Optimization of Number of Processors in VLSI Arrays

Yannick Saouter

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Optimization of Number of Processors in VLSI Arrays

Yannick Saouter

N° 2333

Septembre 1994

PROGRAMME 1

ISSN 0249-6399

INRIA
RENNES

Optimization of Number of Processors in VLSI Arrays

Yannick Saouter

Programme 1 — Architectures parallèles, bases de données, réseaux et systèmes distribués
Projet API

Rapport de recherche n° 2333 — Septembre 1994 — 27 pages

Abstract: Since the work of Kung[1], the systolic architectures have proven their efficiency to deal with many scientific algorithms (LU-decomposition, Gauss-Jordan elimination, ...). Since the early eighties, many works have been made in the area of automatic derivation of systolic architectures. In the general case, there are numerous solutions to the same problem. The final choice of the architecture is often done by comparison of performances. There are several criteria which can be considered: global time of computation, number of processors, latency of the circuit ... In this article we are especially interested with the number of processors of the final architecture for which we present a heuristic method.

Key-words: Automatic derivation of parallel architectures;Non-linear Optimization; Hypervolume.

(Résumé : tsvp)

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 84 71 71

Optimisation du nombre de processeurs dans les réseaux VLSI

Résumé : Depuis les travaux de Kung[1], les architures systoliques ont prouvé leur efficacité à traiter de nombreux algorithmes scientifiques (décomposition LU, élimination de Gauss-Jordan, ...). Depuis le début des années 80, beaucoup de travaux ont été fait dans le domaine de la dérivation automatique d'architectures systoliques. Dans le cas général,, il existe de nombreuses solutions pour un même problème. Le choix final de l'architecture est souvent fait par comparaison de performances. Il y a plusieurs critères qui peuvent être considérés: temps total de calcul, nombre de processeurs, latence du circuit ... Dans cet article, nous nous intéressons plus particulièrement au nombre de processeurs de l'architecture finale pour lequel nous présentons une méthode heuristique.

Mots-clé : Dérivation automatique d'architectures parallèles; Optimisation non-linéaire; Hypervolume.

1 Introduction

Systolic architectures are defined by means of regularity and modularity [1] and allow to obtain high performances in terms of data flow rate. Since the early eighties many researchers have been developping methods to derive automatically dedicated systolic architectures [2, 3, 4, 5, 6]. Most of these methods are based on the same concept of recurrent systems. Those methods have the advantage of ensuring the exactness of the architecture derived. But there is always a great freedom let to the designer and many architectures may be derived for the same algorithm. The performances (global time of computation, number of processors, electronical complexity ...) of the architectures may differ and so the designer may look for an optimal architecture for a given criterion. The problem of optimization of criteria has already been studied by various authors [7, 8] and often leads to problems of non-linear optimization quite hard to process. In this article we are especially interested in the number of processors of the target architecture and present a new method whose complexity is small in pratical cases and independant of the size of the domain of iteration of the algorithm. This new method, which is heuristic, also illustrates the interest of the substitution of criteria for some cases.

2 Definitions

We will suppose that basic notions such as dot product are known by the reader and we will introduce some definitions that will be needed.

2.1 Lattices

The theory of lattices is widely exposed in [9]. We present here some basic properties which will be useful in the following.

Definition 2.1 (poset) *Let E be a non-empty set and let $\preceq$ be a binary relation defined over E. $(E, \preceq)$ will be said to be a* poset *if the relation $\prec$ is:*

- *Reflexive: $\forall x \in E,\ x \preceq x$;*
- *Anti-symmetrical: If $x \preceq y$ and $y \preceq x$ then $x = y$;*
- *Transitive: If $x \preceq y$ and $y \preceq z$ then $x \preceq z$.*

Given such a relation, it is possible to define the greatest lower bound of two elements of a poset:

Definition 2.2 (greatest lower bound) *Let E be a poset and x, y two elements of E. We note $L_{x,y} = \{t \in E \mid t \preceq x, t \preceq y\}$. Let us suppose that there exists $z \in L_{x,y}$ such that $\forall t \in L_{x,y},\ t \preceq z$. Then z is unique, is called* greatest lower bound *of x and y, and is denoted $x \wedge y$.*

Proof Let z_1 and z_2 be two greatest lower bounds of x and y. Then necessarily $z_1 \preceq z_2$ and $z_2 \preceq z_1$ by definition. So by antisymmetricality we have $z_1 = z_2$. □

Reciprocally we define lowest upper bounds:

Definition 2.3 (lowest upper bound) *Let E be a poset and x, y two elements of E. We note $U_{x,y} = \{t \in E \mid x \preceq t, y \preceq t\}$. Let us suppose that there exists $z \in U_{x,y}$ such that $\forall t \in U_{x,y}$, $z \preceq t$. Then z is unique, is called* lowest upper bound *of x and y, and is denoted $x \vee y$.*

Proof Same as above. □

From the definition of the greatest lower bounds and lowest upper bounds, one deduce immediately that the operations $\vee$ and $\wedge$ are commutative and associative when defined.

Then a lattice is defined as follows:

Definition 2.4 (lattice) *A lattice is a poset $(E, \preceq)$ for which any two elements x and y have a greatest lower bound and a lowest upper bound.*

If we suppose the lattice to have a finite number of elements, then we have additionnal properties:

Proposition 2.1 *Let E be a finite lattice. Then there exists an* absolute minimum $\bot$ *and an* absolute maximum $\top$ *in E such that $\forall x \in E$, $\bot \preceq x \preceq \top$.*

Proof The reader may verify $\bot = \bigwedge_{x \in E} x$ and $\top = \bigvee_{x \in E} x$. □
Then we define:

Definition 2.5 (maximal proper element) *Let E be a finite lattice whose absolute maximum is $\top$. Then an element x of E will be said to be a* maximal proper element *if we have $x \neq \top$, and $\{y \in E \mid x \preceq y\} = \{x, \top\}$.*

Reciprocally we could have introduced the notion of minimal proper element. At last we define the relationship of adjacency which is very useful in the domain of polyhedral theory:

Definition 2.6 (adjacency) *Let E be a finite lattice. Then $x, y \in E$ are said to be* adjacent *if and only if the equation $t \wedge u = x \wedge y$ implies $x \preceq t$ and $y \preceq u$ or $x \preceq u$ and $y \preceq t$.*

2.2 Convex polyhedra

In this section we define the basic notions of the theory of polyhedra and linear programming. The reader is referred to [10] for more details and for the demonstrations of the results presented here.

Definition 2.7 (convex polyhedron) *Let E be an affine space of dimension n over $\mathbf{R}$. A* polyhedron *is a set D such that:*

$$z \in D \iff A.z \geq B$$

where A is an $n \times m$ matrix and B is an n-vector. Moreover, every row $a.z \geq b$ of the system $A.z \geq B$ is said to be a constraint *of the domain D.*

It is equivalent to say that a polyhedron is the intersection of a finite number of half-spaces, each of these half-spaces corresponding to a constraint.

Definition 2.8 (face) *Let E be an affine space of dimension n over $\mathbf{R}$, let c be a vector of $\mathbf{R}^n$ and let D be a convex polyhedron included in E. We note $F_c = \{z \in D \mid \forall t \in D, c.t \leq c.z\}$. If F_c is not empty it is said to be a* face *of D.*

A face is then defined as the set of points which maximizes a linear form. A special case of face is the vertex case:

Definition 2.9 (vertex) *A face of a polyhedron which is of dimension 0 (i.e. reduced to a single point) is called a* vertex.

Definition 2.10 (lattice of face) *The set of face of a polyhedron D, completed by the empty set, has a canonical structure of lattice. The order is the inclusion, the infimum composition is the intersection of the faces, while the supremum composition is the convex hull of the union of the faces. This lattice is called the* lattice of the faces.

One can note here that any polyhedron D is itself a face. Indeed we have $D = F_0$. It is obvious that D is the maximal face of the lattice.

Definition 2.11 (facet) *A* facet *of a polyhedron D is a maximal proper face of D (that is to say a face which is maximal for the order of the lattice of the faces of D and distinct of D).*

The notion of dimension enables us to make a hierarchy between faces; This notion is, in fact, defined for any polyhedron:

Definition 2.12 (dimension of a polyhedron) *Let D be a polyhedron of an affine space E of finite dimension over $\mathbf{R}$. The* dimension *of D is the dimension of the vectorial space sustaining the smallest affine space containing D. Moreover D will be said to be* full-dimensionnal *if its dimension is equal to the dimension of E.*

This means that a polyhedron is full-dimensionnal if it contains at least $n + 1$ affinely independent points. In the following of the article, we will deal with full-dimensionnal polyhedral domains. Under this assumption we have additionnal results:

Proposition 2.2 *Let D be a full-dimensionnal polyhedron of an affine space E of dimension n. Then every facet is the intersection between D and an hyperplane. Moreover the equation of this hyperplane, say $a.z = b$ belongs as an inequality to the set of the constraints of D (i.e. either $a.z \geq b$ or $-a.z \geq b$ is a row of the system $A.z \geq B$ which defines D).*

Another useful notion is the concept of rays.

Definition 2.13 (ray) *Let D be a non-empty polyhedron. A* ray *is a vector r, such that $\forall z \in D$, $\forall \lambda \in \mathbf{R}^+$, $z + \lambda r \in D$.*

It is easy to see that any positive linear combination of rays is still a ray. The converse is false, so we define extremal rays:

Definition 2.14 (extremal ray) *A ray which cannot be obtained by positive combination of other rays of the domain is said to be an* extremal ray.

At last there is one more type of rays:

Definition 2.15 (line of a domain) *A ray r such that $-r$ is also a ray is called a* line *or also a* bidirectionnal ray *of the domain.*

As for the rays, we define extremal lines. In the following, we will not treat the lines separately but we will consider that they are present as opposite pairs in the set of the extremal rays.

Definition 2.16 (pointed polyhedron) *Let D be a non-empty polyhedron. If D has no lines, D is said to be* pointed.

Then we have a basic theorem:

Theorem 2.1 (theorem of decomposition) *Let D be a non-empty polyhedral domain whose vertices are $s_1, ..., s_n$ and whose extremal rays are $r_1, ..., r_m$. Then every point z of D may be expressed in the following form:*

$$z = \Sigma_{k=1}^{n} \lambda_k s_k + \Sigma_{l=1}^{m} \mu_l r_l$$

with $\lambda_k \geq 0$ for all k in $[1..n]$, $\mu_l \geq 0$ for all l in $[1..m]$, and $Sigma_{k=1}^{n} \lambda_k = 1$. Moreover, if D is a pointed polyhedron this set is minimal under inclusion.

A definition comes from this theorem:

Definition 2.17 (generating system) *If D is a pointed polyhedron, the set of its rays and its vertices is called the* generating system.

From a practical point of view, the computation of the vertices and extremal rays of a domain given by its constraints is an exponential problem. There is several methods [11, 12, 13] and comparisons of several schemes can be found in [14].

Obviously in the case of bounded polyhedra, the theorem of decomposition only involves vertices.

2.3 Linearly dependant convex polyhedra

In practical applications, we are often led to consider a family of polyhedra. These domains are then indiced by one or more special variables (that will be called parameters) and in most of the cases we have to consider linearly dependant domains:

Definition 2.18 (linearly dependant polyhedra) *Let $P = (p_1, ..., p_l)$ be an index point, describing a polyhedron of dimension l ($\mathbf{N}^l$ most of the cases). Then a set of polyhedra $D(p_1, ..., p_l)$ will be said to* depend linearly *on the* parameters *p_1, ..., p_l if we have:*

$$z \in D(p_1, ..., p_l) \iff A.z + C.P \geq B$$

where A is a $n \times m$ matrix, C is a $l \times m$ matrix and B is an n-vector.

We have then a property that will be used in the end of this article:

Proposition 2.3 *The vertices of a linearly dependant convex polyhedron are linear functions of the parameters of this polyhedron.*

Proof Any vertex is the intersection of n linearly independant constraints of the domain. Each of these constraints is a linear function of the parameters, whence the result. □

2.4 Properties of projection

We will need the following technical lemma in order to compute projected points:

Lemma 2.1 *Let O be the origin of an euclidean space E. Let z be a point of E and $\vec{u}$ a vector of E. Then the point z', obtained by orthogonal projection of z in respect with $\vec{u}$ is such that:*

$$z' = z - \frac{\vec{Oz}.\vec{u}}{\vec{u}^2}\vec{u} \quad (1)$$

Proof Classical result of euclidean geometry. □
From a computational point of view, we have:

Lemma 2.2 *In a n-dimensionnal space, the cost of the projection of a point is equal to $O(n)$ in elementary operations.*

Proof The computation of $\vec{Oz}.\vec{u}$ needs n multiplications and $n-1$ additions giving a cost of $O(n)$. The computation of $\vec{u}^2$ is also $O(n)$. The division of $\vec{Oz}.\vec{u}$ by $\vec{u}^2$ is constant, since we have two single values. The multiplication of the result by $\vec{u}$ and the substraction to z are both yet of cost $O(n)$. Whence the result. □

We have also the following proposition:

Proposition 2.4 *If D is a polyhedron of an affine space E, and P is a projection from E to itself, then $P(D)$ is a polyhedron domain.*

Proof It is a general property of linear functions. □

2.5 Building up the lattice of faces

In fact, when computing the hypervolume of a convex domain, we will need the entire lattice of faces. There already exist methods to make this construction [10], but, in fact, we will not need the entire double description of the faces. Indeed, the list of the vertices for each face will be here sufficient so it leads to a simpler algorithm that we present here.

We will suppose to be in the case of the previous section: we have a polyhedron for which we know the sets of vertices and of facets. We will moreover suppose that this polyhedron is bounded, and so having no ray.

We will build the lattice of faces by recursive intersection of the facets. Our algorithm follows immediately from the next theorem:

Theorem 2.2 *The intersection of two faces of dimension $p < n$ is a face of dimension $p-1$ if and only it is contained in no other of dimension p.*

Proof This theorem is clear after the algebraic definition of a lattice. □

So we build our face-lattice recursively. At the first step for each pair of facets, we compute the intersection (by comparisons of the list of the vertices) and check out whether this intersection is included in another facet. If it is not the case we are in presence of two adjacent facets, whose intersection defines a face of dimension of 1 less.

2.6 Simplicial polyhedra

Definition 2.19 (simplicial polyhedron) *Let E be an affine space of dimension n. Then a full-dimensionnal polyhedron P will be said to be* simplicial *if and only if it is the affine hull of $n+1$ points $x_0, x_1, ..., x_n$ free of affine dependance.*

As we will see in section 5.3, every full dimensionnal convex domain may be split into the union of a finite number of disjoint simplicial polyhedra.

Moreover we can establish the proposition:

Proposition 2.5 *The projection of a simplicial of dimension n is a simplicial of dimension $n-1$.*

Proof This property is clear when counting the number of vertices of the projected simplicial. □

2.7 Hypervolume

In this section we introduce the concept of hypervolume that we will use for our new technique of optimization.

Definition 2.20 (hypervolume) *Let D be a subset of an euclidean space E of dimension n, included in an affine subspace of dimension m of E. We pose:*

$$H(D) = \int_D dt_1\ dt_2\ ...\ dt_m \tag{2}$$

If $H(D)$ is defined and is not infinite, $H(D)$ will be called the m-hypervolume *of D. In the following, the n-hypervolume will be called simply the hypervolume when confusion is impossible.*

In a formal presentation of these notions, one would have to define first of all the set of the parts D of the euclidean space such that $H(D)$ exists and is not infinite and verify that all the sets considered in the following theorems always verify this two properties. This would lead to technichal difficulties involving the definition of our integration scheme[1]. This is far from the scope of this article. The reader is invited to admit the following result (not very hard to establish):

Theorem 2.3 *Let D be the union of a finite number of bounded polyhedra of an euclidean space of finite dimension. Then the hypervolume $H(D)$ is defined and has a finite value.*

From its definition in terms of integral, it is clear that the hypervolume is an additive value:

Theorem 2.4 *Let D_1 and D_2 be disjoint bounded polyhedra of an euclidean space of dimension n. Then we have:*

$$H(D_1 \cup D_2) = H(D_1) + H(D_2) \tag{3}$$

Proof D_1, D_2 and $D_1 \cup D_2$ have an hypervolume from theorem 2.3 and the hypervolume is additive because of its definition as an integral. □

Some polyhedra have an hypervolume easy to determine. For instance, we have:

Theorem 2.5 *The hypervolume of an hypercube whose side is 1, is equal to 1.*

Proof This is clear with the use of the Fubini theorem (see [15, p. 265]). □

Likewise the hypervolume of the simplicial has a simple expression:

Theorem 2.6 *Let D be a n-dimensionnal simplicial polyhedron sustained by the vectors x_1, x_2, ..., x_n. Then we have:*

$$H(D) = \frac{det(x_1, x_2, ..., x_n)}{n!}$$

[1] Indeed, if we consider Riemann integration, fractals have no hypervolume, while more powerful integration scheme just as Lebesgue's may give hypervolume to some of them.

Proof D may be parametrized by the equation $z = \Sigma t_i x_i$ with $0 \leq t_i \leq 1$ and $0 \leq \Sigma t_i \leq 1$. The computation of the integral is done by introducing the jacobian of the change of variables (see [15, p. 403]):

$$\int_D dx_1 dx_2 ... dx_n \quad = \quad (\int_{0 \leq t_i \leq 1, 0 \leq \Sigma t_i \leq 1} dt_1 dt_2 ... dt_n) \times det(x_1, x_2, ..., x_n)$$

The integral term of the righthand side of the previous equation can be easily computed by recurrence and one obtains:

$$\int_{0 \leq t_i \leq 1, 0 \leq \Sigma t_i \leq 1} dt_1 dt_2 ... dt_n \quad = \quad \frac{1}{n!}$$

Whence the result. □

This result is quite powerful and enables us with the decomposition scheme described in section 2.6 to compute easily the hypervolume of any polyhedron. However the use of this formula in the case of non full-dimensionnal domains would require at first an orthonormal change of basis. Indeed, one would need to have an orthonormal basis of the affine space sustaining the polyhedron and to express the coordinate of the vertices in this new basis to be able to use the latter formula. This problem might be overcome by using a slight generalization of the following theorem:

Theorem 2.7 (Theorem of Hadamard) *Let $b_1, b_2, ..., b_m$ be vectors of a n-dimensionnal space ($n \geq m$). Then the m-hypervolume of the parallelepiped sustained by $b_1, b_2, ..., b_m$ is equal to $\sqrt{det(B^T.B)}$ where B is the matrix whose columns are the vectors $b_1, b_2, ..., b_m$.*

Proof This classical result of the euclidean geometry is proven for instance in [16]. □

An analogous theorem may be established for the case of the simplicial polyhedra:

Theorem 2.8 *Let $b_1, b_2, ..., b_m$ be vectors of a n-dimensionnal space ($n \geq m$). Then the m-hypervolume of the simplicial polyhedron sustained by $b_1, b_2, ..., b_m$ is equal to $\frac{1}{m!}\sqrt{det(B^T.B)}$ where B is the matrix whose columns are the vectors $b_1, b_2, ..., b_m$.*

Proof This result is established by continuity: The m-hypervolume of a simplicial is proportional to the m-hypervolume of the corresponding hyperparallelepiped. This quotient is locally constant, i.e. any infinitesimal modification of the simplicial does not change the quotient. Moreover topologically speaking, the space of the m-tuples of $\mathbf{R}^n$ is connex. So by application of [17], this quotient is constant. Its value is equal to $\frac{1}{m!}$ in the case of the hypercube of dimension m (theorems 2.5 and 2.6). □

The computational cost of the hypervolume is easy to determine:

Theorem 2.9 *The computational cost of the m-hypervolume of a simplicial sustained by m vectors in dimension n, is $O(m^2.(m+n))$ in elementary operations.*

Proof Indeed the computation of any element of the matrix $B^T.B$ needs n multiplications and $n-1$ additions. So the cost of the computation of one element is $O(n)$. There is m^2 elements so its leads to $O(m^2.n)$ for the whole $m \times m$-matrix $B^T.B$. The computation of the determinant of a $m \times m$-matrix is done by a gaussian elimination [18, p. 40] whose cost

is $O(m^3)$ and then the multiplication of the diagonal coefficients whose cost is $O(m)$. The extraction of the square root leads to a number of elementary operations negligible in regard of the other terms. Note that since $\sqrt{det(B^T.B)}$ is equal to $m!$ times the hypervolume of a simplicial of rationnal coordinates, it is also rationnal. This ensures that the computation of the square root terminates. Adding those terms we establish the bound given above. □

2.8 Infinite norm

We also introduce the infinite norm $|\ .\ |_\infty$:

Definition 2.21 (infinite norm) *Let $x = (x_1, x_2, ..., x_n)$ be a vector of an affine space of dimension n. Then:*

$$| x |_\infty \quad = \quad Sup_{i=1..n} | x_i | .$$

2.9 The eulerian function Γ

We introduce here the eulerian function Γ that permits to have an approximation of the factorial. The interested reader is referred to [19, chap. 9] for the demonstrations of the properties stated in this section.

Definition 2.22 *The function Γ is defined over $\mathbf{R} - \mathbf{Z}^-$ by:*

$$\Gamma(z) \quad = \quad \int_0^{+\infty} e^{-z} t^{z-1} dt \tag{4}$$

We will use in fact two properties of this function:

Proposition 2.6 *If n is a positive non-null integer then $\Gamma(n) = (n-1)!$.*

and:

Proposition 2.7 (Stirling's formula) *When z is near $+\infty$, we have:*

$$\Gamma(z) \quad = \quad (\frac{z}{e})^z \sqrt{2\pi z}(1 + O(\frac{1}{z}))$$

where e is the base of neperian logarithm.

This formula will enable us to avoid consideration of parity in factorial terms in our study of complexity as we will see in paragraph 5.4.

3 The problem of the optimization

In the method of automatic synthesis ([4, 6, 5]) the algorithm is represented by a collection of recurrence equations of the form:

$$\forall z \in D_U,\ U(z) \quad = \quad f(..., V(I(z)), ...)$$

where D_U is a polyhedral domain. The instances of the variables are then defined by a least fixed point semantics. The methods then proceed in two steps:

- Determine a schedule $T(U, z)$ which respects the causality of the problem and which represents the time at which $U(z)$ is being computed;

- Determine an allocation function $P(U, z)$ which maps the entire domain of definition of the system onto a subspace of less dimensions.

For reasons of regularity functions T and P are supposed to be linear. The first of these two problems has been widely studied [20, 21, 22] and is out of the scope of this article. As for the second point, let alone some rare exceptions ([23]), most of the researchers have taken a projection as allocation function [4, 6]. In most of the practical cases, there are several functions that are solutions for the specifications. The choice is then made by optimization of technological criterion, such as latency [8], total number of processors in the target architecture [8], composite criterion [24, 7], regularity [22]. We are interested here in the optimization of the number of processors. We give here a naive definition in order to make it clear:

Definition 3.1 (Number of processor) *Let D be the domain of a system of recurrence equations. Let P be the function of allocation of the final architecture. Then the* number of processors *of the architecture is defined to be $Card(P(D))$.*

4 Previous works

As we have seen the problem is the following: we have an algorithm with an iteration domain D that we wish to project in the direction of the vector $\vec{u}$ on a final architecture in order to have the minimal number of processors.

4.1 Infinite domains

First of all one can easily treat the case where the domain D is infinite. Indeed, the final architecture must have a finite number of points. Then if the domain D has exactly one ray, necessarily this ray has to be the direction of projection. Reciprocally, this direction leads to a finite number of processors.

If the domain has more than one ray then it is impossible to find a direction of projection which leads to an architecture with a finite number of processors.

4.2 Gachet's method

In [8], Gachet designs a method to optimize the number of processors in the final architecture of VLSI arrays. Indeed in practical cases the direction of projection which realizes the minimum of processors has often coordinates equal to 0, 1 and -1. The method then amounts to enumerating all the non-null vectors of this kind and to determine the one that gives the least number of points in the projection of the domain of iteration. Since there is a finite number of vectors to be considered (in fact $\frac{3^n-1}{2}$ vectors if n is the dimension of the iteration space), the algorithm ends.

Comments: The principle of this heuristic is to obtain the maximum number of occultations: two points on the same line are projected on the same processor of the target architecture.

There are several problems in this method. First of all, Gachet does not deal with how to count the number of points in the projected domain. So we can suppose a systematic enumeration that is of significant computational cost. Moreover this cost is multiplicated by the number of possible directions that is to say $\frac{3^n-1}{2}$. For example, when dealing with an hypercube of dimension n, whose size is only s, this leads to $s^{n-1}(3^n-1)$ computations and

so we can conclude that this method is highly and exponentially dependant of the numerical values appearing in the definition of the domain. In practical applications this growth makes this method quite unsuitable.

At last one may add that there is counterexamples for which this algorithm does not exhibit the direction that realize the absolute minimum. This is the case if the minimization direction has in fact coordinates with integral coordinates greater than 1. Such a case is illustrated by Fig. 1 and Fig. 2.

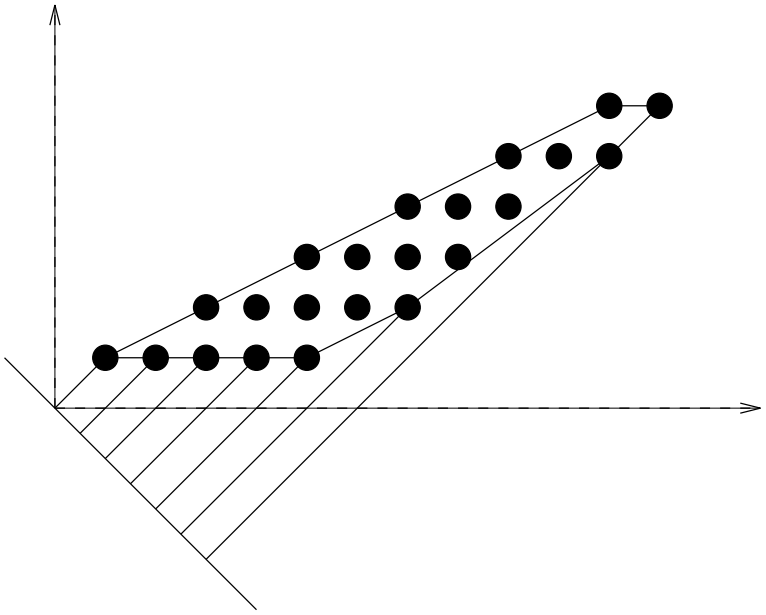

Figure 1: Minimal reached with Gachet's method

5 Number of Processors: A new heuristic

In this section we present a new method to optimize the number of processors. This method amounts to substituting this criterion with an alternative one easier to compute and to selecting directions of projection in order to limit the combinatorial growth of the problem.

5.1 Informal description of the method

First of all we develop the main outlines of our method on the example illustrated by Fig. 1. As we have already seen, the minimum of processors is obtained for direction $(2, 1)$. This direction is in fact nearly the one of the largest diameter of the domain, which relies the point $(12, 7)$ to the point $(1, 2)$, whose value is so $(11, 5)$. However, if we consider a direction of projection such as $(7, 3)$ which is nearer the largest diameter, this leads to a worse result of 15 points in the projected domain.

The optimality of the direction $(2, 1)$ comes also from the small values of its coordinates, which maximize the number of occultations of points. The conjonction of these two properties is necessary to expect an absolute minimum. Indeed if we take $(1, -1)$ as direction of projection we are led to 17 points. Then, informally, when considering only directions of

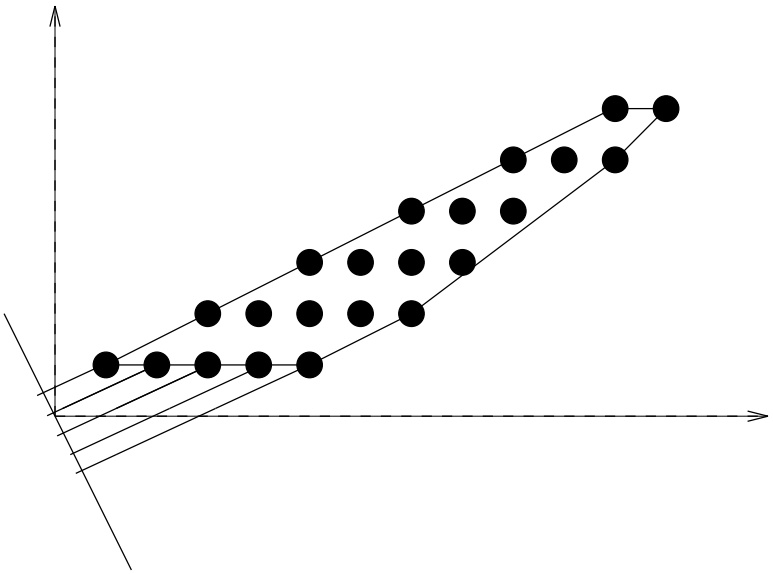

Figure 2: Absolute minimum

projection which are near the large diameters of the domain, and with small absolute values of coordinates, we can expect to obtain good results.

The second problem that we have to overcome is to avoid a systematic enumeration to count the points in the projected domain. One may remark on both Fig. 1 and Fig. 2, that the points on the projected domain are equidistant, so the measurement of the length of this domain gives a good approximation of the number of points. This property of equidistance is not true in the general case but in most cases, projected points are periodic, all the more in the interior of the domain. Another remark is that the points are not separated with the same length on both figures. In order to render this length meaningfull one has to divide it with the distance of two neighbours on the projection domain. As we will see, lengths or areas are much easier to compute and so we will substitute this kind of criteria to the previous one.

5.2 The algorithm

Let us denote $(e_1, e_2, ..., e_n)$ a basis of the iteration space. Let $(s_1, s_2, ..., s_r)$ be the vertices of the domain. Then the algorithm will be the following one:

- **For each pair** s_a, s_b
  - Let M be the maximum of the absolute values of the coordinates of $s_a - s_b$,
  - **For each integer** l from 1 to M
    - Compute V_l which is the integral vector minimizing $\mid V_l - \frac{l}{M}(s_a - s_b) \mid_\infty$. If V_l is not unique then consider V_l as the family of the matching vector.
    - **For each vector** $\vec{v}$ in V_l,
      . Compute the projection of each point $(s_1, s_2, ..., s_r)$ along v,
      . Compute the projection of the basis $(e_1, e_2, ..., e_n)$,
      . Decompose both polyhedra in simplicial polyhedra and compute the hypervolume of both polyhedra,
      . Divide the hypervolume of the projection of the domain by the hypervolume of the projection of the basis giving $Vol_{\vec{v}}$.
- Select the vector $\vec{v}$ which minimizes $Vol_{\vec{v}}$.

5.3 Decomposition of a domain in simplicial polyhedra

The method of decomposition we present is exponential with the dimension but polynomial with the number of vertices of the domain. Indeed, this algorithm avoids the general methods of linear programming. This method is in fact recursive on the dimension of the domain. First of all, we have the following theorem:

Theorem 5.1 *Let H be an hyperplane of an affine space E of dimension n. Let D be a simplicial of dimension $n-1$ included in H and let s be a point such that $s \notin H$. Then the convex hull of s and D is a simplicial of dimension n.*

Proof Just count the number of vertices. □
Given this theorem, we can state:

Theorem 5.2 *Let C be a bounded convex domain of an affine space E of dimension n. Suppose that C is full-dimensionnal and that each of its facet is decomposed into simplicials of dimension $n-1$. At last let us suppose that s is an interior point of C. Then C can be decomposed into simplicials of dimension n, by the set $\{conv.hull(s, D) \mid D$ is a simplicial of a facet$\}$.*

Proof The theorem 5.1 ensures that any domain of the form $conv.hull(s, D)$ is indeed a simplicial. The disjointness of these domains comes by induction since the facets are also disjointively decomposed and since the facets of a polyhedron are always disjoint. At last if we suppose that we have a point s' of C which does not belong to any set of the form $conv.hull(s, D)$ then since the domain is bounded the straight half-line $[s, s')$ must intersect at least one facet and so we are easily led to a contradiction since the facets are supposed to be decomposed. □

So given this algorithm, if we are given an interior point, we recursively address the same problem but in least dimension.

We are led to dimension 1, for which the problem is immediate:

Theorem 5.3 *Let C be a bounded full-dimensionnal convex domain of dimension 1. Then C is a simplicial.*

Proof In dimension 1, the only bounded convex domain are segments, which have exactly two vertices. □

This algorithm poses the problem of the election of an interior point. But here the domains are given by their vertices. Then we have:

Lemma 5.1 *Let C be a bounded full-dimensionnal convex domain of an affine space of dimension E, whose vertices are $\{s_1, s_2, ..., s_p\}$ with $p > n$. Then the point: $s = (1/p)\Sigma s_i$ is an interior point of C.*

Proof Trivial. □

The last problem left with this algorithm is then to give the list of the vertices of each facet of a convex domain given.

Let $s_1, s_2, ..., s_p$ be the set of the vertices of a domain D. We will use the following algorithm in the case where equations of the facets are not given:

- **For each n-uple** $s_{i_1}, s_{i_2}, ..., s_{i_n}$
  - Compute the affine form f of the supporting hyperplane
  - Let $F = \emptyset$
  - **For each vertex** s'
    - $F = F \cup \{f(s')\}$
  - If all the values of F are of the same sign (or eventually null) then $f(z) = 0$ defines a facet.
  - Select all the vertices for which the corresponding value in F is equal to 0.
    At this point you have the facet given by its vertices.

This algorithm is clear on the characterization of a facet. In the case, fairly common, where the equations of the facets are given one avoids the four first steps.

5.4 Decomposition of a projected domain

In fact the whole decomposition is not to be made at each step: the decomposition of the initial domain D induces a decomposition of the projected domain $P(D)$. Indeed as we have already seen, the projection of a simplicial is still a simplicial. Every point of the domain D belongs to an unique simplicial S of the decomposition. So every point of the projected domain $P(D)$ belongs to the projection $P(S)$ of one simplicial of the first domain. But here we lose the unicity since the projection of two simplicials of D may not be disjoint.

In order to infere a decomposition on the projected domain, we define a subset of simplicials of the initial domain such that their projection form a partition of the projected domain.

Definition 5.1 (Supporting simplicials) *Let D be a polyhedral domain of dimension n and let F be one on its facet. A simplicial S of dimension n will be said to be a supporting simplicial for F if and only if $S \subset D$ and there is a facet of S which is included in F. This relation will be denoted $S \mathcal{R} F$*

Given this notion we can establish the following theorem:

Theorem 5.4 *Let D be a full-dimensionnal polyhedral domain of dimension n, let F_i be the set of its facets, defined by their equation $a_i.z \geq b_i$, and let S_j be a decomposition of D in terms of simplicial polyhedra of dimension n. We will suppose that each simplicial supports exactly one facet. Let P be a projection whose direction is the vector $\vec{u}$. Then a decomposition of $P(D)$ is given by the set $\{P(S_j \cap F_i) \mid S_j \mathcal{R} F_i,\ a_i.\vec{u} > 0\}$.*

Proof Let y be a point of $P(D)$. Then the line $(y, \vec{u})$ has a non null intersection with D. This intersection is necessarily bounded since D is bounded and so is a segment of the form $I = [a, b]$ eventually reduced to a single point. The intersection of two facets or more is a negligible set respectively to a facet (its measurement is null). So we neglect these cases and suppose that both a and b belongs to exactly one facet of D. For the same reason, we will suppose that $a \neq b$. Indeed if $a = b$ then this point is necessarily on the intersection of at least two facets. If $a \neq b$ then $(a + b)/2$ is interior to D. Let $a_1 x \geq b_1$ (resp. $a_2 x \geq b_2$) be the saturating inequality of a (resp. b). We have $a - (a+b)/2 = k_1\vec{u}$ and $b - (a+b)/2 = k_2\vec{u}$ with k_1 and k_2 of opposite sign. Then for all $\epsilon \in]0, 1]$ the point $a - \epsilon k_1\vec{u}$ is in D. In particular this implies:

$$\begin{aligned} a_1(a - \epsilon k_1\vec{u}) > b_1 &\rightarrow a_1.a - b_1 > \epsilon k_1 a_1.\vec{u} \\ &\rightarrow 0 > \epsilon k_1 a_1.\vec{u} \\ &\rightarrow 0 > k_1 a_1.\vec{u} \end{aligned}$$

We have respectively $0 > k_2 a_2.\vec{u}$. So since k_1 and k_2 are of opposite signs exactly one of the two numbers $a_1.\vec{u}$ and $a_2.\vec{u}$ is positive. That is to say that exactly one of the points a and b belongs to a facet F_i such that $a_i.\vec{u} > 0$. Let us suppose for instance, that it is point a. Since the simplicials S_j form a decomposition of D, a belongs to at least one of them. But a is interior of the facet F_i and the simplicials are supposed to support exactly one facet. This implies that a belongs to a unique simplicial S_j. So we have $a \in S_j \cap F_i$ and $y = P(a) \in P(S_j \cap F_i)$. So we have proved that except to a set negligible of $P(D)$, for every point $y \in P(D)$ there exists a unique facet F_i and a unique simplicial S_j such that $y \in P(S_j \cap F_i)$, $S_j \mathcal{R} F_i$ and $a_i.\vec{u} > 0$. This means that the set $\{P(S_j \cap F_i) \mid S_j \mathcal{R} F_i,\ a_i.\vec{u} > 0\}$ forms a decomposition of $P(D)$. □

A little remark is in order: the points of the negligible set are in fact in the intersection of several simplicials and so correspond on the projected domain to points in the intersection of several sets $P(S_j \cap F_i)$.

At last we have to establish that the decomposition given by our algorithm is such that every simplicial supports exactly one facet. But this property is clear under the construction: every simplicial is indeed the convex hull of an interior point of the domain and of a simplicial of one facet.

Complexity We are here specially interested in the average complexity of the algorithm of decomposition into simplicials since we wish to compare our method of optimization with Gachet's one. This study will be made after the work of May and Smith who establish in [25] the following theorem:

Theorem 5.5 *The expected number of k-dimensionnal faces $E_k(m)$ of a random polytope in $\mathbf{R}^n$ generated by m constraints is given by:*

$$E_k(m) = 2^{n-k} \sum_{i=n-k}^{n} \binom{i}{n-k} \binom{m}{i} \Big/ \sum_{i=0}^{n} \binom{m}{i},$$

where $k = 0, 1, 2, ..., n$ *and* $m > n$. *Moreover* $E_k(m) \leq \begin{pmatrix} n \\ k \end{pmatrix} 2^{n-k}$ *for all* $m > n$ *and* $lim_{m \to \infty} E_k(m) = \begin{pmatrix} n \\ k \end{pmatrix} 2^{n-k}$

Since our study does not involve the number of constraints, we will take this latter value as a majorant of $E_k(m)$. Using this theorem, we are able to determine the average number of simplicials in our decomposition of the domains:

Theorem 5.6 *The expected number of simplicials in the decomposition scheme presented in section 5.3 is* $2^n.n!$.

Proof Let us denote $\mathcal{S}_n$, the average number of simplicials for a n-dimensionnal domain. The decomposition is done by computing the barycentre of all the vertices of the domain and recursively decomposing each of the facets. Each simplicial of one facet will lead to a simplicial for the domain itself. So we have the relation:

$$\begin{aligned} \mathcal{S}_n &= E_{n-1}(m)\mathcal{S}_{n-1} \\ &= \begin{pmatrix} n \\ n-1 \end{pmatrix} 2\mathcal{S}_{n-1} \\ &= 2n\mathcal{S}_{n-1} \end{aligned}$$

since $E_{n-1}(m)$ is the expected number of facets for a domain of dimension n. We deduce then the result from an immediate recurrence. □

This number of simplicials will determine the cost of the decomposition of a projected domain. Indeed, we have the following theorem:

Theorem 5.7 *The average cost of the decomposition of a projected domain is* $O(3^n.n)$ *in elementary operations, if the equations of the facets of the original domain are given.*

Proof Each simplicial supports exactly one facet of the original domain D. This one to one relation is determined when decomposing D, so we will suppose this information to be preserved. One has then for this facet to compute the product $a_i.\vec{u}$. The cost of this product is $O(n)$. But this computation must be done only once for each facet. The average number of facets is $2n$ so this cost is negligible in regard to the complexity expected. Then each vertex of the selected simplicials is to be projected. This projection will cost $O(n)$ but a point may be common to several simplicials. The points to be projected are the vertices of D and each of the barycentres of each face of D. The number of vertices is expected to be 2^n according to the formula of theorem 5.5. The total number of the barycentres of the faces is equal to:

$$\begin{aligned} \sum_{k=1}^{n-1} E_k(m) &= \sum_{k=1}^{n-1} \begin{pmatrix} n \\ k \end{pmatrix} 2^{n-k} \\ &= \sum_{k=1}^{n-1} \begin{pmatrix} n \\ k \end{pmatrix} 2^{n-k} 1^k \\ &= 3^n - \begin{pmatrix} n \\ 0 \end{pmatrix} 2^n - \begin{pmatrix} n \\ n \end{pmatrix} 2^0 \\ &= 3^n - 2^n - 1 \end{aligned}$$

according to binomial formula.

So the total number of points to be effectively projected is $3^n - 1$, whence the result. □

From this theorem we deduce the following one:

Theorem 5.8 *The computation of the hypervolume of the projected domain is* $O(2^n.n!.n^3)$.

Proof For each of the simplicial appearing in the decomposition of the projected domain, one has to compute the hypervolume. According to theorem 2.9, this cost is $O((n-1)^2.(2n-1))$ i.e. $O(n^3)$. The number of simplicials appearing in the decomposition is expected to be $O(2^n.n!/2)$ i.e. $O(2^n.n!)$ since the supporting facets of the simplicials of the original domain have equal chances to have their product $a_i.\vec{u}$ positive or negative. The cost of the addition of all these values is $O(2^n.n!)$ and so negligible in regard to it. □

Since the cost of the computation of the hypervolume of the projection of the basis is negligible in regard to $O(2^n.n!.n^3)$, we will take this value as the global cost of the examination of one direction. Another source of complexity is the building of the lattice of faces. We have the following theorem:

Theorem 5.9 *The average complexity of the building of the lattice of faces is majored by* $O(n.2^{6n})$ *when the vertices and the equations of the facets are given.*

Proof As stated in section 2.5, the lattice is built in three steps:

- At first we sort the vertices by facets; for each facet we determine the set of vertices which belong to it. So each vertex has to be checked whether it saturates the equation of each facet. The complexity of this step is equal to

$$\begin{aligned} E_0^n(m) \times E_{n-1}^n(m) \times O(n) &= 2^n \times 2n \times O(n) \\ &= O(n^2.2^{n+1}) \end{aligned}$$

  since, in average, there is $E_0^n(m)$ vertices, $E_{n-1}^n(m)$ facets and since a verification for a vertex and a facet has $O(n)$ as computational cost.

- Then recursively we compute the intersection of the set of vertices for all pairs of faces of dimension p. That is to say that for any pair of faces one has to check by comparisons which vertices are in both. The expected number of vertices of a face of dimension p is equal to $E_0^p(m) = 2^p$, so for one pair the complexity is $[E_0^p(m)]^2 \times O(n) = 2^{2p} \times O(n)$, then since there is $[E_p^n(m) \times (E_p^n(m) - 1)]/2 \equiv [E_p^n(m)]^2/2 = 2^{2n-2p} \left(\begin{array}{c} n \\ p \end{array} \right)^2 /2$ pairs of faces of dimension p, the global complexity of this step at iteration p is equal to:

$$[E_p^n(m)]^2 \times [E_0^p(m)]^2 \times O(n)/2 = 2^{2n} \left(\begin{array}{c} n \\ p \end{array} \right)^2 \times O(n)/2$$

- For each of these previous sets obtained by intersection, we need to check whether the set is also included in another face of dimension p. These sets are in fact included in faces of dimension $p-1$ so the number of their vertices is expected to be majored by $E_0^{p-1}(m)$, each of them to be compared to every face of dimension p. Moreover, there is $[E_p^n(m) \times (E_p^n(m)-1)]/2 \times [E_p^n(m)-2] \equiv [E_p^n(m)]^3/2$ triples to consider so the test

of inclusion for the step p is expected to be majored by:

$$\begin{aligned}[E_p^n(m)]^3 \times E_0^{p-1}(m) \times E_0^p(m) \times O(n)/2 &= 2^{3n-3p}\left(\begin{array}{c} n \\ p \end{array}\right)^3 \times 2^{2p-1} \\ &\quad \times O(n)/2 \\ &= 2^{3n-p}\left(\begin{array}{c} n \\ p \end{array}\right)^3 \times O(n)/2\end{aligned}$$

Then adding those three terms we find that the global complexity is majored by:

$$O(n^2.2^{n+1}) + \sum_{p=1}^{n}(2^{2n}\left(\begin{array}{c} n \\ p \end{array}\right)^2 \times O(n)/2) + \sum_{p=1}^{n}(2^{3n-p}\left(\begin{array}{c} n \\ p \end{array}\right)^3 \times O(n)/2)$$

From these three terms, the third is dominant. Moreover every term of the form $\left(\begin{array}{c} n \\ p \end{array}\right)$ is majored by $\left(\begin{array}{c} n \\ \lfloor (n+1)/2 \rfloor \end{array}\right)$. Using Stirling's formula, we find that this term is asymptotically equal to 2^n. So the global complexity is majored by :

$$\begin{aligned}\sum_{p=1}^{n}(2^{3n-p}2^3 n) \times O(n)/2 &< 2^{6n}\sum_{p=1}^{+\infty}(2^{-p}) \times O(n)/2 \\ &= 2^{6n-1} \times O(n)\end{aligned}$$

Whence the result.

□

In fact, more subtle algorithms may reduce a bit this upper bound, but from this computation we see that the initial computation of the lattice of face is negligible in regard to the treatment for one vector (cf. theorem 5.8), so we will neglect this cost in the global cost of the procedure.

This complexity may seem high but one has to point out that it is independent of the values appearing in the expression of the constraints and is therefore much more suitable than the other methods and especially Gachet's one as we will see in the next sections. One may also remark that our study of complexity has been made in the average case and that in practical cases the polyhedra that we wish to treat have less vertices than the number predicted by May and Smith and so this leads to a smaller practical complexity.

6 Improvements

Before discussing this method on examples, we make three remarks that will lead to faster algorithms.

6.1 Divisions

First of all one of the major problem with the formula 1 is that it involves one division. If we try to implement directly this formula we encounter the problem of the precision on the one hand and the problem of performing a division which has an algorithmic cost higher than the other operation. In fact this problem can be easily overcome. Indeed in Gachet's

method, we are interested in counting the number of points in the projected domain. So this number remains the same if we scale the lattice of the points by $\vec{u}^2$. Doing so the formula of projection becomes :

$$Z = \vec{u}^2.z - (\vec{Oz}.\vec{u})\vec{u} \tag{5}$$

which totally avoids the division.

In our new method presented in this article we can apply the same trick to avoid computation of divisions for the projection. But in this method there is also a division by the hypervolume of the projection of the basis. Once again, it is in fact a division by an integer. This division gives a value which approximates the number of points in the projected domain and which is to be compared to the current minimum. So we can also multiply the minimum by the quotient and compare it to the hypervolume of the projection of the whole domain. This minimum is in fact in general a rational number to be multiplied by an integer and then to be compared to an integer. By reducing to the same denominator, it is obvious that this test can be made only by multiplications. So we will suppose that our algorithm involves no division and for each vector there will be a fixed number of multiplications for the final test.

6.2 Halting criteria

First of all, we need to remember that the optimal is obtained for small values of integers in the components of the projection vector. As a consequence the for-loop on l in the algorithm of section 5.2 will generally hit the optimal in one of the first iterations. This remark underscores the interest of an halting criteria that would cut exploration trees. Let us suppose that we project the domain in a direction with $0, 1$ components. Now if we search with $0, 1, 2$ components, in the worst case this direction doesn't seemingly decrease the size of the projected domain but has the disadvantage to double the number of projected points. If we are in such a case, we can reasonnably think that the optimal in this direction has already been met. So we will give up with this direction and try with another one. The worst case behavior between $0, 1, 2$ components and $0, 1, 2, 3$ components is as one can easily verify to multiply out the number of projected points by $\frac{3}{2}$, and in general between $0, 1, 2, ..., n$ and $0, 1, 2, ..., n+1$ components to multiply out by $\frac{n+1}{n}$. In the examples, this criterion will be compared with the criterion of Gachet which, as we have seen, consists in the enumeration only with $0, 1$ components.

6.3 Selective research

Another remark to be made is that minimum is often obtained in the direction of a pair of vertices whose distance is large in regard to the polyhedron. An idea is then to select in the main algorithm only the pairs of vertices (s_a, s_b) which bypass the diameter multiplied by a certain ratio. If we consider the case of any hypercube of dimension n and of size s, integer, i.e. $H = \{(x_1, x_2, ..., x_n) \mid 0 \leq x_i \leq s\}$, then the minimal number of points is always obtained for a projection parallel to the coordinate axis. The corresponding direction $(0, ..., 0, s, 0, ..., 0)$ has a length of s. As for it the diameter of H is equal to $\sqrt{n}.s$. That is to say that here one would have to keep all the diameters whose lengths are at least $\frac{1}{\sqrt{n}}$ of the diameter[2]. In general we will keep this threshold value of $\sqrt{n}$. This has two justifications: (*i*) indeed many scientific algorithms involve variables defined over hypercubes and so it is absolutely necessary that such examples are well treated; (*ii*) an hypercube is a domain

[2] ... and so, in this example, to reject none of the possible pairs of vertices.

	Gachet's method	New method
Number of possible projection vectors	$\frac{3^n-1}{2}$	$\frac{3^n-1}{2}$
Number of elementary operations by projection vector	$O(n.s^n)$	$O(2^n.n!.n^3)$
Total cost	$O(n.(3s)^n)$ el. op.	$O(6^n.n!.n^3)$el. op.

Table 1: Algorithmic costs for the hypercubes

rather compact and so in the general case a threshold of $\sqrt{n}$ will give a great selection. One can remark that the cost of the implementation of this selection is known: some preliminary computations (a sort to obtain the largest diameter and a division by $\sqrt{n}$), n multiplications and additions, and a comparison for every pair of vertices which is $O(nN)$ where N is the number of the vertices of the domain. In the general case its cost is negligible in regard to the rest of the algorithm.

7 Examples and comparisons

In this section, we will compare the algorithm with every combination of the two possible improvements with the algorithmic cost in regard to the one in Gachet's method. We remind the reader here that in all cases, contrary to Gachet's method, we avoid the enumeration of the domain. Our halting criterion is here of no effect since for every pair of vertices, the direction has values $0, 1$ in its components. So here we have no further research than the first one in every diameter direction.

7.1 Hypercubes

In this section we will study formally the case of the hypercubes of the form $H = \{(x_1,x_2,...,x_n)| 0 \leq x_i \leq s\}$ where n and s are integers. This polyhedron contains s^n points, the optimal projection direction is of the form $(0, 0, ..., 0, 1, 0, ..., 0)$ leading to s^{n-1} in the projected domain. One remarks here that the selective research is of no use in this case since it rejects none of the rays. However the number of vertices of the domain is 2^n and so the selection cost is negligible in regard to the results of the different methods, summed up in the Tab. 7.1.

As our criterion of selective search rejects none of the rays and as values of the components of the rays are all 0 and $\pm s$, the number of rays to be considered is exactly the same as in the original Gachet's method, that is to say $\frac{3^n-1}{2}$ (For each component we choose between -1, 0 and 1, we let alone the null vector, and we divide by 2 in order to identify opposite vectors). If we look now for Gachet's method, then for each vector to be considered, one has to project each point of the domain along this vector and verify whether this point projects

on a point already computed. If we only consider the first part of this algorithm (the second one being negligible in comparison), according to the formula 5 we have to perform for each point of the domain, a multiplication of a scalar with an n-vector, so of complexity $O(n)$, a scalar product between n-vectors, again $O(n)$ elementary operations, a multiplication with an n-vector, and an addition between two n-vectors, also both $O(n)$. We will let alone the complexity of the computation of $\vec{u}^2$, since it is computed once for all the points and then is negligible in regard to the rest. This leads then to $O(n)$ and as there is s^n points, this leads to a complexity of $nO(s^n)$ for each projection vector that is considered.

If we consider now the new method, the same formula of projection is applied, in most cases, to half of the simplicials appearing in the decomposition of the hypercube. The total number of simplicial is exactly $2^n.n!$. Indeed each simplicial is obtained by convexity from the center of the hypercube and a simplicial of a facet. But there is $2n$ facets and again any of it is an hypercube of dimension 1 less, whence the result by recurrence[3].

We would like to know when our algorithm becomes better. This leads to:

$$\begin{aligned} n.(3s)^n >> 6^n.n!.n^3 &\rightarrow n^{1/n}3s >> 6.(n!)^{1/n}.n^{3/n} \\ &\rightarrow 3s >> 6\frac{n}{e} \\ &\rightarrow s >> \frac{2n}{e} \end{aligned}$$

So we see that for small values of n, as soon as we consider hypercubes of reasonable size in terms of practical applications, the method of optimization presented here is much faster and in this case, the optimal direction is obtained by both methods.

7.2 A practical example

In this section, we will treat a practical example outlined from [26] for an architecture for the Gauss-Jordan elimination. This domain is the convex hull of the set of vertices $\{ (1,1,1), (n,1,1), (1,n+1,1), (n,n+1,1), (1,n,n), (n,n,n), (1,n+1,n), (n,n+1,n) \}$ and is illustrated by Fig. 3. The exact number of points of this domain can be easily computed by the method described in [27] and is equal to $(n^3+n^2)/2$. With the method of Gachet in dimension 3 there is 13 vectors to be examined, so the global cost of this method is $O(n^3)$. With our new method, the selective research limits the number of vectors to 5 vectors (the diameters of the domain) as soon as $n \geq 2$. The domain is split into 24 simplicials. When examining one vector at most 12 of them have to be projected. This projection is constant in time. So the computation of the hypervolume of the projected domain is also constant. We can conclude that whatever the value of the parameter, the computation of the optimal direction is constant with our new method. This example clearly illustrated the great superiority of this method for fixed small dimension space. In this case, our method will determine the same optimum as the method of Gachet does.

8 Optimization and Parameters

The methods that we have discussed above are effective for fixed bounded domains.

[3] One can deduce from this and the other estimations that an hypercube fits very well the model of random polytopes, let alone of course the character of randomness.

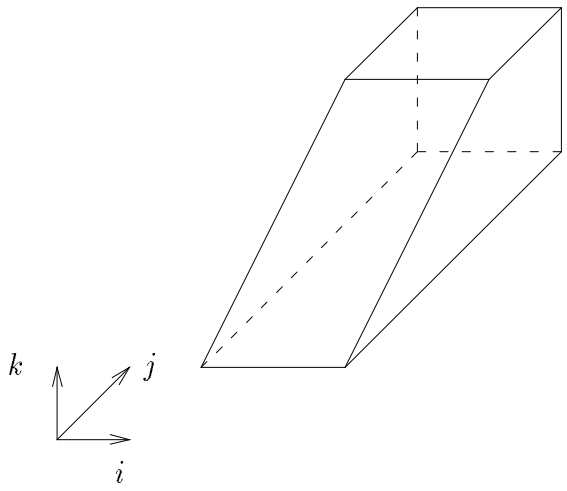

Figure 3: Domain of the Gauss-Jordan elimination algorithm

8.1 An illustrative example

In particular this excludes domains which depend linearly on parameters. Let us consider for example the simple rectangle D defined by $\{(i,j) \mid 1 \leq i \leq m,\ and\ 1 \leq j \leq n\}$ where m and n are the parameters. The optimal projections are $(0,1)$ or $(1,0)$ whether $m \geq n$ or not as illustrated in Fig. 4. If we represent the function which associate the optimal direction to a couple (m,n), then in the latter example this function is constant on two convex domains: $m \geq n$ and $n \geq m$. In fact all the practical examples behave in the same way, and so we make the following conjecture:

Conjecture 8.1 *The direction of projection which minimizes the number of points is constant by convex domains in the space of the parameters.*

This conjecture, if it appears to be true, poses the problem of determining the equations of the convex domains in the space of parameters, on which the optimal projection is constant. This problem seems to be difficult, nevertheless this conjecture enables to sketch a general optimization in the case of only one parameter.

8.2 Optimization for one parameter

Informal approach We suppose here that we are given domains D_n of dimension k linearly dependant on a parameter n. As we have seen in section 2.3, the vertices of D_n also linearly depend on the parameter n. Moreover, amongst the possible vertices (defined by the intersection of k linearly independant facets), near the infinity, the effective vertices of the polyhedron are the same and can be determined. As a consequence, near infinity, the shape of the polyhedra is always the same so it leads to the same optimization vector, say u_∞. The computation of this vector will be the first step of our optimization.

The second step will consist in finding the least value n_0 of n for which the optimization vector is u_∞. At this point, we will then make the assumption that for all $n \geq n_0$, the optimization vector is still u_∞. This is indeed the case in all the practical examples and is

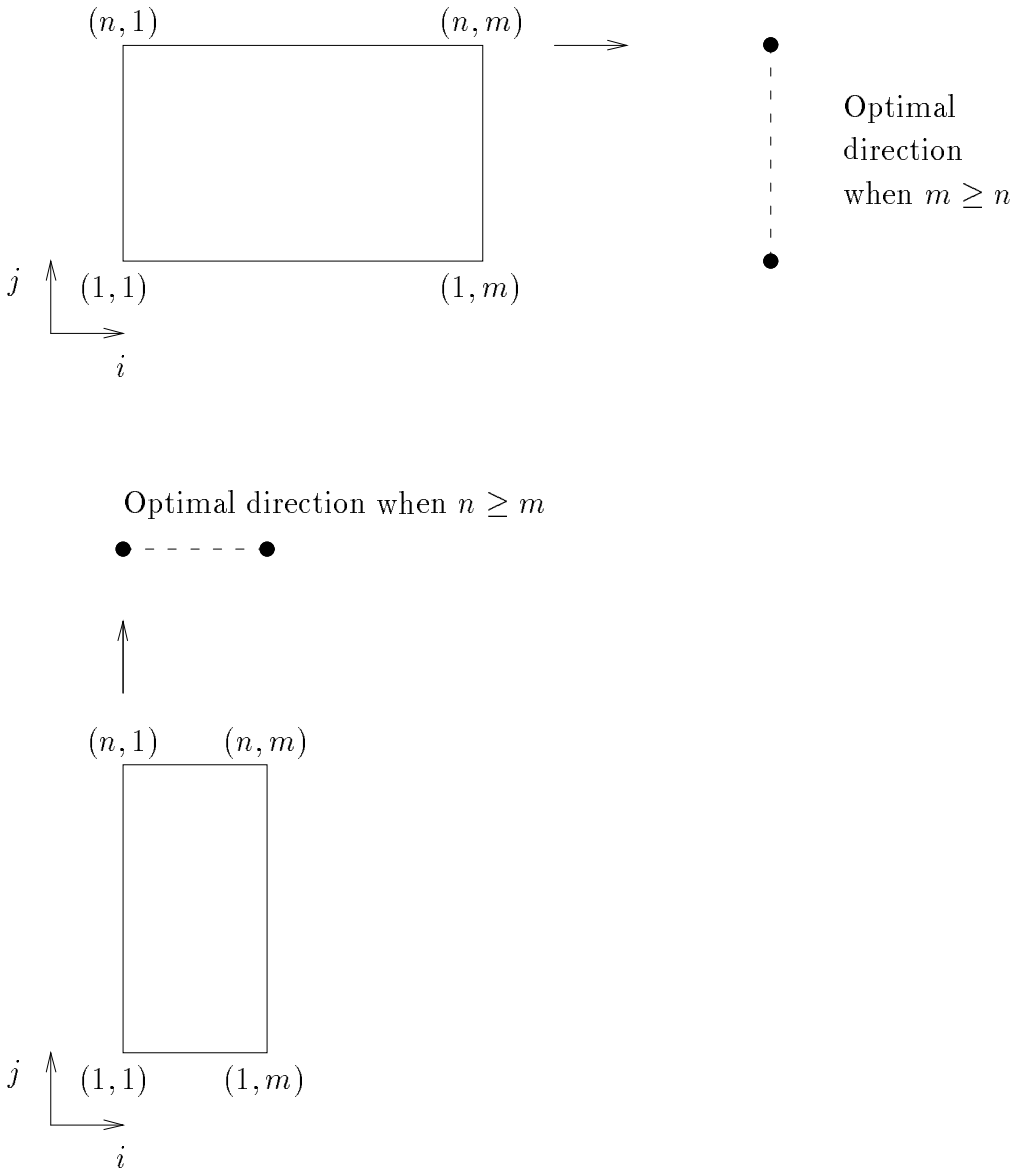

Figure 4: Optimal directions for a rectangle

likely to be true according the author's intuition. The third step will be to determine the value of the optimization vector for each value of n smaller than n_0.

It can be noticed that the optimization scheme is independant on the optimization technique used for a fixed value of n; our new method or Gachet's method, as well as any other method may be used.

Computation of u_∞ and n_0 Let $s_1(n)$, $s_2(n)$, ... , $s_l(n)$, be the expressions of the possible vertices. Those expressions are obtained by the intersection of k linearly independant facets (i.e. constraints). We pose $S_i = lim_{n\to\infty}(s_i(n)/n)$. Let D' be the convex hull, not necessarily full dimensionnal, of the points S_i. Two cases must be differenciated for the computation of u_∞ whether D' is full dimensionnal or not.

If D' is not full-dimensionnal, then any vector of the smallest vectorial space containing D' can take the role of u_∞. If D' is full-dimensionnal, then u_∞ will be the optimal projection vector of D'. If there is more than one vector which realizes this optimum, one makes the list of all these vectors.

At this point, n_0 will be the smallest value of n such that the optimal direction of projection of D_n belongs to the set of the optimal directions for D'. This common vector will be u_∞. A dichotomic searching procedure may be used in order to limit the time of computation.

An example We illustrate this method on the example that we already used in section 7.2. Let us call D_n this domain. The points $s_i(n)$ are here explicitely given and D' is the convex hull of the points $(0,0,0)$, $(1,0,0)$, $(0,1,0)$, $(1,1,0)$, $(0,1,1)$ and $(1,1,1)$. This domain is illustrated by Fig. 5.

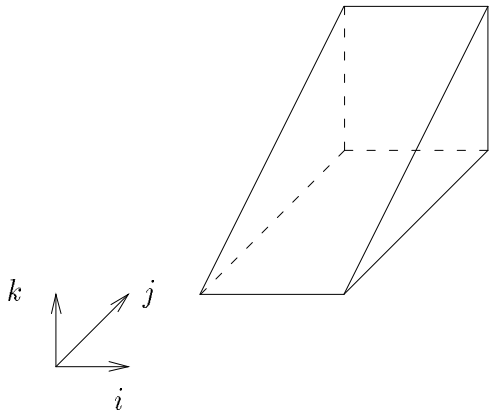

Figure 5: Optimisation by parameter of Gauss-Jordan domain

This domain admits only the vector $(1,0,0)$ as optimal direction of projection. This vector is then u_∞. For value $n = 1$, the domain D_1 contains only two points $(1,1,1)$ and $(1,2,1)$. Then for this domain the vector $(0,1,0)$ is the optimal direction of projection. Again for D_2, the reader may verify that $(0,1,0)$ is the optimal direction of projection. The first value of n such that the optimal direction is $(1,0,0)$, is $n_0 = 3$. So, by our method, the

global optimization vector will be described by the equation:

$$\vec{u} = \begin{cases} \textbf{if} & n \leq 2 & \textbf{then} & (0,1,0) \\ \textbf{if} & n \geq 3 & \textbf{then} & (1,0,0) \end{cases}$$

This solution is exactly the one that is given by separate optimization for each value of n.

9 Conclusion

In this article we presented a new method to optimize the number of processors of an architecture. The key to this method is the substitution of this criterion quite uneasy to compute and handle by another one much easier to determine. This principle of substitution seems to be quite useful when dealing with non linear criteria and so one may expect that this kind of substitution may also apply for e.g. the composite criteria PT and PT^2 [24]. Another interesting direction of research is to demonstrate the conjecture 8.1. Indeed, this would permit to understand better the criterion of number of processors and so to expect a procedure of exact optimization. Moreover this would pose the problem to know whether there is similar conjecture for composite criterion in the space of P and T. The author suspects that this conjecture is very hard. At last, if we admit (or prove) this conjecture one can also try to design methods of optimization for several parameters. One may for instance figure heuristic algorithms to determine the cutting planes of the domain on which the optimal direction is constant.

10 Acknowledgements

The author wish especially to thank Jean-Claude Raoult of IRISA for his helpful advices in the redaction of this article.

References

[1] H.T. Kung, Why systolic architectures? *Computer*, **15**(1)(Jan. 1982)37-46.

[2] D.I. Moldovan, On the analysis and synthesis of vlsi algorithms. *IEEE Trans. on Comp.*, **C-31**(11), (Nov. 1982).

[3] S.K. Rao, Regular Iterative Algorithms and their Implementation on Processor Arrays, Ph.D. thesis, Stanford Univ., USA, 1985.

[4] P. Quinton, *The Systematic Design of Systolic Arrays*, chapter 9, (Manchester University Press, 1987)pp. 229-260.

[5] S. V. Rajopadhye, S. Purushothaman, and R. M. Fujimoto, On Systolic Array Synthesis from Recurrence Equations with Linear Dependencies. Tech. Rep. UUCS-86-009, Univ. of Utah, 1986.

[6] J-M. Delosme and I.C.F. Ipsen, Design methodology for systolic arrays, in: *Proc. SPIE 30th Ann. Int. Tech. Symp. on Optical and Optoelectronic Applied Sciences and Engineering* (San Diego, USA, 1986).

[7] Y. Wong and J-M. Delosme, Optimal systolic implementations of n-dimensionnal recurrences, In *Proc. of IEEE Int'l Conf. on Computer Design: VLSI in Computers*, (New York, USA, 1985)618-621.

[8] P. Gachet, *Conception d'algorithmes et d'architectures systoliques*, Ph.D. thesis, Rennes I Univ., France, 1987.

[9] G. Birkhoff, *Lattice theory*, (American mathematical society, 1967).

[10] A. Schrijver, *Theory of linear and integer programming*, (John Wiley and sons, 1986).

[11] N. Chernikova, Algorithm for Discovering the Set of All the Solutions of a Linear Programming Problem, *USSR Comput. Math. and Math. Phys.*, **8**(6)(1968)282-293.

[12] V. Chvátal, *Linear Programming*, (W. H. Freeman and Company, 1983).

[13] H. Le Verge, A note on Chernikova's algorithm, Internal Publication 635, IRISA, Rennes, France, 1992.

[14] T. Mattheiss and D. Rubin, A Survey and Comparison of Methods for Finding all Vertices of Convex Polyhedral Sets, *Math. of Op. Research*, **5**(2)(1980)167-185.

[15] S. Lang, *Analysis II*, (Addison-Wesley, 1969).

[16] E.F. Beckenbach and R. Bellman, *Inequalities*, (Springer, 1983).

[17] J. Dieudonné, *Eléments d'analyse 1 : Fondements de l'analyse moderne*, (Cahiers scientifiques, 1968).

[18] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, (Springer-Verlag, 1988).

[19] S. Saks and A. Zygmund, *Analytic Functions*, (PWN - Polish Scientific Publishers, 1965).

[20] C. Mauras, P. Quinton, S. Rajopadhye, and Y. Saouter, Scheduling affine parameterized recurrences by means of variable dependent timing functions, in: S.Y. Kung, E.E. Schwartzlander, J.A.B. Fortes, and K.W. Przytula, eds., *Application Specific Array Processors* (IEEE Computer Society Press, Princeton University, 1990), pp. 100-110.

[21] S.V. Rajopadhye, Synthesizing systolic arrays with control signals from recurrence equations, *Distr. Comput.*, **3**(1989)88-105.

[22] M. van Swaaij, Data Flow Geometry: Exploiting Regularity in System-level Synthesis, Ph.D. thesis, Interuniversitair Mikro-Elektronica Centrum, Leuven, Belgium, 1992.

[23] P. Lee and Z.M. Kedem, Mapping Nested Loop Algorithms into Multi-dimensionnal arrays, Tech. Rep., New York Univ., 1988.

[24] J.D. Ullman, *Computational Aspects of VLSI*, (Computer Science Press, 1984).

[25] J.H. May and R.L. Smith, Random polytopes, *Math. Programming*, **24**(1982)39-54.

[26] P. Quinton and V. Van Dongen, The Mapping of Linear Recurrence Equations on Regular Arrays, *The J. of VLSI Signal Proc.*, **1**(1989)95-113.

[27] N. Tawbi, Parallélisation automatique : Estimation des durées d'exécution et allocation statique de processeurs, Ph.D. thesis, Paris VI Univ., 1991.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399