec{i})$ for each particular $\vec{i}$ corresponds to a super-approximation so that Δ is a superset of CS. This is illustrated with the following example.

Example 2. Consider the code of Example 1. Expressed in terms of a multi-dimensional schedule (as in [15, 19]), $Ni + j$ corresponds to the schedule (i, j). (Note however that $Ni + j + 1$, the pipeline of the second loop, cannot be directly expressed as a multi-dimensional schedule. To get a similar behavior, we need to assume that it has the same schedule (i, j) but starts one "physical" clock-cycle later.) The maximal lexicographic difference is $\vec{t}_S = (1, 1 - N)$, which corresponds to the approximation of DS as the polytope with vertices $(0, N - 1)$, $(0, 1 - N)$, $(1, 1 - N)$, and $(-1, N - 1)$ (with volume $2N - 2$), see Figure 2. We will

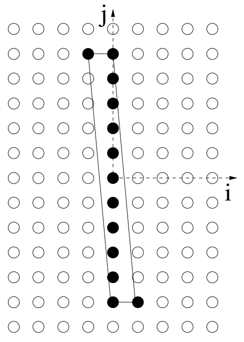

Figure 2: An approximation of the set DS.

show that the optimal memory size is closely related to the volume of the set DS. Therefore, with this slight approximation, we already see that we will loose an order of magnitude (when N is large) in memory size compared to the optimal solution (which, as seen before, is 2)! □

2.3 Strictly Admissible Integral Lattices

To analyze the link between modular allocations and the set of conflicting indices, we introduce admissible integral lattices. Let $\vec{a}_1, \ldots, \vec{a}_n$ be n linearly independent points in $\mathbb{R}^n$. The set Λ of points $\vec{x} = u_1\vec{a}_1 + \cdots + u_n\vec{a}_n$ where $u_1, \ldots, u_n$ are integers is called a **lattice**. The system of points $(\vec{a}_1, \ldots, \vec{a}_n)$ is a basis of Λ; $\det(\vec{a}_1, \ldots, \vec{a}_n)$, a quantity that does not depend on the choice of a basis for Λ, is called the determinant of Λ, denoted by $d(\Lambda)$. We write $\vec{x} = A\vec{u}$ where A is the matrix with column vectors $\vec{a}_1, \ldots, \vec{a}_n$ and $\Lambda = A\mathbb{Z}^n$. Let K be an arbitrary set in $\mathbb{R}^n$. A lattice is called **admissible** for K if it has no point $\neq 0$ in the interior of K. If is called **strictly admissible** for K if it does not contain a point $\neq 0$ in K. The **critical determinant** of a set K is the quantity $\Delta(K) = \inf\{d(\Lambda) \mid \Lambda \text{ strictly admissible for } K\}$. We also define $\Delta^0(K) = \inf\{d(\Lambda) \mid \Lambda \text{ admissible for } K\}$ and the quantity we are really interested in, which is $\Delta_{\mathbb{Z}}(K) = \inf\{d(\Lambda) \mid \Lambda \subset \mathbb{Z}^n \text{ strictly admissible for } K\}$. While $\Delta(K)$ may not be attained, $\Delta_{\mathbb{Z}}(K)$ is an integer and there always exists an integral lattice Λ such that $d(\Lambda) = \Delta_{\mathbb{Z}}(K)$. The next theorem gives the link we were looking for.

THEOREM 1. *A valid modular mapping $(M, \vec{b})$ corresponds to a strictly admissible integral lattice for DS whose determinant divides $\prod_i b_i$. Conversely, if Λ is a strictly admissible integral lattice for DS, with basis $A = Q_1SQ_2$ (Smith form [18]), $S = diag(\vec{s})$, then the mapping $(Q_1^{-1}, \vec{s})$ is valid.*

PROOF. It is not difficult to see that for a $m \times n$ integral matrix M and an integral vector $\vec{b}$, the set of integral points $\vec{x}$ such that $M\vec{x} \bmod \vec{b} = 0$ is a sublattice of $\mathbb{Z}^n$, whose determinant divides the product of the components of $\vec{b}$, and a basis for this lattice can be built by integral matrix computations (see for example [2]). Furthermore, given any sublattice Λ of $\mathbb{Z}^n$, we can always build a matrix M and a vector $\vec{b}$ such that $M\vec{x} \bmod \vec{b} = 0$ if and only if $\vec{x} \in \Lambda$. Indeed, consider a basis A of Λ. Write $A = Q_1SQ_2$, the Smith form of A, where S is diagonal, and Q_1, Q_2 are unimodular. Let $\vec{b}$ be the vector such that $S = \text{diag}(\vec{b})$ and $M = Q_1^{-1}$. Then $M\vec{x} \bmod \vec{b} = 0$ iff there exists $\vec{u} \in \mathbb{Z}^n$ such that $M\vec{x} = S\vec{u}$ iff there exists $\vec{v} \in \mathbb{Z}^n$ ($\vec{v} = Q_2^{-1}\vec{u}$) such that $\vec{x} = A\vec{v}$. In other words, there is a direct correspondence between sublattices Λ of $\mathbb{Z}^n$ and modular allocations $(M, \vec{b})$, where M is a $n \times n$ unimodular matrix and the product of the components of $\vec{b}$ is the determinant of the lattice Λ. □

This gives two dual views on the problem. In Section 3.2, we build a strictly admissible integral lattice, reasoning in the space where DS is defined, and deduce a valid mapping, while in Section 3.3, we build the mapping directly, reasoning on the matrix M and the vector $\vec{b}$.

The rest of the paper is devoted to a study of the quantity $\Delta_{\mathbb{Z}}(K)$ for a 0-symmetric convex body K with positive volume. We will consider the general case, i.e., we will ignore the fact that, for our original problem, $K = DS$ was built in some particular way (from affine schedules for example): further studies will be needed to better exploit particular cases. [4] What we seek here is a general framework that defines the problem and general mechanisms that are robust enough to handle even extreme cases that we can encounter for example with "skewed" schedules as used in [3]. However, we will make two important assumptions. First, for computability reasons, we will assume that K is a rational

[3]Actually, to make notations simpler, we will even use the same notation, when there is no ambiguity, for the polytope and for the set of integer points it contains.

[4]Note however that this general case is, at least in theory, possible; given a 0-symmetric n-dimensional polytope K, simply consider the case of n nested loops, storing (with no reuse) a portion – equal to a translated copy of K – of a n-dimensional array; in this case, $DS = 2K$ can thus be an arbitrarily "skewed" 0-symmetric polytope. A typical example would be a loop that accesses only the three main diagonals of a 2D square array of size N, and we should be able to find that the mapping $(i, j \bmod 3)$ is enough to store $3N$ values instead of N^2.

polytope, i.e., defined by integral linear inequalities. Second, to be able to evaluate the performances of heuristics, we will assume that K is full-dimensional, and even more, that *it contains n linearly independent integer points*; otherwise, we can make a preliminary change of basis (at least conceptually for the moment) and work with the polytope K', intersection of K with the smallest vector subspace that contains all integer points in K. This will allow us to talk about the volume of K' and to get upper bounds in terms of this volume (otherwise, this would not be possible since $\Delta_{\mathbb{Z}}(K)$ is a positive integer while the volume of K, even if positive, could be arbitrarily small).

3. EXPLORING THE QUANTITY $\Delta_{\mathbb{Z}}(K)$

Many theoretical results exist for the quantities $\Delta(K)$ and $\Delta^0(K)$ (see [10] for an extensive study), but it is an open problem to be able to compute $\Delta(K)$ for most bodies K, even simple ones. However, we can wonder if the combinatorial nature of $\Delta_{\mathbb{Z}}(K)$ (sublattices of $\mathbb{Z}^n$ instead of general lattices in $\mathbb{R}^n$) can make the problem easier. But, as Lemma 1 shows, there is no hope, for large bodies, to be able to do better. Our goal will be then to try to adapt to $\Delta_{\mathbb{Z}}(K)$ well-known lower and upper bounds for $\Delta(K)$, and to discuss heuristics within this framework.

We have $\Delta^0(K) \leq \Delta(K) \leq \Delta_{\mathbb{Z}}(K)$. Theorem 17.3 in [10] states that, if K is a star body [5], $\Delta(K) = \Delta^0(K)$. Lemma 1 shows that for large bodies, $\Delta_{\mathbb{Z}}(K) \approx \Delta(K)$. Before, note the important following "scaling" property: for all $\lambda > 0$, $\Delta(\lambda K) = \lambda^n \Delta(K)$ while $\Delta_{\mathbb{Z}}(\lambda K) \leq \lambda^n \Delta_{\mathbb{Z}}(K)$.

LEMMA 1. *For a bounded star body K,* $\lim_{\lambda\to\infty} \dfrac{\Delta_{\mathbb{Z}}(\lambda K)}{\Delta(\lambda K)} = 1$

PROOF. Remark: the proof of this lemma has not much interest and can be skipped at first reading.

Let $\epsilon > 0$ and let Λ be a strictly admissible lattice for K with $d(\Lambda) \leq \Delta(K)(1+\epsilon)$. Define the quantity $\lambda_1(K,\Lambda) = \inf\{\lambda > 0 \mid \lambda K \cap \Lambda \neq \{0\}\}$. We have $\lambda_1(K,\Lambda) > 1$. Since, for a bounded star body, $\lambda_1(K,\Lambda)$ depends continuously on Λ (see [10][p. 185]), there exists a rational matrix A_ϵ, with $\lambda_1(K, A_\epsilon\mathbb{Z}^n) > 1$ (thus strictly admissible for K) and $\det(A_\epsilon) \leq \Delta(K)(1+2\epsilon)$. Let μ_ϵ be the common denominator of the coefficients of A_ϵ: $\mu_\epsilon A_\epsilon \mathbb{Z}^n$ is a sublattice of $\mathbb{Z}^n$, strictly admissible for $\mu_\epsilon K$, and therefore strictly admissible for any μK with $\mu \leq \mu_\epsilon$. Now, let $\lambda > 0$ and let $k = \lceil \frac{\lambda}{\mu_\epsilon} \rceil$, i.e., $(k-1)\mu_\epsilon < \lambda \leq k\mu_\epsilon$. The lattice $k\mu_\epsilon A_\epsilon \mathbb{Z}^n$ is strictly admissible for λK. Furthermore:

$$\begin{aligned}\Delta(\lambda K) &\leq \Delta_{\mathbb{Z}}(\lambda K) \leq \det(k\mu_\epsilon A_\epsilon)\\ &\leq k^n \mu_\epsilon^n \Delta(K)(1+2\epsilon) \leq (\lambda+\mu_\epsilon)^n \Delta(K)(1+2\epsilon)\end{aligned}$$

Therefore:

$$1 \leq \frac{\Delta_{\mathbb{Z}}(\lambda K)}{\Delta(\lambda K)} \leq (1+\frac{\mu_\epsilon}{\lambda})^n (1+2\epsilon)$$

which means that the ratio $\Delta_{\mathbb{Z}}(\lambda K)/\Delta(\lambda K)$ is less than $1+3\epsilon$ when λ is large enough. In other words, the limit of $\Delta_{\mathbb{Z}}(\lambda K)/\Delta(\lambda K)$, when λ grows to infinity, is 1. ☐

Note however that, since the search is limited to integral lattices, it is possible to determine the exact value of $\Delta_{\mathbb{Z}}(K)$ with an exhaustive search (through not practical for bodies with very large volumes). Indeed, since any integral lattice has a basis whose corresponding matrix is nonnegative, lower triangular, with all components below the diagonal strictly smaller than the diagonal element of the same row (Hermite form, see [18]), we can generate all integral lattices with a given determinant d. For each such lattice, we can check (either with integer linear programming or by solving an integral triangular system, for each fixed element in DS, if we can enumerate them) that it intersects K only in 0. If we find no strictly admissible lattice with determinant d, we continue the search with the value $d+1$. The procedure will stop with a solution for a determinant less than a linear function of $\mathrm{Vol}(K)$ as the next sections show.

Example 1 (Cont'd). The body DS in Figure 1 has a very small volume, equal to 2, which makes the exhaustive search practical. We just have to consider 4 lattices, the lattice generated by a) the vectors $(1,0)$ and $(1,0)$ (the only lattice with determinant 1), b) the vectors $(1,0)$ and $(0,2)$, c) the vectors $(2,0)$ and $(0,1)$, and d) the vectors $(1,1)$ and $(0,2)$. Depending on the parity of N, we find the two optimal allocations (with memory size 2) mentioned before, $j \bmod 2$ (Case b and N even) and $i+j \bmod 2$ (Case d and N odd). Applying this principle "blindly" to the body of Figure 2, when N is large, will be obviously too expensive. ☐

3.1 Lower Bounds & Minkowski's Theorems

Remember the definition of MAXLIVE. Suppose that there is a set S such that $S \times S \subset CS$. All integer points in S must have different values through a valid modular allocation $(M, \vec{b})$, thus $\prod_{i=1}^n b_i \geq \mathrm{Card}(S)$, and finally $\Delta_{\mathbb{Z}}(DS) \geq \mathrm{Card}(S)$. We can make a similar reasoning for a 0-symmetric bounded convex body K in $\mathbb{R}^n$. Let $\Lambda \subset \mathbb{Z}^n$ be a strictly admissible lattice for K. Then 2Λ is strictly admissible for $2K$ and we can build a corresponding modular allocation $(M, 2\vec{b})$ (as in Theorem 1) such that $\vec{x} \in 2K$ and $M\vec{x} \bmod 2\vec{b} = 0$ imply $\vec{x} = 0$. All integer points in K have different values through this modular allocation (indeed, if $\vec{x} \in K$ and $\vec{y} \in K$ have the same value, then $\vec{z} = \vec{x} - \vec{y} \in 2K$ and $M\vec{z} = 0 \bmod 2\vec{b}$), therefore $2^n \prod_{i=1}^n b_i \geq \mathrm{Card}(K)$, and finally $\Delta_{\mathbb{Z}}(K) \geq \mathrm{Card}(K)/2^n$.

We can get similar inequalities using volumes, thanks to Minkowski's first theorem (see for example [14]). If Λ is a lattice of $\mathbb{R}^n$ and K a 0-symmetric bounded convex body such that $\mathrm{Vol}(K) > 2^n d(\Lambda)$, then K contains a nonzero lattice point of Λ. In other words, if Λ is a strictly admissible lattice for K, then $d(\Lambda) \geq \mathrm{Vol}(K)/2^n$. This shows that $\Delta_{\mathbb{Z}}(K) \geq \mathrm{Vol}(K)/2^n$ (with strict inequality if K is closed). Similarly, if there is a convex set S such that $S \times S \subset CS$, then $S - S \subset DS$ and $\Delta_{\mathbb{Z}}(DS) \geq \mathrm{Vol}(S-S)/2^n$. Because of Brunn-Minkowski theorem (see [10][p.12]), we get $\Delta_{\mathbb{Z}}(DS) \geq \mathrm{Vol}(S)$, an inequality similar to the one for MAXLIVE, but with volumes.

Example 2 (Cont'd). For the set DS of Figure 2, the first theorem of Minkowski leads to the inequality $\Delta_{\mathbb{Z}}(DS) \geq (N-1)/2$. As mentioned before, this shows that this (apparently slight) over-approximation prevents valid modular allocations with a $O(1)$ memory size. ☐

For an upper bound, we could expect to adapt Minkowski-Hlawka theorem (see [9] for an elementary "constructive"

[5]A star body is a closed set K such that λx is in the interior of K, for all $x \in K$ and $|\lambda| < 1$. A full-dimensional convex region is a star body, so Lemma 1 applies to DS with our assumptions.

proof), which states that if K is any bounded Jordan measurable set in $\mathbb{R}^n$, then $\Delta(K) \leq \mathrm{Vol}(K)$. Unfortunately, the construction needs to "refine" the lattice $\mathbb{Z}^n$ (i.e., it uses lattices for which $\mathbb{Z}^n$ is a sublattice) while, for $\Delta_{\mathbb{Z}}(K)$, we want to restrict to sublattices of $\mathbb{Z}^n$. Thus, it is not clear whether an equivalent of this theorem can be given for $\Delta_{\mathbb{Z}}(K)$. Nevertheless, since $\Delta_{\mathbb{Z}}(K)/\mathrm{Vol}(K)$ is lower bounded by a constant that depends on the problem dimension n only, we can try to look for heuristics that build integral lattices Λ, strictly admissible for K, such that the ratio $d(\Lambda)/\mathrm{Vol}(K)$ is upper bounded by a constant that depends on n only. This would give us heuristics that are **optimal up to a multiplicative factor**. Before, let us recall some important notions concerning convex bodies, successive minima, and polar reciprocal bodies (see for example [10] for details).

When K is a 0-symmetric closed bounded convex body, then $F(\vec{x}) = \inf\{\lambda > 0 \mid \vec{x} \in \lambda K\}$ defines a distance function such that $F(\alpha\vec{x}) = |\alpha|F(\vec{x})$, called the **gauge (or distance) function** of K. The set K is then the set of points $\vec{x}$ such that $F(\vec{x}) \leq 1$. Given a lattice Λ, the n **successive minima** of K with respect to Λ are defined, for $1 \leq i \leq n$, by $\lambda_i(K,\Lambda) = \inf\{\lambda > 0 \mid \dim(\mathrm{Vect}(\lambda K \cap \Lambda)) \geq i\}$. In other words, $\lambda_i(K,\Lambda)$ is the smallest value λ such that K contains at least i linearly independent points in Λ with $F(x) \leq \lambda$. As mentioned earlier, here we will restrict to the case where $\Lambda = \mathbb{Z}^n$ (and we will simply write $\lambda_i(K)$), but most upcoming results can be extended to the case of regularly spaced integer points in K (i.e., a set of the form $K \cap \Lambda$). The **polar reciprocal** K^* of K is the set $\{\vec{y} \in \mathbb{R}^n \mid \vec{x}.\vec{y} \leq 1 \text{ for all } \vec{x} \in K\}$, where . is the inner product of $\vec{x}$ and $\vec{y}$. The set K^* is also a 0-symmetric closed bounded convex body and its gauge function F^* is such that $F^*(\vec{y}) = \sup\{\vec{x}.\vec{y} \mid \vec{x} \in K\}$. Furthermore $(K^*)^* = K$.

3.2 Heuristics Based on the Body K

We first describe a general and intuitive mechanism that, given n linearly independent vectors $(\vec{a}_1, \dots, \vec{a}_n)$, allows us to scale the vectors $(\vec{a}_i)_{1\leq i\leq n}$ enough (with a multiplicative factor ρ_i) so that the vectors $(\rho_i\vec{a}_i)_{1\leq i\leq n}$ generate a strictly admissible integral lattice for K. For this, we make use of the functions $F_i(\vec{x}) = \inf\{F(\vec{y}) \mid \vec{y} \in \vec{x} + \mathrm{Vect}(\vec{a}_1, \dots, \vec{a}_{i-1})\}$. The function F_i is connected to the gauge function of the projection of K along the vectors $\vec{a}_1, \dots, \vec{a}_{i-1}$. Using linear programming duality, it is possible to show that $F_i(\vec{x}) = \sup\{\vec{z}.\vec{x} \mid \vec{z} \in K^*, \vec{z}.\vec{a}_1 = \dots = \vec{z}.\vec{a}_{i-1} = 0\}$. For more details about this result and the functions F_i, see [17] and its annotated version [11].

Heuristic 1.

- *Choose a set of n linearly independent integral vectors $(\vec{a}_1, \dots, \vec{a}_n)$.*
- *Compute, for each i, $0 \leq i \leq n$, the quantity $F_i(\vec{a}_i) = \inf\{F(\vec{y}) \mid \vec{y} \in \vec{a}_i + \mathit{Vect}(\vec{a}_1, \dots, \vec{a}_{i-1})\}$.*
- *Choose n integers ρ_i such that $\rho_i F_i(\vec{a}_i) > 1$.*
- *Define Λ, lattice generated by the vectors $(\rho_i\vec{a}_i)_{1\leq i\leq n}$.*

Theorem 2. *The lattice Λ built by Heuristic 1 is a strictly admissible integral lattice for K. Furthermore, if the vectors $(\vec{a}_i)_{1\leq i\leq n}$ form a basis of $\mathbb{Z}^n$ such that $F_i(\vec{a}_i) \leq 1$ for $1 \leq i \leq n$, then the smallest choice for $(\rho_i)_{1\leq i\leq n}$ leads to $d(\Lambda)\prod_i F_i(\vec{a}_i) \leq 2^n$, hence $d(\Lambda) \leq (n!)^2\, \mathit{Vol}(K)$.*

Proof. Let $\vec{x} \in \Lambda$, $\vec{x} = \sum_{j=1}^{i} x_j\rho_j\vec{a}_j$ with $x_1, \dots, x_i$ integers, and $x_i \neq 0$. What we want is that, for such a point $\vec{x}$, ρ_i is chosen large enough so that $F(\vec{x}) > 1$ (i.e., $\vec{x}$ is out of K). Let us check this. By construction, $\vec{x}/(\rho_i x_i)$ belongs to $\vec{a}_i + \mathrm{Vect}(\vec{a}_1, \dots, \vec{a}_{i-1})$, therefore $F(\vec{x}/(\rho_i x_i)) \geq F_i(\vec{a}_i)$. Thus, $F(\vec{x}) \geq |x_i|\rho_i F_i(\vec{a}_i) \geq \rho_i F_i(\vec{a}_i) > 1$. This shows that $\vec{x} \notin K$. Since all $\vec{x} \in \Lambda$, $\vec{x} \neq 0$, have been considered this way, Λ is a strictly admissible integral lattice for K.

Now suppose that the vectors $(\vec{a}_i)_{1\leq i\leq n}$ form a basis of $\mathbb{Z}^n$, then $d(\Lambda) = \prod_i \rho_i$. Choose the smallest possible integer for ρ_i, i.e., $\rho_i = \lfloor 1/F_i(\vec{a}_i)\rfloor + 1$. If $F_i(\vec{a}_i) > 1$, then $\rho_i = 1$ and it can be ignored in the product. Otherwise $\rho_i \leq 1/F_i(\vec{a}_i) + 1 \leq 2/F_i(\vec{a}_i)$ and $d(\Lambda) = \prod_{i\in I}\rho_i \leq \prod_{i\in I}(2/F_i(\vec{a}_i))$, where I is the set of indices i such that $F_i(\vec{a}_i) \leq 1$. When $I = [1..n]$, it remains to find a lower bound for $\prod_i F_i(\vec{a}_i)$. Let A be the matrix with column vectors $(\vec{a}_i)_{1\leq i\leq n}$ and consider $(\vec{b}_i)_{1\leq i\leq n}$ the row vectors of A^{-1}. We have:

$$\begin{aligned}
F_i(\vec{a}_i) &= \inf\{F(y) \mid \vec{y} \in \vec{a}_i + \mathrm{Vect}(\vec{a}_1, \dots, \vec{a}_{i-1})\}\\
&= \inf\{F(y) \mid \vec{b}_i.\vec{y} = 1,\ \vec{b}_{i+1}.\vec{y} = \dots = \vec{b}_n.\vec{y} = 0\}\\
&= \inf\{\rho > 0 \mid \vec{y} \in \rho K,\ \vec{b}_i.\vec{y} = 1,\ \vec{b}_{i+1}.\vec{y} = \dots = \vec{b}_n.\vec{y} = 0\}\\
&= \inf\{\rho > 0 \mid \vec{y} \in K,\ \vec{b}_i.\vec{y} = 1/\rho,\ \vec{b}_{i+1}.\vec{y} = \dots = \vec{b}_n.\vec{y} = 0\}\\
&= 1/\sup\{\vec{b}_i.\vec{y} \mid \vec{y} \in K,\ \vec{b}_{i+1}.\vec{y} = \dots = \vec{b}_n.\vec{y} = 0\}
\end{aligned}$$

Now let $\vec{c}_i = \vec{b}_{n+1-i}$. The last expression is nothing but $1/F_i^*(\vec{c}_i)$ (property mentioned before applied to K^* and the basis $(\vec{c}_i)_{1\leq i\leq n}$), thus $F_i(\vec{a}_i)F_i^*(\vec{c}_i) = 1$ for $1 \leq i \leq n$. We now use the inequality $\mathrm{Vol}(K^*)\prod_i F_i^*(\vec{c}_i) \leq 2^n$ (see again [17]) valid for any basis $(\vec{c}_i)_{1\leq i\leq n}$ of $\mathbb{Z}^n$ and we get:

$$d(\Lambda) \leq 2^n \prod_{i=1}^{n} F_i^*(\vec{c}_i) \leq 4^n/\mathrm{Vol}(K^*) \leq (n!)^2\mathrm{Vol}(K)$$

the last inequality because $\mathrm{Vol}(K)\mathrm{Vol}(K^*) \geq 4^n/(n!)^2$ (see Theorem 14.4 in [10]). □

Not any basis leads to the performance guarantee of Theorem 2 since it is not true that $\prod_{i\in I} F_i(\vec{a}_i)$ is always lower bounded (in I, we excluded the indices i with $F_i(\vec{a}_i) \geq 1$). However, if K is not too skewed in the canonical basis (as it is often the case for DS in practice), for example if $\vec{a}_i \in K$, then $F_i(\vec{a}_i) \leq F(\vec{a}_i) \leq 1$ and using Heuristic 1 in the canonical basis will give a solution with guaranteed performance. But, in the general case, we need to choose a more adequate basis $(\vec{a}_i)_{1\leq i\leq n}$. Remember that we assumed that K contains n linearly independent integral points. Consider n such points $(\vec{x}_i)_{1\leq i\leq n}$. They may not form a basis of $\mathbb{Z}^n$, but we can build a basis $(\vec{a}_i)_{1\leq i\leq n}$ of $\mathbb{Z}^n$ such that $\mathrm{Vect}(\vec{x}_1, \dots, \vec{x}_i) = \mathrm{Vect}(\vec{a}_1, \dots, \vec{a}_i)$ for $1 \leq i \leq n$. Then, we get $F_i(\vec{a}_i) \leq F_i(\vec{x}_i) \leq F(\vec{x}_i) \leq 1$ and the previous theorem applies. This is for example the case when $F(\vec{x}_i) = \lambda_i(K)$, the successive minima of K, which can be built, with some effort, thanks to integer linear programming (ILP).

With a slight modification of the lattice Λ built by Heuristic 1, we can even get the same performance with a 1D modular allocation. Indeed, given a basis $(\vec{a}_i)_{1\leq i\leq n}$ of $\mathbb{Z}^n$ and the corresponding ρ_i, consider the lattice Λ' generated by the vectors $(\rho_i\vec{a}_i + \vec{a}_{i-1})_{1\leq i\leq n}$ with $\vec{a}_0 = 0$. The same reasoning shows that Λ' is strictly admissible for K because, if $\vec{x} = x_1\rho_1\vec{a}_1 + \sum_{j=2}^{i} x_j(\rho_j\vec{a}_j + \vec{a}_{j-1})$, then $\vec{x} = \sum_{j=1}^{i-1}(x_j\rho_j + x_{j+1})\,\vec{a}_j + x_i\rho_i\vec{a}_i$ and $\vec{x}/(\rho_i x_i)$ belongs to $\vec{a}_i + \mathrm{Vect}(\vec{a}_1, \dots, \vec{a}_{i-1})$. Furthermore, Λ' and Λ have the same determinant and it is easy to see that all diagonal components of the Smith form of Λ' are equal to 1 except the last one. Going back to the associated mapping (as in Section 2.3), this means that all components of $\vec{b}$ are equal to 1, so that only the last row of M matters: it is a one-dimensional modular mapping!

Note also that, even if the $F_i(\vec{a}_i)$ are not all smaller than 1, we can still get a performance guarantee if we consider a basis $(\vec{a}_i)_{1\leq i\leq n}$ of $\mathbb{Z}^n$ such that each ρ_i can be upper bounded. This is the case for a generalized reduced basis in the sense of [17] for which $F_i(\vec{a}_i) \geq \lambda_i(K)(1/2-\epsilon)^{i-1}$ for $0 < \epsilon < 1/2$. For example, with $\epsilon = 1/4$, we get:

$$\begin{aligned} d(\Lambda) &\leq \textstyle\prod_{i=1}^n (\frac{1}{F_i(\vec{a}_i)} + 1) \leq \prod_{i=1}^n (\frac{4^{i-1}}{\lambda_i(K)} + 1) \\ &\leq \textstyle\prod_{i=1}^n \frac{4^i}{\lambda_i(K)} \leq 2^{n^2} n!\mathrm{Vol}(K) \end{aligned}$$

the third inequality because we assumed that K contains n linearly independent integral points (i.e., $\lambda_i(K) \leq 1$ for all i), the fourth inequality because $\prod_i \lambda_i(K)\mathrm{Vol}(K) \geq 2^n/n!$ (lower bound in Minkowski's second theorem [10][p. 62] for a 0-symmetric convex body K). It can be shown that this technique will work even if K contains only $p < n$ linearly independent integral points, by first identifying the smallest vector space V of dimension $q \leq p$ that contains K thanks to rational linear programming. The performance will then depend both on p and q.

Example 1 (Cont'd). Consider again the body $K = DS$ in Figure 1: we have $\lambda_1(K) = \lambda_2(K) = 1$, reached by the 2 linearly independent vectors $(0,1)$ and $(1,1-N)$. Using Heuristic 1 for the basis $\vec{a}_1 = (1,0)$, $\vec{a}_2 = (0,1)$ leads to $F_1(\vec{a}_1) \geq 1$ (thus $\rho_1 = 1$) and $F_2(\vec{a}_2) = 1/(N-1)$ (thus $\rho_2 = N$), for a memory of size N. With the basis in the opposite order, we get $\rho_1 = 2$, then $\rho_2 = 2$, for a lattice with determinant 4 (depicted in Figure 1 as grey points), with corresponding mapping $(j \bmod 2, i \bmod 2)$. The lattice Λ', which we defined previously to get a 1D mapping, is the lattice generated by $2(0,1) = (0,2)$ and $2(1,0)+(0,1) = (2,1)$, which corresponds to the mapping $i + 2j \bmod 4$. Considering a basis formed with successive minima leads to the basis with vectors $(0,1)$ and $(1,1-N)$, with $\rho_1 = \rho_2 = 2$. □

We point out that we can obtain another approximation scheme by adapting to $\Delta_{\mathbb{Z}}(K)$ the technique that Rogers developed for $\Delta(K)$ (see Theorem 18.1 in [10]). While the functions F_i are based on rational points, the technique of Rogers captures integral points with more accuracy, leading to a better (by a factor $n!$) upper bound in the worse case. It also seems to be more suitable when K is not a polytope.

THEOREM 3. *If K is a 0-symmetric convex body in $\mathbb{R}^n$, with n linearly independent integral points, one can build an integral lattice Λ, strictly admissible for K, such that $d(\Lambda)\prod_{i=1}^n \lambda_i(K) \leq 2^n$ and, consequently, $d(\Lambda) \leq n!\ Vol(K)$.*

PROOF. Define ρ_i to be the largest power of 2 such that $\rho_i\lambda_i(K) > 1$. By definition, $\rho_i\lambda_i(K) \leq 2$ and since $\lambda_i(K) \leq 1$ (we assumed that K contains n linearly independent integer points), ρ_i is an integer. Furthermore, since $\lambda_i(K) \leq \lambda_{i+1}(K)$, we have $\rho_i \geq \rho_{i+1}$ and $m_i = \rho_i/\rho_{i+1}$ is an integer. Since $1/\rho_i < \lambda_i(K)$, $\dim(\mathrm{Vect}(K/\rho_i \cap \mathbb{Z}^n)) \leq i-1$. Furthermore, if $\vec{x} \in K/\rho_i \cap \mathbb{Z}^n$, then $m_i\vec{x} \in K/\rho_{i+1} \cap \mathbb{Z}^n$. Thus, if L_i denotes the linear subspace spanned by $K/\rho_i \cap \mathbb{Z}^n$, then $\dim L_i \leq i-1$ and $L_i \subset L_{i+1}$. We can then build a basis $(\vec{a}_i)_{1\leq i\leq n}$ of $\mathbb{Z}^n$ such that $L_i \subset \mathrm{Vect}(\vec{a}_1, \ldots, \vec{a}_{i-1})$.

Now consider the sublattice Λ of $\mathbb{Z}^n$ with basis $(\rho_1\vec{a}_1, \ldots, \rho_n\vec{a}_n)$. Let $\vec{x} \in \Lambda$, $\vec{x} = \sum_{j=1}^i x_j\rho_j\vec{a}_j$ with $x_1, \ldots, x_i$ integers, and $x_i \neq 0$. We get $\vec{x}/\rho_i = v_1\vec{a}_1 + \cdots + v_i\vec{a}_i$, where $v_1, \ldots, v_i$ are integers, $v_i \neq 0$ (note the subtle difference with the proof of Theorem 2). Therefore, $\vec{x}/\rho_i \in \mathbb{Z}^n$. Moreover, $\vec{x}/\rho_i \notin K/\rho_i \cap \mathbb{Z}^n$ since $K/\rho_i \cap \mathbb{Z}^n \subset L_i \subset L(\vec{a}_1, \ldots, \vec{a}_{i-1})$. Hence, $\vec{x}/\rho_i \notin K/\rho_i$, i.e., $\vec{x} \notin K$. It follows that Λ is strictly admissible for K. Then we get:

$$d(\Lambda)\prod_{i=1}^n \lambda_i(K) = \prod_{i=1}^n \rho_i\lambda_i(K) \leq 2^n$$

Finally, with $\prod_{i=1}^n \lambda_i(K)\mathrm{Vol}(K) \geq 2^n/n!$, we get the desired upper bound $d(\Lambda) \leq n!\,\mathrm{Vol}(K)$. This upper bound may not be the best possible. In particular, the first inequality is valid even for an arbitrary K. There must be space for improvement to better exploit the fact that K is a 0-symmetric convex body. However, there does not seem to exist a simple mechanism to derive, from this solution, a 1D mapping as for Heuristic 1. □

Example 1 (Cont'd). In this very particular example, the technique developed in the proof of Theorem 3 shows that whatever the choice of a basis $(\vec{a}_1, \vec{a}_2)$ of $\mathbb{Z}^n$, the lattice generated by $2\vec{a}_1$ and $2\vec{a}_2$ is strictly admissible. Thus, this shows that for any unimodular matrix M, the allocation $M\vec{x} \bmod \vec{b}$ is valid for $\vec{b} = (2,2)$, which is actually obvious since DS contains only primitive vectors while $M\vec{x} = 0 \bmod \vec{b}$ iff $\vec{x} = 2M^{-1}\vec{y}$ for some integral vector $\vec{y}$. □

Example 2 (Cont'd). For the set in Figure 2, $\lambda_1(K) = 1/(N-1)$ reached for the vector $(0,1)$ and $\lambda_2(K) = 1$ reached for the vector $(1,1-N)$. Here, the technique of Rogers chooses two powers of 2, ρ_1 and ρ_2, with $\rho_1\lambda_1(K) > 1$ and $\rho_2\lambda_2(K) > 1$, for example $\rho_1 = 2^p \geq N$ and $\rho_2 = 2$. Then it chooses $\vec{a}_1 = (0,1)$ (i.e., in the direction of the first minimum) and $\vec{a}_2$, any vector such that $(\vec{a}_1, \vec{a}_2)$ is a basis of $\mathbb{Z}^n$, for example $(1,0)$. By construction, the lattice generated by $2^p\vec{a}_1$ and $2\vec{a}_2$ is strictly admissible, which corresponds to the mapping $(i \bmod 2, j \bmod 2^p)$. Note that, this time, the heuristic imposes $\vec{a}_1 = (0,1)$: for example, the mapping $(i \bmod 2^p, j \bmod 2)$ is not valid, one needs to be careful about the unequal extents of the polytope. □

3.3 Heuristics Based on the Body K^*

Instead of working in the space of admissible lattices for K, working with K^*, the polar reciprocal of K, amounts to work in the space of matrices M of modular allocations. We now consider an approximation scheme, which is the dual of Heuristic 1 in the sense that it uses the functions F_i^* instead of the functions F_i. This mechanism is exactly the principle used by Lefebvre and Feautrier [15], but generalized to any set of n linearly independent integral vectors.

HEURISTIC 2.

- *Choose a set of n linearly independent integral vectors $(\vec{c}_1, \ldots, \vec{c}_n)$.*
- *Compute, for each i, $0 \leq i \leq n$, the quantity $F_i^*(\vec{c}_i) = \sup\{\vec{c}_i.\vec{z} \mid \vec{z} \in K, \vec{c}_1.\vec{z} = \ldots = \vec{c}_{i-1}.\vec{z} = 0\}$.*
- *Choose n integers ρ_i such that $\rho_i > F_i^*(\vec{c}_i)$.*
- *Define M the matrix with row vectors $(\vec{c}_i)_{1\leq i\leq n}$ and $\vec{b}$ the vector such that $b_i = \rho_i$.*

Theorem 4. *The modular mapping built by Heuristic 2 is a valid mapping for K. Furthermore, if the "successive widths" of K in the directions $\vec{c}_i$ are more than 1 (i.e., $F_i^*(\vec{c}_i) \geq 1$) then, with the smallest choice for $(\rho_i)_{1\leq i\leq n}$, the corresponding lattice Λ is such that $d(\Lambda) \leq 2^n \prod_i F_i^*(\vec{c}_i)$, hence $d(\Lambda) \leq (n!)^2$ Vol(K).*

Proof. This process is nothing but the dual of the technique used for Heuristic 1, thanks to the relation (proved in Theorem 2) $F_i(\vec{a}_i)F^*_{n+i-1}(\vec{c}_{n+1-i}) = 1$. Therefore, we could use the same proof, reasoning with the dual basis of $(\vec{c}_i)_{1\leq i\leq n}$. Nevertheless, to better understand the heuristic of Lefebvre and Feautrier [15] which uses this mechanism (but in one particular basis, the canonical basis in which $DS = K$ was defined), here is a direct proof of its validity.

Let $\vec{x} \in K$ with $M\vec{x} \bmod \vec{b} = 0$. We have $\vec{c}_1.\vec{x} = 0 \bmod b_1$, but $b_1 > F_1^*(\vec{c}_1) = \sup\{\vec{c}_1.\vec{x} \mid \vec{x} \in K\} = \sup\{|\vec{c}_1.\vec{x}| \mid \vec{x} \in K\}$ because K is 0-symmetric. Thus, $\vec{c}_1.\vec{x} = 0$. Then, $\vec{c}_2.\vec{x} = 0 \bmod b_2$, but $b_2 > F_2^*(\vec{c}_2) = \sup\{|\vec{c}_2.\vec{x}| \mid \vec{x} \in K, \vec{c}_1.\vec{x} = 0\}$, thus $\vec{c}_2.\vec{x} = 0$. Continuing this process, we get $\vec{c}_i.\vec{x} = 0$ for all i. Since the vectors $\vec{c}_i$ are n linearly independent vectors, this implies $\vec{x} = 0$, thus $(M, \vec{b})$ is a valid modular mapping.

As mentioned in Section 2.3, the determinant of the corresponding lattice Λ divides the product of the components of $\vec{b}$, thus $d(\Lambda) \leq \prod_i \rho_i$. With the smallest possible ρ_i, we get $d(\Lambda) \leq \prod_i(\lfloor F_i^*(\vec{c}_i)\rfloor + 1)$, and finally, when $F_i^*(\vec{c}_i) \geq 1$, exactly the same upper bound as in Theorem 2 since this is how we derived it. $\square$

As for Heuristic 1, we can analyze the performance of Heuristic 2 for particular sets of linearly independent integral vectors $(\vec{c}_i)_{1\leq i\leq n}$. But inequalities have to be used in the opposite direction. We will use the fact that, when K contains n linearly independent integral vectors (i.e., for all i, $\lambda_i(K) \leq 1$), we have $\lambda_i(K^*) \geq 1$ (this is because $\lambda_i(K^*)\lambda_{n+1-i}(K) \geq 1$, see Theorem 14.5 in [10]).

Consider n linearly independent integral vectors $\vec{c}_i$ such that $F^*(\vec{c}_i) = \lambda_i(K^*)$ (the successive minima of K^*), then $F_i^*(\vec{c}_i) \leq \lambda_i(K^*)$ so we cannot conclude that $F_i^*(\vec{c}_i) \geq 1$. Nevertheless, we get $d(\Lambda) \leq \prod_i(\lambda_i(K^*)+1) \leq 2^n \prod_i \lambda_i(K^*)$. But $\prod_i \lambda_i(K^*) \leq 2^n/\text{Vol}(K^*)$ (Minkowski's second theorem applied to K^*) and $\text{Vol}(K^*)\text{Vol}(K) \geq 4^n/(n!)^2$ (see Theorem 14.4 in [10]). Therefore, we get $d(\Lambda) \leq (n!)^2\text{Vol}(K)$, same kind of upper bound as before. The same study can be done for a generalized reduced basis for K^* and for a Korkine-Zolotarev basis for K^* since, in this case, we also have $F_i^*(\vec{c}_i) \leq \lambda_i(K^*)$.

In practice, when the loop nest is scheduled by a multi-dimensional sequential schedule, one can consider the vectors $\vec{c}_i$ to be the vectors that define the schedule. In this case, it is often true (except for extreme cases that may not occur in practice) that all $F_i^*(\vec{c}_i)$ are equal to 0 for the first dimensions, then are larger than 1 for the remaining dimensions (indeed, as noticed in [19] with a different terminology, if the maximal lexicographic conflicting difference has level l, i.e., $l-1$ leading zeros, DS is flat for the first l dimensions, and in general contains some nonzero integer vectors for all remaining dimensions, except if the "writing" operations belong to a very skewed subspace). In this case, Heuristic 2 (as well as Heuristic 1) directly identifies the subspace in which DS lies and leads to a guaranteed performance (see Example 3). This amounts to mix the techniques of Quilleré and Rajopadhye [19] (for choosing the adequate basis), and of Lefebvre and Feautrier [15] (for choosing the modulos).

Finally, note that, as for Heuristic 1, there is a mechanism to slightly modify the mapping found by Heuristic 2 to get a valid one-dimensional mapping. Indeed, it is not difficult to see that the modular mapping $\vec{c}_1.\vec{x} + b_1(\vec{c}_2.\vec{x} + b_2(\cdots + b_{n-1}\vec{c}_n.\vec{x})) \bmod \prod_i b_i$ is also valid. In other words, this is a second (but equivalent) guaranteed mechanism for deriving valid one-dimensional modular mappings.

We conclude this section with an example that illustrates the fact that, in some cases, it is indeed important to choose an adequate basis to get performance (as for the previous heuristics). It shows that Lefebvre-Feautrier heuristic, which is always applied in the canonical basis, can be arbitrarily bad. The same example shows that considering all canonical linearizations of the original array, as suggested by De Greef, Catthoor, and De Man, is not enough.

Example 3. To make Lefebvre-Feautrier heuristic fail, it is sufficient to "turn" the body of Figure 1, i.e., to apply a change of basis so that the (new) canonical basis is inadequate. Consider a code similar to the code of Example 1, iteration (i, j) writing in array $A(i, j)$, but with the iteration domain represented on the left of Figure 3 and traversed sequentially with successive diagonals (from bottom-left to upper-right) as depicted, i.e., with multi-dimensional schedule $(i - j, i)$. If a second identical loop, traversed the same way but one clock-cycle later, uses the results of the first loop, we get the set DS represented in Figure 3, with vertices $(1, 1)$, $(-1, -1)$, $(N - 1, N)$, and $(1 - N, -N)$ (to build DS, just consider that any two successive points in the schedule of the first loop are conflicting).

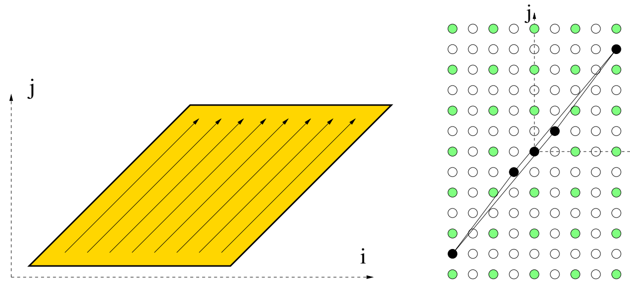

Figure 3: Iteration domain for Example 3 and corresponding set DS.

For this example, Heuristic 2 applied for the canonical basis $\{(1, 0), (0, 1)\}$ (i.e., exactly Lefebvre-Feautrier heuristic) chooses $\rho_1 = N + 1$, $\rho_2 = 1$, with a corresponding memory size of order N. Considering the basis in the opposite order does not help ($\rho_1 = N$, $\rho_2 = 1$). Similarly, the technique of De Greef *et al.* [8] only considers the mappings $Ni \pm j$ and $Nj \pm i$, and all of them have a maximal conflicting distance of order N^2, two orders of magnitude worse than the optimal! However, if we apply the proposed heuristics so as to choose an adequate basis, we will find again a memory of size $O(1)$. This is because these heuristics do not depend on the basis in which the domain DS is described, which makes their robustness. For example, following the schedule as previously suggested (an "often-quite-good" strategy), i.e., considering the "change of basis" $\{\vec{c}_1, \vec{c}_2\} = \{(1, -1), (1, 0)\}$, we indeed retrieve the set DS depicted in Figure 1 and we get the allocation $(i - j \bmod 2, i \bmod 2)$. Actually, we even saw that $\rho_1 = \rho_2 = 2$ was acceptable for this set whatever the basis.

Thus, the sublattice depicted as grey points in Figure 3 is also valid for the skewed domain of Figure 3. □

3.4 Using the LLL Basis Reduction

None of the heuristics mentioned before, for selecting a basis in which performances are always guaranteed, have a fully polynomial complexity. For completeness (and also to give the intuitive link with basis reduction), we now give a polynomial-time heuristic based on the LLL basis reduction (see for example [16, 5]). The general idea is to build an ellipsoid $\mathcal{E}$ such that $\mathcal{E}/n \subset K \subset \mathcal{E}$ (see [16]) (the factor n can be reduced to $\sqrt{n+1}$ if K is a 0-symmetric polytope), then to apply an affine transformation that transforms this ellipsoid into the unit ball (and $\mathbb{Z}^n$ into a particular lattice Λ). We now have an Euclidean norm and we can apply the LLL basis reduction to find a reduced basis for Λ. Multiplying the vectors of the basis with adequate coefficients (as for Heuristic 1) will give a sublattice of Λ, strictly admissible for the unit ball. Going back to the original body, we will have a strictly admissible integral lattice for the ellipsoid $\mathcal{E}$, thus for K. This shows the link between strictly admissible integral lattices and approximations with the LLL basis reduction. We get the following performance:

Theorem 5. *If K is a 0-symmetric bounded convex body, then one can compute in polynomial time a lattice $\Lambda \subset \mathbb{Z}^n$ such that $d(\Lambda) \leq 2^{n(n+3)/4} n^n \, Vol(K)$.*

Proof. Let Λ be a lattice in $\mathbb{R}^n$, $(\vec{b}_i)_{1\leq i \leq n}$ a basis of Λ with Gram-Schmidt orthogonalization $(\vec{b}_i^*)_{1\leq i\leq n}$. Define integers ρ_i such that $\rho_i > 1/||\vec{b}_i^*||$. It is easy to see that the sublattice Λ_1 of Λ with basis $(\rho_i \vec{b}_i)_{1\leq i\leq n}$ is strictly admissible for the unit ball $\mathcal{B}$ in $\mathbb{R}^n$, with determinant $d(\Lambda)\prod_i \rho_i$. Furthermore, if $(\vec{b}_i)_{1\leq i\leq n}$ is a LLL reduced basis for Λ and if ρ_i is as small as possible, then $\rho_i \leq 1/||\vec{b}_i^*|| + 1$, which implies $\rho_i \leq 2^{(i-1)/2}/\lambda_i(\mathcal{B},\Lambda) + 1$ because of properties of the LLL reduced basis. Assuming again that $\mathcal{B}$ contains n linearly independent points of Λ, $\lambda_i(\mathcal{B},\Lambda) \leq 1$ and we get $\rho_i \leq 2^{(i+1)/2}/\lambda_i(\mathcal{B},\Lambda)$. Since $\prod_i \lambda_i(\mathcal{B},\Lambda) \geq d(\Lambda)$, we finally get $d(\Lambda_1) \leq \prod_{i=1}^n 2^{(i+1)/2} = 2^{n(n+3)/4}$.

Now, consider a 0-symmetric bounded convex body: one can build in polynomial time an ellipsoid $\mathcal{E}$ such that $\mathcal{E}/n \subset K \subset \mathcal{E}$ (see [16]) (the factor n can be reduced to $\sqrt{n+1}$ if K is a 0-symmetric polytope). Build an affine transformation f such that $f(\mathcal{E}) = \mathcal{B}$ and apply the previous technique to $\Lambda = f(\mathbb{Z}^n)$. The lattice $\Lambda_2 = f^{-1}(\Lambda_1)$ is the desired sublattice of $\mathbb{Z}^n$, strictly admissible for $\mathcal{E}$ and thus for K. Furthermore, $d(\Lambda_2) = \mathrm{Vol}(\mathcal{E}) d(\Lambda_1)$. Putting it all together, we get $d(\Lambda_2) \leq 2^{n(n+3)/4} n^n \mathrm{Vol}(K)$. We can replace n^n by $(n+1)^{n/2}$ if K is a 0-symmetric polytope. □

4. A MORE COMPLEX CASE STUDY

We illustrate the interest of modular mappings with a more involved and fully detailed example, in 4 dimensions, similar to the DCT benchmark, but where some details are abstracted so as to make the discussion simpler. The code accesses a 4-dimensional array $A(b_r, b_c, r, c)$, in two pipelined communicating loops, the first one writing each "row" $A(b_r, b_c, r, *)$ of the array successively, the second reading each "column" $A(b_r, b_c, *, c)$ successively (see code hereafter). Here, to make things simpler, we assume that each operation of S writes all elements of a row in "parallel", i.e., at the same "macro-time" $(64\times b_r + b_c)\times 8 + r$ (in other words, the loop is scheduled sequentially as it is written), and each operation of T reads all elements of a column at macro-time $(64\times b_r + b_c)\times 8 + c + \rho$, where ρ is such that dependences are respected. (In a fully detailed implementation, S and T are formed of several "micro-statements", each one accessing 1 element of A instead of 8. These micro-statements can be software pipelined as done in [12], taking into account the available resources, in particular load-store units, possibly leading to different ρ's for each micro-statement.)

```
DO $b_r = 0$, 63
  DO $b_c = 0$, 63
    DO $r = 0$, 7
      S: $A(b_r, b_c, r, *) = \ldots$
    ENDDO
  ENDDO
ENDDO

DO $b_r = 0$, 63
  DO $b_c = 0$, 63
    DO $c = 0$, 7
      T: $\ldots = A(b_r, b_c, *, c)$
    ENDDO
  ENDDO
ENDDO
```

For all dependences to be respected, ρ must be such that $(64 \times b_r + b_c) \times 8 + c + \rho \geq (64 \times b_r + b_c) \times 8 + r + 1$ (in a more accurate model again, the delay 1 may be replaced by a larger quantity, depending on the delay for accessing the communicating buffer where the values created by the first loop are going to be stored), i.e., $\rho \geq r - c + 1$ for all $0 \leq r, c < 8$. For the rest of this case study, we pick the smallest possible value for ρ, i.e., $\rho = 8$.

We now need to decide how we want to represent the values that are going to be stored in the intermediate buffer. We can identify each operation of S by the loop indices (b_r, b_c, r) of the surrounding loops, but as each operation of S writes 8 values, we need an extra index to distinguish the different values. In other words, we identify each created value by its indices (b_r, b_c, r, c), as in the original array. As in Example 1, reasoning with loop or array indices is similar (except that we need an extra dimension for loop indices) because the array accesses are aligned with the loop indices.

We can now define the set DS, with respect to the representation (b_r, b_c, r, c), as the set of all $(\delta_{br}, \delta_{bc}, \delta_r, \delta_c)$ such that:

$$\begin{cases} \delta_{br} = b_r - b'_r,\ \delta_{cr} = c_r - c'_r,\ \delta_r = r - r',\ \delta_c = c - c' \\ 0 \leq b_r, b'_r, b_c, b'_c \leq 63,\ \ 0 \leq r, r', c, c' \leq 7, \\ 64 \times 8 \times \delta_{br} + 8 \times \delta_{bc} + r - c' \leq \rho, \\ 64 \times 8 \times \delta_{br} + 8 \times \delta_{bc} + c - r' \geq -\rho. \end{cases}$$

with $\rho = 8$. From this representation, it is clear that DS is a 0-symmetric polytope (it is even linearly parameterized by ρ, which would make a parametric derivation of memory allocations possible). To get an idea of the shape of DS, consider an element of the form $A(b_r, b_c, 0, 0)$. It is written at time $64 \times (8 \times b_r + b_c)$ and it is read 8 iterations later, thus it conflicts with all elements of the next 8 written rows. An element of the form $A(b_r, b_c, 7, 0)$, written at time $64 \times (8 \times b_r + b_c)$, is read at the next iteration and thus conflicts only with the elements of the next written row. The next written row(s) can be written by S at the same iterations of the b_r and b_c loops (i.e., for $\delta_{br} = \delta_{bc} = 0$), but can

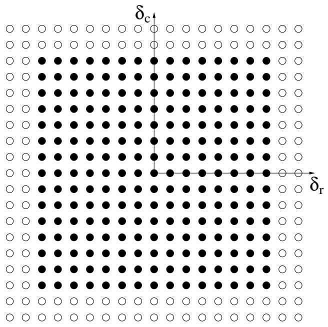

Figure 4: DS part for $(\delta_{br}, \delta_{bc}) = (0, 0)$.

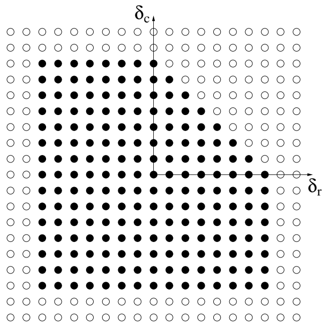

Figure 5: DS part for $(\delta_{br}, \delta_{bc}) = (0, 1)$ and $(\delta_{br}, \delta_{bc}) = (1, -63)$.

also be written at the next iteration of the b_c loop (and same iteration of the b_c loop), or, extreme case (when $b_c = 63$), at the next iteration of the b_r loop and the very first iteration of the b_c loop (i.e., for $\delta_{br} = 1$ and $\delta_{bc} = -63$). The set DS is 4-dimensional but, for clarity, it is better represented as the union of five 2-dimensional parts. The central part, which corresponds to the particular values $\delta_{br} = \delta_{bc} = 0$, is depicted in Figure 4. It is the square of all $(0, 0, \delta_r, \delta_c)$, where $-7 \leq \delta_r, \delta_c \leq 7$; it corresponds to a memory of 8 full rows, i.e., 64 elements. The set depicted in Figure 5 corresponds to two parts, the set of all (δ_r, δ_c) for $\delta_{br} = 0$ and $\delta_{bc} = 1$, and for $\delta_{br} = 1$, $\delta_{bc} = -63$. Its symmetric with respect to 0 is the set of all (δ_r, δ_c) for $\delta_{br} = 0$ and $\delta_{bc} = -1$, and for $\delta_{br} = -1$, $\delta_{bc} = 63$. These five pieces form the whole set DS. To get yet a better view of the set DS, consider again the set depicted in Figure 1. It is also a representation of the projection of DS onto the first two components δ_{br} and δ_{bc}. In other words, for the two first indices, the situation is similar to Example 1.

We can now derive modular allocations. If we apply Heuristic 2 in the canonical basis, we find successively $\rho_{br} = 2$ and $\rho_{bc} = 2$ (as for Example 1), then we need to consider the set of Figure 4 and we get $\rho_r = 8$, and finally $\rho_c = 8$. This leads to the mapping $(b_r \bmod 2, b_c \bmod 2, r \bmod 8, c \bmod 8)$ with a memory size equal to 256. This is exactly what Lefebvre and Feautrier would find since, here, we picked the basis given by the schedule, which is also the basis of the original code. If the original code was written with the b_c loop outside the b_r loop, but scheduled the same way with the b_r loop outside, then Lefebvre and Feautrier would consider the index b_c first and they would find successively $\rho_{bc} = 64$, $\rho_{br} = 1$, $\rho_r = 8$, and finally $\rho_c = 8$, for a memory size equal to 4096! Again, we see that the choice of basis, and the order in which we evaluate the F_i^*'s for Heuristic 2 (in the opposite order of the F_i's for Heuristic 1) is important.

Following our linearization mechanism for Heuristic 2, we can find a linear allocation with the same memory size than the mapping $(b_r \bmod 2, b_c \bmod 2, r \bmod 8, c \bmod 8)$ we just found; this is the mapping $b_r + 2b_c + 4r + 32c \bmod 256$.

As for Example 1, we also see that we could follow the schedule more closely, with the index $t = 64b_r + b_c$ (i.e., coalescing the two outer loops) so as to try to find a modular allocation with a memory size equal to 2 instead of 4 for the two outermost indices. If we redefine DS with the indices (t, r, c) instead of (b_r, b_c, r, c), we get a 3-dimensional space with three 2-dimensional parts, the central part of Figure 4 for $t = 0$, the part of Figure 5 for $t = 1$, and its symmetric with respect to 0 for $t = -1$. Since DS is now a 3-dimensional space instead of a 4-dimensional space, we are more likely to gain a factor 2 (as in the worse case of the heuristics). Indeed, we find successively $\rho_t = 2$, $\rho_r = 8$, and $\rho_c = 8$, and the mapping $(t \bmod 2, r \bmod 8, c \bmod 8)$, or equivalently $(b_c \bmod 2, r \bmod 8, c \bmod 8)$, with memory size 128, half of what we found before. The linearized version is $b_c + 2r + 16c \bmod 128$. Here, we could also consider the indices in the opposite order, with the same result, $\rho_c = 8$, $\rho_r = 8$, and $\rho_t = 2$. Then, the linearized version is $c + 8r + 64t \bmod 128$, which is a particular linearization of the original array, modulo 128, solution that De Greef *et al.* would have found in this particular case (again, by "luck" because the schedule is aligned with the array accesses).

Actually, the maximal distance between conflicting indices, in the linearized representation $c + 8r + 64b_c + 4096b_r$, is not 127 but 120, therefore De Greef *et al.* would even find the mapping $c + 8r + 64b_c + 4096b_r \bmod 121$, or equivalently $c + 8r + 64b_c + 103b_r \bmod 121$. Can we do better? To answer this question, we can search, as suggested in Section 3, for the smallest (in determinant) strictly admissible integral lattice for DS, generating triangular matrices for the basis of the lattices we try. The numbers involved here are small enough to make this search practical. We find that the best modular allocation has a memory size equal to 112, for the (unique) lattice generated by the vectors $(1, 0, 0, 12), (0, 1, 0, 12), (0, 1, 4, 20), (0, 0, 0, 28)$, which corresponds (among all allocations with same associated lattice) to the allocation $(r \bmod 4, 16(b_r + b_c) + 2r + c \bmod 28)$, a two-dimensional allocation (with no corresponding linearized version). (Note that $16t = 16(64b_r + b_c) = 16(b_r + b_c) \bmod 28$.) The best one-dimensional allocation needs one more memory cell, with memory size 113, for example, the allocation $60b_r + 8b_c + 42r + c \bmod 113$, i.e., $8t + 42r + c \bmod 113$, or the more "natural" allocation $64t + 8r + 3c \bmod 113$.

5. CONCLUSION

In this paper, we proposed a mathematical framework to study modular memory allocations. We showed the link between valid modular allocations and strictly admissible lattices for the set DS of differences of conflicting indices, and the link between the optimal achievable memory size and the volume of DS, in the case of a 0-symmetric polytope. We explored several heuristic mechanisms that allows us to derive optimal modular allocations, up to a multiplicative factor that depends on the problem dimension only. Several important questions remain open, both from theoretical and practical viewpoints.

How much do we loose when restricting to lattices whose Smith form have all diagonal components equal to 1, except the last one? These correspond to 1-dimensional mappings, which are important in practice. We only showed that restricting to them is optimal up to a multiplicative factor.

Can we derive better heuristics in general? With some practical hypotheses? Can we better exploit the fact that DS may not be any 0-symmetric convex body K, but comes from a scheduled program? We already mentioned that, under reasonable assumptions, an "often-quite-good" strategy in the case of a sequential multi-dimensional affine schedule is to mix the two techniques developed in [15] and [19], i.e., to build the modulos as Lefebvre-Feautrier do, but in the basis defined by the schedule as done by Quilleré-Rajopadhye.

In which cases can we prove that modular allocations (whose optimal performances, as we have showed, are linked to the volume of DS when DS is a 0-symmetric polytope) cannot be arbitrarily bad compared to MAXLIVE? In general, how bad can be the optimal modular allocations compared to MAXLIVE? Actually, to reach MAXLIVE, we may need to use polynomials [6] but, of course, the corresponding addressing functions will be much more complicated, the hardware more costly, and may not be acceptable in practice.

A complete study remains to be done to better understand the impact of the different approximations used to build DS and the link with dependences and schedules. In particular, it is also important to be able to handle not only polytopes, but union of polytopes. This paper is just the first step to address all these questions: we only focused on the mathematical framework and its general properties. Future work needs to address more specific theoretical questions as well as strategies for practical implementations.

[6]With reasonable hypotheses, one can indeed show that Ehrhart polynomials, combined with some hardware mechanisms (as in [1]), can be used to derive allocations with a corresponding memory size equal to MAXLIVE, but this goes out of the scope of this paper.

6. REFERENCES

[1] S. Aditya and M. S. Schlansker. Shiftq: A buffered interconnect for custom loop accelerators. In *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES'01)*, pages 158–167, Atlanta, USA, 2001. ACM Press.

[2] A. Darte, M. Dion, and Y. Robert. A characterization of one-to-one modular mappings. *Parallel Processing Letters*, 5(1):145–157, 1996.

[3] A. Darte, R. Schreiber, B. R. Rau, and F. Vivien. Constructing and exploiting linear schedules with prescribed parallelism. *ACM Transactions on Design Automation of Electronic Systems*, 7(1):159–172, 2002.

[4] A. Darte, F. Vivien, and Y. Robert. *Scheduling and Automatic Parallelization.* Birkhäuser, Boston, 2000.

[5] C. Dwork. Lattices and their application to cryptography. Available as lecture notes `http://theory.stanford.edu/~csilvers/cs359/`, 1998.

[6] F. Catthoor. *et al.* Atomium: A toolbox for optimising memory I/O using geometrical models. `http://www.imec.be/design/atomium/`.

[7] P. Feautrier. Data flow analysis of scalar and array references. *International Journal of Parallel Programming*, 20(1):23–53, 1991.

[8] E. De Greef, F. Catthoor, and H. De Man. Memory size reduction through storage order optimization for embedded parallel multimedia applications. *Parallel Computing*, 23:1811–1837, 1997.

[9] P. M. Gruber. Geometry of numbers. In P. Gruber and J. Wills, editors, *Handbook of Convex Geometry*, volume B, chapter 3.1, pages 739–763. Elsevier Science Publishers B.V., 1993.

[10] P. M. Gruber and C. G. Lekkerkerker. *Geometry of Numbers.* North Holland, second edition, 1987.

[11] L. Hafer. The generalized basis reduction algorithm (annotated). `http://www.cs.sfu.ca/~lou/MITACS/grb.pdf`, June 2000.

[12] V. Kathail, S. Aditya, R. Schreiber, B. R. Rau, D. C. Cronquist, and M. Sivaraman. PICO: Automatically designing custom computers. *IEEE Computer*, 35(9):39–47, Sept. 2002.

[13] B. Kienhuis, E. Rijpkema, and E. F. Deprettere. Compaan: Deriving process networks from Matlab for embedded signal processing architectures. In *8th International Workshop on Hardware/Software Codesign (CODES'00)*, San Diego, May 2000.

[14] J. C. Lagarias. Point lattices. In R. Graham, M. Grötschel, and L. Lovász, editors, *Handbook of Combinatorics*, volume I, chapter 19, pages 919–966. Elsevier Science Publishers B.V., 1995.

[15] V. Lefebvre and P. Feautrier. Automatic storage management for parallel programs. *Parallel Computing*, 24:649–671, 1998.

[16] L. Lovász. *An Algorithmic Theory of Numbers, Graphs, and Convexity*, volume 50 of *CBMS-NSF Regional Conference Series in Applied Mathematics.* SIAM, 1986.

[17] L. Lovász and H. E. Scarf. The generalized basis reduction algorithm. *Mathematics of Operations Research*, 17(3):751–764, 1992.

[18] M. Newman. *Integral Matrices.* Academic Press, 1972.

[19] F. Quilleré and S. Rajopadhye. Optimizing memory usage in the polyhedral model. *ACM Transactions on Programming Languages and Systems*, 22(5):773–815, 2000.

[20] P. Quinton, S. Rajopadhye, T. Risset, *et al.* Alpha homepage: A language dedicated to the synthesis of regular architectures. `http://www.irisa.fr/cosi/`.

[21] M. M. Strout, L. Carter, J. Ferrante, and B. Simon. Schedule-independent storage mapping for loops. In *8th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'98)*, pages 24–33, San Jose, USA, 1998. ACM Press.

[22] W. Thies, F. Vivien, J. Sheldon, and S. Amarasinghe. A unified framework for schedule and storage optimization. In *International Conference on Programming Language Design and Implementation (PLDI'01)*, pages 232–242. ACM Press, 2001.

[23] A. Turjan and B. Kienhuis. Storage management in process networks using the lexicographically maximal preimage. In *14th International Conference on Application-specific Systems, Architectures and Processors (ASAP'03)*, The Hague, The Netherlands, June 2003. IEEE Computer Society Press.