CART: Constant Aspect Ratio Tiling[*]

Guillaume Iooss
ENS Lyon, UCBL, Univ. Lyon
CNRS, Inria
Colorado State University
guillaume.iooss[at]ens-lyon.fr

Sanjay Rajopadhye
Colorado State University
sanjay.rajopadhye[at]colostate.edu

Christophe Alias
ENS Lyon, UCBL, Univ. Lyon
CNRS, Inria
christophe.alias[at]ens-lyon.fr

Yun Zou
Colorado State University
yunzou.colostate[at]gmail.com

ABSTRACT

Parametric tiling is a well-known transformation which is widely used to improve locality, parallelism and granularity. However, parametric tiling is also a non-linear transformation and this prevents polyhedral analysis or further polyhedral transformation after parametric tiling. It is therefore generally applied during the code generation phase.

In this paper, we present a method to remain polyhedral, in a special case of parametric tiling, where all the dimensions are tiled and all the tile sizes are constant multiples of a single tile size parameter. We call this *Constant Aspect Ratio Tiling*. We show how to mathematically transform a polyhedron and an affine function into their tiled counterpart, which are the two main operations needed in such transformation.

1. INTRODUCTION

Tiling is a very important transformation with several benefits, including locality improvement and parallelism with a granularity which can be different from that in the original, untiled program. Parametric tiling—when tile sizes are symbolic parameters, unknown at compile time—is however, not polyhedral: indeed, tiling a dimension i with a parametric tile size b ends up replacing i by two indices α and ii satisfying $i = \alpha b + ii$, and $0 \leq ii < b$, which is a quadratic expression. Because of this, we can no longer apply polyhedral transformations (such as skewing) or do any further polyhedral analysis after a parametric tiling transformation. Thus, this transformation is usually managed during the code generation step of a compiler. In addition to preventing further polyhedral analyses and/or transformations a major drawback of this is that many decisions which are, properly speaking, in the purview of the analysis-transformation phase of compilation, are relegated to the code generator. For example, the Dtile code generator in the AlpahZ system hard wires the decision to execute a tiled program with wavefront parallelism with wavefronts normal to the vector $\vec{1}$. In order to change this decision, the code generator has to be modified! Of course, it is also possible to use non-parametric tiling, which is polyhedral but fixes the tile sizes at compile-time, and comes with is own set of drawbacks.

[*]Supported in part by NSF grants CCF-0917319 and CNS-1240991, and by AFOSR grant FA9550-13-1-0064

IMPACT 2014
Fourth International Workshop on Polyhedral Compilation Techniques
Jan 20, 2014, Vienna, Austria
In conjunction with HiPEAC 2014.

http://impact.gforge.inria.fr/impact2014

In this paper, we focus on a special case of parametric tiling, called *Constant Aspect Ratio Tiling* (CART), where all the dimensions are tiled, and tile sizes are a constant multiple of the same tile size parameter. We show that, under these hypotheses, it is possible to apply parametric tiling *within* the polyhedral world, but with a different set of parameters.

More precisely, our contributions are the following:

- We show how to transform a polyhedron into a union of tiled polyhedra along canonical directions. After finding a first expression of this union, we improve it to coalesce polyhedra, such that we have at most one polyhedron per tile.
- We show how to transform an affine function into a tiled piecewise affine function along canonical directions. We show that, in the general case, the conditions of the tiled piecewise affine function are $\mathbb{Z}$-polyhedral and present a method to obtain this expression. We also present a necessary and sufficient condition under which the pieces are polyhedral.
- Finally, we show how to extend this theory: we first show how to tile along any directions, and then we present a sufficient condition to reuse the same reasoning in the case of several size parameters, and/or when only a subset of dimensions are tiled.

The two transformations described above are fundamental in order to apply the CART transformation on programs. For example, in a loop nest, we will need to tile the iteration domain and the access functions. In an Alpha program, we will need to tile the expression domains and the dependence functions.

In the rest of this paper, we will first study the case of a polyhedron in Section 2, then of an affine function in Section 3. The extensions of this theory are presented in Section 4. In Section 5 we examine the literature, before concluding in Section 6.

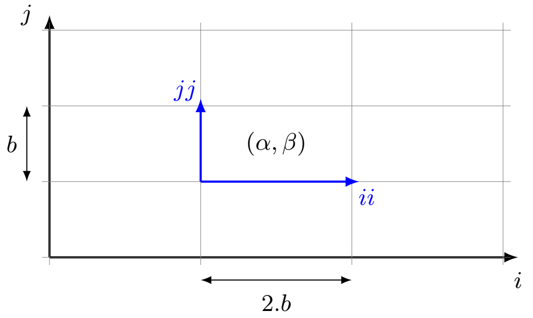
2. CART ON POLYHEDRAL SETS

In this section, we show how to transform a given polyhedron $\mathcal{D}$ into an equivalent tiled union of polyhedra Δ, in the rectangular tiling case where the tile sizes of all the dimensions are multiples of the *same* parameter.

2.1 A First Expression of Δ

Given a polyhedron $\mathcal{D} = \{\vec{i} \mid Q.\vec{i} + Q^{(p)}.\vec{p} + \vec{q} \geq \vec{0}\}$, where $\vec{p}$ are the parameters, we introduce the following notation:

- Each tile size is a multiple of the same *block size parameter* b. Thus, we have: $\vec{i} = b.D.\vec{\alpha} + \vec{ii}$, where:
  - D is called the *scale*, and is a diagonal matrix of strictly positive integer constants.
  - $\vec{\alpha}$ are called the *block number indices*, $\vec{ii}$ are called the *local indices* and we have $\vec{0} \leq \vec{ii} < b.D.\vec{1}$.
- We assume that all parameters $\vec{p}$ can be decomposed in the same fashion: $\vec{p} = b.\vec{\lambda} + \vec{pp}$ where $\vec{\lambda}$ is the vector of *tiled parameters*, $\vec{pp}$ the *local parameters* and $\vec{0} \leq \vec{pp} < b.\vec{1}$.



Example of rectangular CART in the 2D case

We want to obtain $\Delta = \{\ \vec{\alpha}, \vec{ii} \mid \dots\}$ such that Δ is simply an alternate representation of $\mathcal{D}$, using the new indices $(\vec{\alpha}, \vec{ii})$ and the new parameters, $(\vec{\lambda}, \vec{pp})$. To obtain the new constraints of Δ, let us start from the constraints of $\mathcal{D}$:

$$Q.\vec{i} + \vec{q} + Q^{(p)}.\vec{p} \geq \vec{0}$$

We use the definitions of $\vec{\alpha}$, $\vec{ii}$, $\vec{\lambda}$, $\vec{pp}$ (namely, $\vec{i} = b.D.\vec{\alpha} + \vec{ii}$ and $\vec{p} = b.\vec{\lambda} + \vec{pp}$) to get rid of $\vec{i}$ and $\vec{p}$:

$$b.Q.D.\vec{\alpha} + Q.\vec{ii} + b.Q^{(p)}.\vec{\lambda} + Q^{(p)}.\vec{pp} + \vec{q} \geq \vec{0}$$

These constraints are *no longer* polyhedral (b is a parameter and $\vec{\alpha}$ are indices). To get rid of the quadratic part, let us divide both sides by the tile size parameter b (which is known to be strictly positive):

$$Q.D.\vec{\alpha} + Q^{(p)}.\vec{\lambda} + \frac{Q.\vec{ii} + Q^{(p)}.\vec{pp} + \vec{q}}{b} \geq \vec{0}$$

In general, the fraction is a rational vector. Thus, to come back into the integer world, let us take the floor of the previous constraints (because $a \geq 0 \Leftrightarrow \lfloor a \rfloor \geq 0$):

$$Q.D.\vec{\alpha} + Q^{(p)}.\vec{\lambda} + \left\lfloor \frac{Q.\vec{ii} + Q^{(p)}.\vec{pp} + \vec{q}}{b} \right\rfloor \geq \vec{0}$$

Let $\vec{k}(\vec{ii}) = \left\lfloor \frac{Q.\vec{ii}+Q^{(p)}.\vec{pp}+\vec{q}}{b} \right\rfloor$. We will show that $\vec{k}(\vec{ii})$ can only take a *constant* number of different values (because $\vec{0} \leq \vec{ii} < b.D.\vec{1}$ and $\vec{0} \leq \vec{pp} < b.\vec{1}$). Formally, we do this by bounding each dimension of $\vec{k}(\vec{ii})$. Consider, for $0 \leq l < \dim(\vec{k}(\vec{ii}))$,

$$k_l(\vec{ii}) = \left\lfloor \frac{Q_l.\vec{ii} + Q_l^{(p)}.\vec{pp} + q_l}{b} \right\rfloor$$

Let us look at its maximum. Because we have an affine function in the numerator, this maximum is reached when all the coordinates of $\vec{ii}$ are set to either 0 (if the corresponding coefficient of Q_l is negative) or $d.(b-1)$ (if the corresponding coefficient of Q_l is strictly positive and where d is the corresponding coefficient in D). Following the notion of the outset, as introduced by Renganaryana et al. [7]., let QD_l^+ be the vector of non-negative coefficients of $Q_l.D$ and $Q_l^{(p)+}$ the vector of non-negative coefficients of $Q_l^{(p)}$. Also, let $||\vec{v}||_1 = \sum_i |v_i|$ denote the L1-norm of the vector $\vec{v}$. Then, we can get an upper bound on $k_l^{\max}$ as follows.

$$\begin{aligned}
k_l^{\max} &= \max_{\vec{ii}} \left\lfloor \tfrac{Q_l.\vec{ii}+Q_l^{(p)}.\vec{pp}+q_l}{b} \right\rfloor \\
&= \left\lfloor \tfrac{||QD_l^+||_1.(b-1)+Q_l^{(p)}.\vec{pp}+q_l}{b} \right\rfloor \\
&= ||QD_l^+||_1 + \left\lfloor \tfrac{Q_l^{(p)}.\vec{pp}-||QD_l^+||_1+q_l}{b} \right\rfloor \\
&\leq ||QD_l^+||_1 + \left\lfloor \tfrac{||Q_l^{(p)+}||_1.(b-1)-||QD_l^+||_1+q_l}{b} \right\rfloor \\
&\leq ||QD_l^+||_1 + ||Q_l^{(p)+}||_1 + \left\lfloor \tfrac{q_l-||Q_l^{(p)+}||_1-||QD_l^+||_1}{b} \right\rfloor
\end{aligned}$$

We thus have a constant upper bound on all elements of $\vec{k}$. We can prove a similar, constant lower bound on the elements of $\vec{k}(\vec{ii})$, and hence there are only a *constant* number of different values it can take, $\vec{k}(\vec{ii}) \in [\![\vec{k}^{\min}; \vec{k}^{\max}]\!] = \{\vec{k}_1, \dots, \vec{k}_A\}$.

Hence, we manage to express Δ as a finite union of parametric polyhedra:

Proposition 2.1. *The blocked version of a polyhedron* $\mathcal{D} = \{\vec{i} \mid Q.\vec{i} + Q^{(p)}.\vec{p} + \vec{q} \geq \vec{0}\}$ *is:*

$$\Delta = \bigcup_{a=1}^{A} \left\{ \vec{\alpha}, \vec{ii} \mid \begin{array}{c} Q.D.\vec{\alpha} + Q^{(p)}.\vec{\lambda} + \vec{k}_a \geq \vec{0} \\ b.\vec{k}_a \leq Q.\vec{ii} + Q^{(p)}.\vec{pp} + \vec{q} < b.(\vec{k}_a + \vec{1}) \\ \vec{0} \leq \vec{ii} < b.D.\vec{1} \end{array} \right\}$$

where $\vec{k}_a$ is one value of $\left\lfloor \frac{Q.\vec{ii}+Q^{(p)}.\vec{pp}+\vec{q}}{b} \right\rfloor \in [|\vec{k}^{\min}; \vec{k}^{\max}|]$.

Let us now study the values taken by $\vec{k}^{\max}$. In general, b can take any non-negative value, including $b = 1$. Thus, if the fraction is positive, the highest value of $k_l^{\max}$ is reached at $b = 1$ and is $k_l^{\max} = q_l$. If the fraction is negative, the highest value of k_l^{max} is reached for b "large enough" to make the floor equal to -1 and is $k_l^{\max} = ||QD_l^+||_1 + ||Q_l^{(p)+}||_1 - 1$.

In practice, it is likely that the block size of a tile should be large enough (to get the locality benefits). Thus, if we assume that b is large enough, the floor is equal to either 0 or -1 (depending on the sign of the fraction), and we can get a more precise value of b when the fraction is positive. The same reasoning also applies to the computation of the lower bound $\vec{k}^{min}$.

2.2 Example

Let us consider the following parameterized triangle:

$$\mathcal{D} = \{i, j \mid N - 1 - i - j \geq 0 \wedge i \geq 0 \wedge j \geq 0\}$$

Let us introduce $\begin{pmatrix} i \\ j \end{pmatrix} = b.\begin{pmatrix} \alpha \\ \beta \end{pmatrix} + \begin{pmatrix} ii \\ jj \end{pmatrix}$ and, to simplify the result, let us assume that the parameter N is a multiple of the block size parameter b: $N = M.b$. Then, the first inequality becomes:

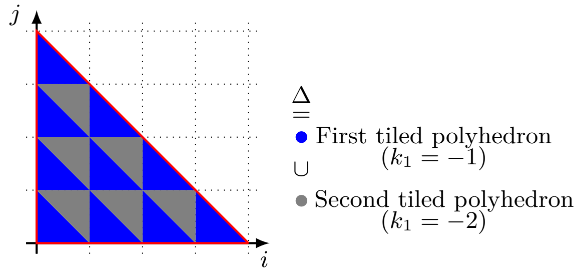
$$\begin{aligned} N - 1 - i - j \geq 0 \quad &\Leftrightarrow \quad M.b - 1 - b.\alpha - ii - b.\beta - jj \geq 0 \\ &\Leftrightarrow \quad M - \alpha - \beta + \left\lfloor \frac{-ii-jj-1}{b} \right\rfloor \geq 0 \end{aligned}$$

Let us study the values of $k_1(ii, jj) = \left\lfloor \frac{-ii-jj-1}{b} \right\rfloor$. Because of the sign of the numerator coefficients, the maximum is -1 ($ii = jj = 0$) and the minimum is -2 ($ii = jj = b-1$). After analyzing the two other inequalities, we obtain:

$$\Delta = \bigcup_{\substack{-2 \leq k_1 \leq -1 \\ k_2 = 0 \\ k_3 = 0}} \left\{ \alpha, \beta, ii, jj \;\middle|\; \begin{array}{c} M - \alpha - \beta + k_1 \geq 0 \\ b.k_1 \leq -ii - jj - 1 \\ -ii - jj - 1 < b.(k_1 + 1) \\ 0 \leq ii, jj < b \\ \alpha + k_2 \geq 0 \\ \beta + k_3 \geq 0 \end{array} \right\}$$

$$= \left\{ \alpha, \beta, ii, jj \;\middle|\; \begin{array}{c} M - \alpha - \beta - 1 \geq 0 \\ \alpha, \beta \geq 0 \\ 0 \leq ii, jj < b \\ -b \leq -ii - jj - 1 < 0 \end{array} \right\}$$

$$\cup \left\{ \alpha, \beta, ii, jj \;\middle|\; \begin{array}{c} M - \alpha - \beta - 2 \geq 0 \\ \alpha, \beta \geq 0 \\ 0 \leq ii, jj < b \\ -2b \leq -ii - jj - 1 < -b \end{array} \right\}$$


j
i
Δ
=
● First tiled polyhedron ($k_1 = -1$)
∪
● Second tiled polyhedron ($k_1 = -2$)

Obtained union of tiled polyhedra Δ

The tiling previously found is correct, but we notice that most tiles are split in two triangles, corresponding to the contribution of the two polyhedra to the tile. We can also notice some regularity in this decomposition: if a block is partially covered by a gray triangle, then the same block will be completed by a blue triangle. By using this regularity, it is possible to reorganize the union of polyhedra forming Δ to distinguish between the diagonal triangular tiles and the full tiles. We describe the method to fuse the tiles together in the following subsection.

2.3 Fusing polyhedra for a better tiling

We managed to transform $\mathcal{D}$ into a union of tiled polyhedra Δ. However, these polyhedra are mostly small triangles or trapezoids, which can be combined together to form a whole tile. To reduce the number of polyhedra (which will allow simpler generated code), we want to reorganize these polyhedra such that, for each tile $\vec{\alpha}$, there is at most one polyhedron from the union Δ contributing to this tile.

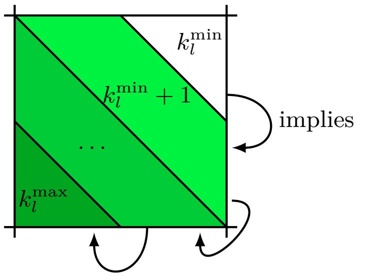
Let us focus on a constraint of $\mathcal{D}$ ($Q_l.\vec{i} + Q_l^{(p)}.\vec{p} + q_l \geq 0$). For the lth constraint, we obtained in Proposition 2.1:

$$\bigcup_{k_l^{\min} \leq k_l \leq k_l^{\max}} \left\{ \begin{array}{l} (Block_{k_l}) : Q_l.D.\vec{\alpha} + Q_l^{(p)}.\vec{\lambda} + k_l \geq 0 \\ (Local_{k_l}) : b.k_l \leq Q_l.\vec{ii} + Q_l^{(p)}.\vec{pp} + q_l \\ \qquad\qquad\qquad\qquad < b.(k_l + 1) \\ \vec{0} \leq \vec{ii} < b.D.\vec{1} \end{array} \right.$$

We can notice some properties among these constraints:

- Each k_l covers a different stripe of the tile (whose equations is given by $(Local_{k_l})$). The union of all these stripes, for $k_l^{\min} \leq k_l \leq k_l^{\max}$ covers the whole tile (by definition of $k_l^{\min}$ and $k_l^{\max}$).

- If a tile $\vec{\alpha}$ satisfies the constraint $(Block_{k_l})$ for a given k_l, then the same tile also satisfies $(Block_{k'_l})$ for every $k'_l > k_l$. In other words, if the k_lth stripe in a tile is non-empty, the tile will have all the k'_l stripes, for every $k'_l > k_l$.



Stripe coverage inside a given tile

Thus, if a block $\vec{\alpha}$ satisfies $(Block_{k_l^{\min}})$, the whole tile is covered by Δ. Also, if a block satisfies exactly $(Block_{k_l})$ (i.e. if $Q_l.D.\vec{\alpha} + Q_l^{(p)}.\vec{\lambda} + k_l = 0$), then we do not have the stripes below k_l and only the local indices ii which satisfy $(b.k_l \leq Q_l.\vec{ii} + Q_l^{(p)}.\vec{pp} + q_l)$ are covered by Δ. Using these observations, we can separate the tiles into two categories: those which satisfy $(Block_{k_l^{\min}})$ (corresponding to a full tile), and those which satisfy exactly a $(Block_{k_l})$ where $k_l^{\min} < k_l$ (corresponding to a portion of the tile).

By using these observations, we can reorganize the polyhedra of Δ in the following way:

$$\Delta = \bigcap_l \left[\begin{array}{c} \displaystyle\bigcup_{k_l^{\min} < k_l \leq k_l^{\max}} \left\{ \vec{\alpha}, \vec{ii} \;\middle|\; \begin{array}{c} Q_l.D\vec{\alpha} + Q_l^{(p)}.\vec{\lambda} + k_l = 0 \\ b.k_l \leq Q_l.\vec{ii} + Q_l^{(p)}.\vec{pp} + q_l \\ \vec{0} \leq \vec{ii} < b.D.\vec{1} \end{array} \right\} \\ \\ \cup \left\{ \vec{\alpha}, \vec{ii} \;\middle|\; \begin{array}{c} Q_l.D\vec{\alpha} + Q_l^{(p)}.\vec{\lambda} + k_l^{\min} \geq 0 \\ \vec{0} \leq \vec{ii} < b.D.\vec{1} \end{array} \right\} \end{array} \right]$$

Example.

Let us go back to the example we have previously developed. With this new expression of Δ, we obtain the following union of polyhedra:

$$\Delta \;=\; \left\{ \alpha,\beta,ii,jj \;\middle|\; \begin{array}{c} M-\alpha-\beta-1=0 \\ \alpha,\beta \geq 0 \\ 0 \leq ii, jj < b \\ -b \leq -ii-jj-1 \end{array} \right\} \cup \left\{ \alpha,\beta,ii,jj \;\middle|\; \begin{array}{c} M-\alpha-\beta-2 \geq 0 \\ \alpha,\beta \geq 0 \\ 0 \leq ii, jj < b \end{array} \right\}$$

The first polyhedron covers the lower triangles of the diagonal tiles. The second polyhedron covers the full-tiles. Thus, we only have a single polyhedron per tile (α, β).

3. CART ON AFFINE FUNCTIONS

In this section, we show how to transform a given affine function f into an equivalent tiled piecewise affine function ϕ, in the rectangular tiling case where the tile size of all the dimensions are multiples of the same parameter.

3.1 Piecewise affine function with polyhedral conditions

Given an affine function $f : (\vec{i} \mapsto Q.\vec{i} + Q^{(p)}.\vec{p} + \vec{q})$, we introduce the following notation:

- Each tile size is a multiple of the same *block size parameter* b. Thus, we have, for the *input indices*: $\vec{i} = b.D.\vec{\alpha} + \vec{ii}$, where:
  - $\vec{\alpha}$ are called the *block number indices*, $\vec{ii}$ are called the *local indices* and we have $\vec{0} \leq \vec{ii} < b.D.\vec{1}$.
  - D is called the *scale*, and is a diagonal matrix of strictly positive integer coefficients.
- Likewise, for the *output indices*: $\vec{i}' = b.D'.\vec{\alpha}' + \vec{ii}'$ with similar assumptions.
- We assume that all parameters $\vec{p}$ can be decomposed in the same fashion: $\vec{p} = b.\vec{\lambda} + \vec{pp}$ where $\vec{\lambda}$ is called the *tiled parameters*, $\vec{pp}$ the *local parameters* and $\vec{0} \leq \vec{pp} < b.\vec{1}$.

We want to transform the affine function f into a function $\phi : (\vec{\alpha}, \vec{ii} \mapsto \ldots,\ \ldots)$ which operates on the tiled space and such that $f(\vec{i}) = \vec{i}' \;\Leftrightarrow\; \phi(\vec{\alpha}, \vec{ii}) = (\vec{\alpha}', \vec{ii}')$. To obtain the expressions of $\vec{\alpha}'$ as a function of $\vec{\alpha}$ and $\vec{ii}$, we start from the definition of f:

$$\vec{i}' = Q.\vec{i} + Q^{(p)}.\vec{p} + \vec{q}$$

By substituting $\vec{i}$ and $\vec{i}'$ by their tiled counterparts, respectively, $\vec{\alpha}$, $\vec{ii}$ and $\vec{\beta}$, $\vec{jj}$, we obtain:

$$b.D'.\vec{\alpha}' + \vec{ii}' = Q.(b.D.\vec{\alpha} + \vec{ii}) + Q^{(p)}.(b.\vec{\lambda} + \vec{pp}) + \vec{q}$$

After dividing by b, we obtain:

$$D'.\vec{\alpha}' + \frac{\vec{ii}'}{b} = Q.D.\vec{\alpha} + Q^{(p)}.\vec{\lambda} + \frac{Q.\vec{ii} + Q^{(p)}.\vec{pp} + \vec{q}}{b}$$

Then, we multiply both sides by D'^{-1}:

$$\vec{\alpha}' + \frac{D'^{-1}.\vec{ii}'}{b} = \begin{array}{c} D'^{-1}.Q.D.\vec{\alpha} + D'^{-1}.Q^{(p)}.\vec{\lambda} \\ + \frac{D'^{-1}.(Q.\vec{ii} + Q^{(p)}.\vec{pp} + \vec{q})}{b} \end{array}$$

Because $\vec{0} \leq \vec{ii}' < b.D'.\vec{1}$, we can eliminate $\vec{ii}'$ by taking the floor:

$$\vec{\alpha}' = \left\lfloor D'^{-1}.Q.D.\vec{\alpha} + D'^{-1}.Q^{(p)}.\vec{\lambda} + \frac{D'^{-1}.(Q.\vec{ii} + Q^{(p)}.\vec{pp} + \vec{q})}{b} \right\rfloor$$

However, we have no guaranty that in general, $(D'^{-1}.Q.D.\vec{\alpha})$ and $(D'^{-1}.Q^{(p)}.\vec{\lambda})$ are integral vectors. To allow us to draw these terms outside the floor operator, we will assume that $(D'^{-1}.Q.D)$ and $(D'^{-1}.Q^{(p)})$ are integer matrices. We will show later that these two hypotheses form a necessary and sufficient condition to have only polyhedral conditions in the piecewise affine function ϕ. We obtain:

$$\vec{\alpha}' = D'^{-1}.Q.D.\vec{\alpha} + D'^{-1}.Q^{(p)}.\vec{\lambda} + \left\lfloor \frac{D'^{-1}.(Q.\vec{ii} + Q^{(p)}.\vec{pp} + \vec{q})}{b} \right\rfloor$$

By defining $\vec{k}(\vec{ii}) = \left\lfloor \frac{D'^{-1}.(Q.\vec{ii} + Q^{(p)}.\vec{pp} + \vec{q})}{b} \right\rfloor$ and by conducting the same kind of analysis as in Section 2, we manage to bound $\vec{k}(\vec{ii})$ between $\vec{k}^{\min}$ and $\vec{k}^{\max}$. Finally, we obtain a piecewise expression of $\vec{\alpha}'$, in which each branch corresponds to a value of $\vec{k}(\vec{ii})$, and which is:

$$\begin{array}{l} \vec{\alpha}' = D'^{-1}.Q.D.\vec{\alpha} + D'^{-1}.Q^{(p)}.\vec{\lambda} + \vec{k}_a \\ \quad \text{if} \quad b.\vec{k}_a \leq D'^{-1}.Q.\vec{ii} + D'^{-1}.Q^{(p)}.\vec{pp} + D'^{-1}.\vec{q} < b.(\vec{k}_a + \vec{1}) \end{array}$$

for each $\vec{k}_a \in [|\vec{k}^{\min}; \vec{k}^{\max}|] = \{\vec{k}_1, \ldots, \vec{k}_A\}$.

We can easily compute $\vec{ii}'$ for each obtained branch by using the definition of $\vec{\alpha}'$. Finally, we obtain the following expression of ϕ as a piecewise affine function:

Proposition 3.1. *The blocked version of an affine function $f(\vec{i}) = Q.\vec{i} + Q^{(p)}.\vec{p} + \vec{q}$ is a piecewise affine function, whose branches are:*

$$\begin{array}{l} \phi(\vec{\alpha}, \vec{ii}) = \begin{pmatrix} D'^{-1}.Q.D.\vec{\alpha} + D'^{-1}.Q^{(p)}.\vec{\lambda} + \vec{k}_a \\ Q.\vec{ii} + Q^{(p)}.\vec{pp} + \vec{q} - b.D'.\vec{k}_a \end{pmatrix} \\ \quad \textit{if} \quad b.\vec{k}_a \leq D'^{-1}.Q.\vec{ii} + D'^{-1}.Q^{(p)}.\vec{pp} + D'^{-1}.\vec{q} < b.(\vec{k}_a + \vec{1}) \end{array}$$

for each $\vec{k}_a \in [|\vec{k}^{\min}; \vec{k}^{\max}|]$.

3.2 Example

Let us consider the following affine function:

$$f : (i, j \mapsto 2i, N - j - 1, i + j)$$

Let us introduce $i = 2.\alpha.b + ii$, $j = 2.\beta.b + jj$ and let us assume that $N = b.M$ to reduce the number of branches of the piecewise affine function ϕ. Moreover, if $f(i,j) = (i', j', k')$, let us introduce $i' = \alpha'.b + ii'$, $j' = \beta'.b + jj'$ and $k' = 2.\gamma'.b + kk'$. The conditions to have only polyhedral conditions in ϕ are satisfied:

$$D'^{-1}.Q.D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix}^{-1} . \begin{pmatrix} 2 & 0 \\ 0 & -1 \\ 1 & 1 \end{pmatrix} . \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = \begin{pmatrix} 4 & 0 \\ 0 & -2 \\ 1 & 1 \end{pmatrix} \text{ which is integral.}$$

$$D'^{-1}.Q^{(p)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix}^{-1} . \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \text{ which is integer.}$$

$$\phi\begin{pmatrix}\alpha\\ \beta\\ ii\\ jj\end{pmatrix} = \begin{cases} (\ 4\alpha, M-2\beta-1, \alpha+\beta, & 2ii, b-jj-1, ii+jj\ )^T \quad \text{if} \quad 0\leq ii<b \wedge 0\leq jj<b \wedge 0\leq ii+jj<2b \\ (\ 4\alpha+1, M-2\beta-1, \alpha+\beta, & 2ii-b, b-jj-1, ii+jj\ )^T \quad \text{if} \quad b\leq ii<2b \wedge 0\leq jj<b \wedge 0\leq ii+jj<2b \\ (\ 4\alpha, M-2\beta-2, \alpha+\beta, & 2ii, 2b-jj-1, ii+jj\ )^T \quad \text{if} \quad 0\leq ii<b \wedge b\leq jj<2b \wedge 0\leq ii+jj<2b \\ (\ 4\alpha, M-2\beta-2, \alpha+\beta+1, & 2ii, 2b-jj-1, ii+jj-2bb\ )^T \quad \text{if} \quad 0\leq ii<b \wedge b\leq jj<2b \wedge 2b\leq ii+jj<4b \\ (\ 4\alpha+1, M-2\beta-1, \alpha+\beta+1, & 2ii-b, b-jj-1, ii+jj-2bb\ )^T \quad \text{if} \quad b\leq ii<2b \wedge 0\leq jj<b \wedge 2b\leq ii+jj<4b \\ (\ 4\alpha+1, M-2\beta-2, \alpha+\beta+1, & 2ii-b, 2b-jj-1, ii+jj-2bb\ )^T \quad \text{if} \quad b\leq ii<2b \wedge b\leq jj<2b \wedge 2b\leq ii+jj<4b \end{cases}$$

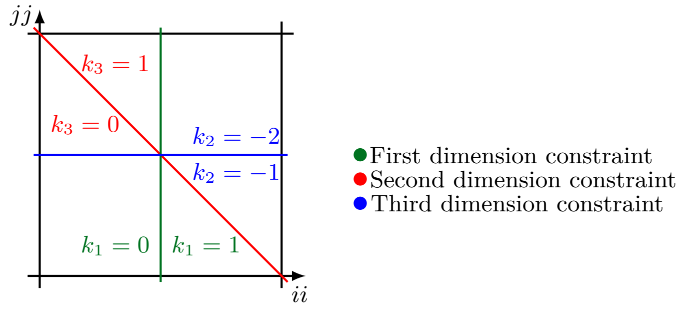
Figure 1: Value of the tiled affine function in the example of Section 3.2

Therefore, after doing the operations described in the previous subsection, we obtain an expression of $\vec{\alpha'}$:

$$\begin{bmatrix}\alpha'\\ \beta'\\ \gamma'\end{bmatrix} = \begin{pmatrix}4 & 0\\ 0 & -2\\ 1 & 1\end{pmatrix}\begin{bmatrix}\alpha\\ \beta\end{bmatrix} + \begin{pmatrix}0\\ 1\\ 0\end{pmatrix}.\begin{bmatrix}M\end{bmatrix} + \begin{bmatrix}k_1\\ k_2\\ k_3\end{bmatrix}$$

where $k_1 = \left\lfloor \frac{2ii}{b} \right\rfloor$, $k_2 = \left\lfloor \frac{-jj-1}{b} \right\rfloor$ and $k_3 = \left\lfloor \frac{ii+jj}{2b} \right\rfloor$. Thus, $0 \leq k_1 \leq 1$, $-2 \leq k_2 \leq -1$ and $0 \leq k_3 \leq 1$.

Two out of the resulting eight branches have unsatisfiable conditions. Therefore, after pruning them out, we obtain the expression of ϕ described in Figure 1. The domains of each branch inside a given tile are described in the following figure:



jj
ii
$k_3 = 1$
$k_3 = 0$
$k_2 = -2$
$k_2 = -1$
$k_1 = 0$
$k_1 = 1$
● First dimension constraint
● Second dimension constraint
● Third dimension constraint

Domains of each branch in Example 3.2

3.3 General case: piecewise affine function with Z-polyhedral conditions

In Section 3.1, we presented a condition to have only polyhedral constraints in the tiled piecewise affine function ϕ. We will now study the general case, and show that the constraints of ϕ are $\mathbb{Z}$-polyhedral in general. We obtained the following equality:

$$\vec{\alpha}' = \left\lfloor D'^{-1}.Q.D.\vec{\alpha} + D'^{-1}.Q^{(p)}.\vec{\lambda} + \frac{D'^{-1}.(Q.\vec{ii} + Q^{(p)}.\vec{pp} + \vec{q})}{b} \right\rfloor$$

Let us consider a given dimension $0 \leq l < |\vec{\alpha}'|$:

$$\alpha'_l = \left\lfloor \frac{Q_l.D.\vec{\alpha}}{D'_{l,l}} + \frac{Q^{(p)}_l.\vec{\lambda}}{D'_{l,l}} + \frac{Q_l.\vec{ii} + Q^{(p)}_l.\vec{pp} + q_l}{D'_{l,l}.b} \right\rfloor$$

The vector $\frac{Q_l.D}{D'_{l,l}}$ can have rational coordinates. This means that the block indices $\vec{\alpha}$ and parameters $\vec{\lambda}$ may contribute to the expression of the local indices. To quantify this contribution, we introduce $\vec{\alpha} = \vec{\mu}^l.D'_{l,l} + \vec{\alpha\alpha}^l$ and $\vec{\lambda} = \vec{\theta}^l.D'_{l,l} + \vec{\lambda\lambda}^l$ where $\vec{0} \leq \vec{\alpha\alpha}^l < D'_{l,l}.\vec{1}$ and $\vec{0} \leq \vec{\lambda\lambda}^l < D'_{l,l}.\vec{1}$.

The previous equality becomes:

$$\alpha'_l = Q_l.D.\vec{\mu}^l + Q^{(p)}_l.\vec{\theta}^l + \left\lfloor \frac{Q_l.D.\vec{\alpha\alpha}^l + Q^{(p)}_l.\vec{\lambda\lambda}^l}{D'_{l,l}} + \frac{Q_l.\vec{ii} + Q^{(p)}_l.\vec{pp} + q_l}{D'_{l,l}.b} \right\rfloor$$

If we observe the floor term, we notice that its right part corresponds to the expression of $\vec{k}$ we have obtained in Section 3.1. Thus, if we are able to fix its left part by assuming that $\vec{\alpha\alpha}^l$ and $\vec{\lambda\lambda}^l$ are constants, we can conduct exactly the same analysis seen before. This can be done by enumerating all the possible values of $\vec{\alpha\alpha}^l$ af$\vec{\lambda\lambda}^l$ and by creating one branch per value. Because $\vec{0} \leq \vec{\alpha\alpha}^l < D'_{l,l}.\vec{1}$ (resp. $\vec{\lambda\lambda}^l$), we have only a constant number of such values. Therefore, if we enumerate all the values of $\vec{\alpha\alpha}^l$ (i.e., if we create one branch of ϕ per value of $\vec{\alpha\alpha}^l$), we can compute the expression of ϕ as seen in the previous subsection.

Because $\vec{\alpha\alpha}^l = \vec{\alpha} \bmod D_{l,l}$ (by definition), a condition on the value of $\vec{\alpha\alpha}^l$ is a condition on the congruency of $\vec{\alpha}$. Thus, we obtain two kinds of conditions in the branches of ϕ: the congruency conditions on $\vec{\alpha}$ and the polyhedral conditions (coming from the analysis of $\vec{k}$). Therefore, a given branch of the piecewise affine function will be selected if $\vec{\alpha}$ belongs to an affine lattice (corresponding to the congruency conditions) and if $(\vec{\alpha}, \vec{ii})$ belongs to a polyhedron. Because a $\mathbb{Z}$-polyhedron is the intersection of an affine lattice and a polyhedron, we conclude that the conditions are $\mathbb{Z}$-polyhedral.

Let us estimate the number of branches of the tiled affine function we can obtain in the worst case. For the lth dimension, the range of values $\vec{k}$ can take is only slightly shifted compared to each other (the shift being the fractional part of $\frac{Q_l.D.\vec{\alpha}}{D'_{l,l}}$). Thus, we have almost the same number $Card(\vec{k})_l$ of $\vec{k}$ for any $\vec{\alpha\alpha}^l$. Moreover, we have one branch per values of $(\vec{k}, \vec{\alpha\alpha}^l, \vec{\lambda\lambda}^l)$, if $\frac{Q_l.D.\vec{\alpha}}{D'_{l,l}}$ and $\frac{Q^{(p)}_l.\vec{\lambda}}{D'_{l,l}}$ are both not integer. Thus, in the worst case where none of these fractions are integer for every dimension l (in short, nothing divides nicely anything), the number of branches of the resulting blocked affine function is about:

$$\prod_l \left(Card(\vec{k})_l \times \mathrm{Card}\{\alpha\alpha^l, \vec{\lambda\lambda^l}\} \right) = \prod_l O(q_l + ||(Q.D)_l||_1 + ||Q^{(p)}_l||_1) \times D'^{\,dim(\vec{\alpha}+\vec{\lambda})}_{l,l}$$

Example.

Let us consider $f : (i,j \mapsto i,j)$ where the input indices are tiled as $\begin{pmatrix}i\\ j\end{pmatrix} = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}.b + \begin{pmatrix}ii\\ jj\end{pmatrix}$ and the output indices are tiled as $\begin{pmatrix}i'\\ j'\end{pmatrix} = \begin{pmatrix}2\alpha'\\ 3\beta'\end{pmatrix}.b + \begin{pmatrix}ii'\\ jj'\end{pmatrix}$. Let us consider the

$$\phi : \begin{bmatrix}\alpha\\ \beta\\ ii\\ jj\end{bmatrix} \mapsto \begin{cases} (\ \alpha/2, \beta/3, & ii, jj \ )^T & \text{if } \alpha \equiv 0 \bmod 2 \wedge \beta \equiv 0 \bmod 3 \\ (\ \alpha/2, (\beta-1)/3, & ii, jj+b \ )^T & \text{if } \alpha \equiv 0 \bmod 2 \wedge \beta \equiv 1 \bmod 3 \\ (\ \alpha/2, (\beta-2)/3, & ii, jj+2b \ )^T & \text{if } \alpha \equiv 0 \bmod 2 \wedge \beta \equiv 2 \bmod 3 \\ (\ (\alpha-1)/2, \beta/3, & ii+b, jj \ )^T & \text{if } \alpha \equiv 1 \bmod 2 \wedge \beta \equiv 0 \bmod 3 \\ (\ (\alpha-1)/2, (\beta-1)/3, & ii+b, jj+b \ )^T & \text{if } \alpha \equiv 1 \bmod 2 \wedge \beta \equiv 1 \bmod 3 \\ (\ (\alpha-1)/2, (\beta-2)/3, & ii+b, jj+2b \ )^T & \text{if } \alpha \equiv 1 \bmod 2 \wedge \beta \equiv 2 \bmod 3 \end{cases}$$

Figure 2: Value of the tiled affine function in the example of Section 3.3

first output dimension:

$$\begin{aligned} i' = i \quad &\Leftrightarrow \quad 2.\alpha'.b + ii' = \alpha.b + ii \\ &\Rightarrow \quad \alpha' = \left\lfloor \tfrac{\alpha}{2} + \tfrac{ii}{2b} \right\rfloor = \mu^{(1)} + \left\lfloor \tfrac{\alpha\alpha^{(1)}}{2} + \tfrac{ii}{2b} \right\rfloor \end{aligned}$$

where $\alpha = 2.\mu^{(1)} + \alpha\alpha^{(1)}$ and $0 \leq \alpha\alpha^{(1)} \leq 1$. Likewise, we have:

$$\beta' = \nu^{(2)} + \left\lfloor \frac{\beta\beta^{(2)}}{3} + \frac{jj}{3b} \right\rfloor$$

where $\beta = 3.\nu^{(2)} + \beta\beta^{(2)}$ and $0 \leq \beta\beta^{(2)} \leq 2$. Finally, we can build the pieces of ϕ by enumerating all the possible values of $\alpha\alpha^{(1)}$ and $\beta\beta^{(2)}$. For example, for $\alpha\alpha^{(1)} = \beta\beta^{(2)} = 0$:

$$\vec{k}(ii, jj) = \begin{pmatrix} \frac{ii}{2b} & \frac{jj}{3b} \end{pmatrix}^T$$

We only have one possible value for $k_1(ii, jj)$ and $k_2(ii, jj)$ (which is 0 in both cases), thus we will only have one branch in ϕ corresponding to these values. The full obtained expression of ϕ is given in Figure 2.

3.4 Condition to have polyhedral conditions

In Section 3.1, we introduced a condition such that the resulting piecewise affine function ϕ admits only polyhedral constraints. Let us show informally that this condition is necessary (we already showed in Section 3.1 that it is sufficient).

We obtained the following equation in Section 3.1:

$$\vec{\alpha}' + \frac{D'^{-1}.\vec{ii}'}{b} = \begin{array}{l} D'^{-1}.Q.D.\vec{\alpha} + D'^{-1}.Q^{(p)}.\vec{\lambda} \\ + \frac{D'^{-1}.(Q.\vec{ii} + Q^{(p)}.\vec{pp} + \vec{q})}{b} \end{array}$$

Let us assume that $D'^{-1}.Q.D$ is not integer, and let us show that ϕ admits congruency conditions. In the previous equation, because $\vec{\alpha}'$ is integer, if we look at the fractional part, the value of $\vec{\alpha}$ has an impact on $D'^{-1}.\vec{ii}'/b$, thus on $\vec{ii}'$. However, as we saw in the Section 3.3, for a given branch of ϕ, the value of the piecewise affine function corresponding to $\vec{ii}'$ is independent of $\vec{\alpha}$. Thus, the only impact $\vec{\alpha}$ could have on $\vec{ii}'$ is through the conditions of the branches of ϕ. Because the only conditions using α' are congruency conditions, ϕ must have such conditions. Therefore, ϕ admits non-polyhedral conditions.

Likewise, we can prove that if $D'^{-1}.Q^{(p)}$ is not integer, ϕ also admits some non-polyhedral conditions (by looking at the influence of $\vec{\lambda}$ on $\vec{ii}'$). Therefore, we conclude:

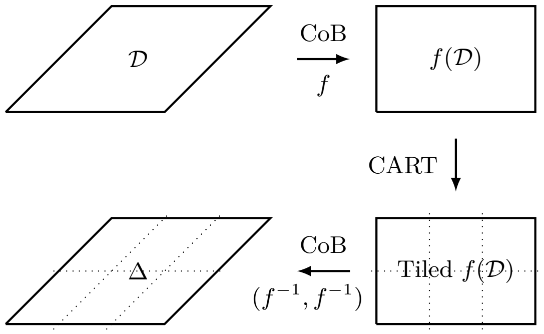
PROPOSITION 3.2. *The function ϕ has only polyhedral conditions if and only if $D'^{-1}.Q.D$ and $D'^{-1}.Q^{(p)}$ are integral.*

4. EXTENSIONS

In the previous sections, we presented a way to parametrically tile a polyhedron and an affine function, under several hypothesis. These hypothesis were the following: (i) all the dimensions must be tiled (ii) these dimensions must be the canonical dimensions and (iii) all the tile sizes must be integer multiples of the same tile size parameter b. In this section, we will first present a way to get rid of hypothesis (ii), then we will show how to weaken hypothesis (i) and (iii), in particular to manage several tile size parameters.

4.1 CART along non-canonic axis

In the previous sections, we forced the tiling to be along the canonical axis (because of the definition of the block and local indices: $\vec{i} = b.D.\vec{\alpha} + \vec{ii}$). This restriction can be lifted by using an appropriately chosen Change of Basis (CoB) transformation. If we want to tile along some arbitrary hyperplanes, we do a preprocessing unimodular transformation, f, to render the normal vectors of these hyperplanes as unit vectors. Then, we apply the Constant Aspect Ratio Tiling to obtain a tiled polyhedron. To return to the original tile space, we need to reverse the initial CoB, which can be done by another CoB of bijection (f^{-1}, f^{-1}).



4.2 Managing several tile size parameters

In the previous sections, we assumed that all the dimensions were tiled and that all the tile sizes are multiples of a single tile size parameter b. Let us see how far we can go by relaxing these hypotheses while still being able to apply the same mathematical method.

To explain the intuition, let us first remove the latter hypotheses: we assume that all the dimensions admit a different tile size parameter $b_1, \ldots, b_n$. Thus, $\vec{i} = B.\vec{\alpha} + \vec{ii}$ where $B = diag(b_1, \ldots, b_n)$. Given a polyhedron $\mathcal{D} = \{\vec{i} \mid Q.\vec{i} + Q^{(p)}.\vec{p} + \vec{q} \geq \vec{0}\ \}$, if we apply the method we presented, we obtain:

$$\begin{aligned} & Q.(B.\vec{\alpha} + \vec{ii}) + Q^{(p)}.\vec{p} + \vec{q} \geq 0 \\ \Leftrightarrow \quad & (Q.B).\vec{\alpha} + Q.\vec{ii} + \vec{q} + Q_p.\vec{p} \geq 0 \end{aligned}$$

We want to divide each row of these inequalities by a tile size parameter b_i to get rid of the quadratic form $Q.B.\alpha$. However, if several tile sizes appear in the same line, we will end up with fractions of tile size parameters $\frac{b_i}{b_j}$. We cannot manage these fraction simply statically. So, to prevent such fractions, we restrict ourselves to tilings where only a single tile size parameter can appear in any considered affine expression (constraint from a polyhedron or value of an affine function). Intuitively, we can see this restriction as a cartesian product between tilings with different tile size parameters.

We observe the same kind of phenomenon if we try to remove the first hypotheses: if a non-tiled index appears in the same equation as a tiled index, then we cannot divide the equation by a tile size parameter to get rid of the quadratic form.

In the case of multiple tile size parameters, this relaxed hypotheses means that we have a partition of the indices, according to their tile size parameter. Moreover, if two indices appear in the same affine expression, then these two indices must have the same tile size parameter. Thus, if we consider the matrix Q, modulo a permutation over its columns (to group the indices according to their tile size parameter) it must be a diagonal block matrix (each diagonal block corresponds to one of a set of tile size parameters). For the parameters, we can introduce a new blocked (resp. local) parameter $\lambda_{i,j}$ (resp. $pp_{i,j}$) such that $\vec{p}_j = b_i.\lambda_{i,j} + pp_{i,j}$, if the parameter $\vec{p}_j$ appears in an affine expression being divided by the tile size parameter b_i. Such a polyhedron is a cartesian product of single-parameter polyhedra as studied in the previous sections.

5. RELATED WORK

Several techniques exist in the polyhedral model community to generate parametric tiled code. The earliest ones are based on a symbolic version of Fourier-Motzkin elimination (cf. Appendix B of the work of Renganaryana et al. [7]), or tile the bounding box of the iteration domain [9] to obtain a rectangular (and simpler) problem.

Lakshminarayanan et al. [7, 6] describe an improvement of the bounding box technique which avoids iterating over empty tiles. The main idea is to compute the set of non-empty tile coordinates (called *outset*) and the set of full tile coordinates (called *inset*) to improve the efficiency of code generation. Kim and Rajopadhye [5] extended this to imperfectly nested loops and developed the *DTile* tool. However, in these methods, the associated sets are non-polyhedral, and the generated code has quadratic expressions. In our (restricted) case, we can easily obtain these two sets from the expression of the tiled union of polyhedra (for any tile form, we can even get the set of tiles which admit this form).

Tavarageri et al. [8] present a study of three recent parametric tiling code generation techniques: *PrimeTile* [3], *DynTile* [4] and *PTile* [1]. *PrimeTile* decomposes the iteration domain into stripes, and places inside these stripes as many full-tile as possible, while taking care of the extra part not fitting inside a full tile through a prelude and a postlude. However, because the tile origins are not guaranteed to form a lattice, it is not trivial to parallelize across tiles. *DynTile* is an improvement of this method where the tile origins are constrained to be on a lattice, and were we can exploit parallelism across tiles according to a predetermined wavefront pattern. However, this method is also based on a dynamic schedule, given by an inspector which determines the set of tiles belonging to a given wavefront. Finally, *Ptile* is similar to the Dtile approach: the equation of the outset is computed by relaxing the constraints of the considered polyhedron. This set is also non-polyhedral, and they also have quadratic expressions inside their generated code.

As opposed to parametric tiling, the non-parametric tiling transformation is a polyhedral transformation. Thus, to generate tiled code, we can use any polyhedral code generator (based on tools such as CLooG [2]). Likewise, because we obtain a (disjoint) union of polyhedra at the end of the CART transformation, we can use such a code generator.

6. CONCLUSION

We have proposed a method to tile polyhedra and affine functions, in the case where we tile all the dimensions and we have only one tile size parameter. In the case of a polyhedron, we have shown how to obtain a mathematical expression for the union of polyhedra corresponding to the blocked polyhedron. Then, we have reorganized this union to coalesce the tiled union of polyhedra such that we have only one contributing polyhedron per tile. In the case of an affine functions, we have shown that a tiled affine function is a piece-wise affine function admitting, in the general case, $\mathbb{Z}$-polyhedral conditions on its branches. Then, we have presented a necessary and sufficient condition to have only polyhedral constraints. Finally, we have presented some extensions of this work, to tile along any given directions and to manage, under some restricted conditions, several tile size parameters.

We have partially implemented our algorithms in Java.[1] In the case of a polyhedron, we have implemented the expression of Δ described in Section 2.3, and in the case of an affine function, we have implemented the polyhedral case (Section 3.1). In all methods, we assumed that the block size parameter is "large enough", such that we can compute precisely $\vec{k}_{min}$ and $\vec{k}_{max}$ (Section 2.2).

Polyhedral domains and affine functions are the two fundamental mathematical objects which support polyhedral programs. Thus, it should be possible to apply the CART transformation on loop nests to obtain a polyhedral loop nest at the end of the transformation (modulo a parametric part to link the old parameters with the new tiled parameters). In our compiler, *AlphaZ* [10], polyhedral domains and affine functions appear explicitly, thus the CART transformation can be easily applied.

To get a full parametric tiling transformation, we need, of course, to ensure the legality of tiling. Moreover, in the case of the *Alpha* language, we also have to infer the scale D for intermediate sets of indices appearing inside an expression (between two affine functions). It might be interesting to find out which scale leads to the best performance after code generation.

In the future, we plan to use this transformation as the first step of the *semantic tiling* transformation. In short, this is a tiling which transform a program reasoning on scalars into a program reasoning on blocks of data, by using algebraic properties (such as associativity and commutativity) to change the computation itself (and to modify some dependences). In our current approach to formalize such a transformation, we are planning to isolate the computation

[1]In the Gecos repository (`svn://scm.gforge.inria.fr/svnroot/gecos/trunk`), the Java class is `org.polymodel.polyhedralIR.transformation.ParametricTiling.java`

touching a given block of data (through a parametric tiling), then to compare each computation block with a blocked operation candidate (using an equivalence algorithm, modulo associativity and commutativity).

7. REFERENCES

[1] M. M. Baskaran, A. Hartono, S. Tavarageri, T. Henretty, J. Ramanujam, and P. Sadayappan. Parameterized tiling revisited. In *Proceedings of the 8th Annual IEEE/ACM International Symposium on Code Generation and Optimization*, CGO '10, pages 200–209, New York, NY, USA, 2010. ACM.

[2] C. Bastoul. Code generation in the polyhedral model is easier than you think. In *International Conference on Parallel Architecture and Compilation Techniques*, pages 7–16, Juan-les-Pins, France, September 2004.

[3] A. Hartono, M. M. Baskaran, C. Bastoul, A. Cohen, S. Krishnamoorthy, B. Norris, J. Ramanujam, and P. Sadayappan. Parametric multi-level tiling of imperfectly nested loops. In *Proceedings of the 23rd International Conference on Supercomputing*, ICS '09, pages 147–157, New York, NY, USA, 2009. ACM.

[4] A. Hartono, M. M. Baskaran, J. Ramanujam, and P. Sadayappan. Dyntile: Parametric tiled loop generation for parallel execution on multicore processors. In *IEEE International Symposium on Parallel and Distributed Processing*, IDPDS'10, 2010.

[5] D. Kim and S. Rajopadhye. Efficient Tiled Loop Generation: D-Tiling. In G. Gao, L. Pollock, J. Cavazos, and X. Li, editors, *Languages and Compilers for Parallel Computing*, volume 5898 of *Lecture Notes in Computer Science*, pages 293–307. Springer Berlin Heidelberg, 2010.

[6] L. Renganarayanan, D. Kim, S. Rajopadhye, and M. M. Strout. Parameterized tiled loops for free. In *Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation*, PLDI '07, pages 405–414, New York, NY, USA, 2007. ACM.

[7] L. Renganarayanan, D. Kim, M. M. Strout, and S. Rajopadhye. Parameterized loop tiling. *ACM Trans. Program. Lang. Syst.*, 34(1):3:1–3:41, May 2012.

[8] S. Tavarageri, A. Hartono, M. Baskaran, L. noël Pouchet, and J. Ramanujam. Parametric tiling of affine loop nests.

[9] J. Xue. *Loop tiling for parallelism.* Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[10] T. Yuki, G. Gupta, D. Kim, T. Pathan, and S. Rajopadhye. AlphaZ: A System for Design Space Exploration in the Polyhedral Model. In *Proceedings of the 25th International Workshop on Languages and Compilers for Parallel Computing*, 2012. For more detail, see http://www.cs.colostate.edu/TechReports/Reports/2012/tr12-101.pdf.