Polyedral model and program analysis

Claude Tadonki

LAL/CNRS/IN2P3 University of Paris-Sud
claude.tadonki@u-psud.fr

December 2010

The polyhedral model is a well developed formalism that has been extensively used in a variety of contexts

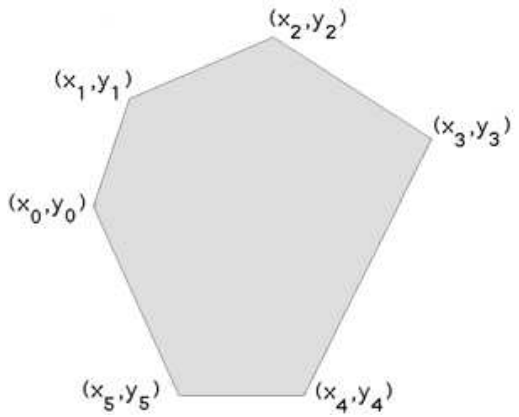
- mathematical programming (linear/quadratic programming)
- instructions space-time scheduling
- automatic transformation and parallelization of loop programs
- program verification
- data locality and cache analysis
- low level code and hardware generation
- automatic reduction of asymptotic program complexity.

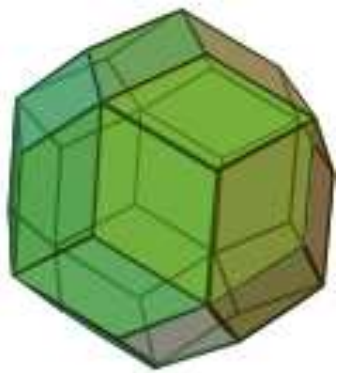
Definition

A polyhedron is a set $P = \{x \in R^n : Ax \leq b\}$, where A is a $m \times n$ matrix, and b a $m \times 1$ vector. A Z-polyhedron is the intersection of a polyhedron and an affine integral lattice.

Example

(a) $P = \{(x, y) : (3x + 2y \leq -1) \wedge (x - y \leq 4) \wedge (-2x + 5y \leq 0)\}$.



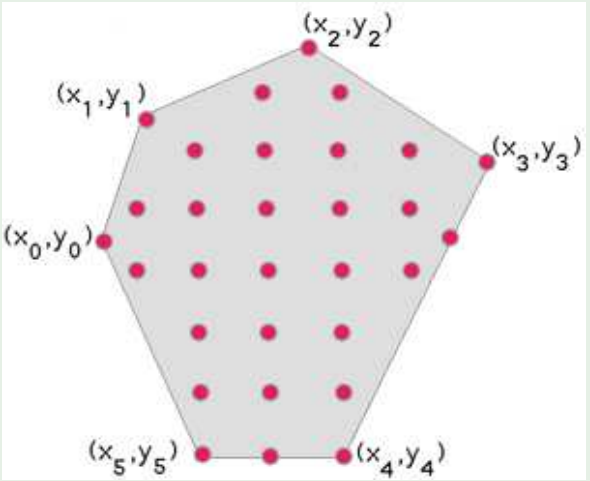


Important elements/properties of a polyhedron include:

- dimension
- shape, convexity, bounds (qualitative properties!)
- vertices, edges, sides, faces
- peaks
- volume (number of vertices, very useful for complexity analysis!)

Example

- shape = *hexagon*
- dimension = 2
- peaks = $\{(x_0, y_0), (x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)\}$
- vertices = red points
- volume = 30



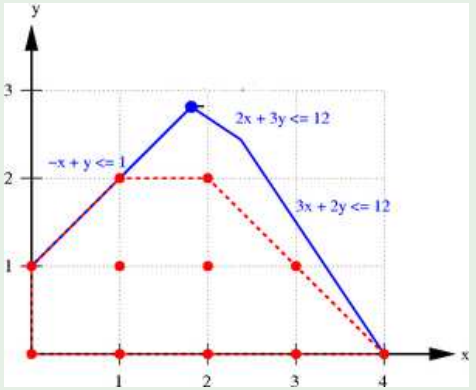
When considering a (Z-)polyhedron (from an explicit picture or the representation we have in mind), we should note that

- not all points are part of the polyhedron (additional constrains)
- the peaks and/or the edges/faces are not necessarily included
- every canonical projection (i.e. $x_j = cste$) is also a polyhedron

Its important to keep in mind that most of the time, we draw the envelop of the polyhedron from the analytical formalism.

Example

- shape = *pentagon*
- dimension = 2
- peaks = $\{(0,0),(0,1),(1,2),(2,2),(4,0)\}$
- vertices = red points
- volume = 11



Exercise 1. For each of the following polyhedra, write down the corresponding matrix A and vector b.
(a) $P\{i,j,k)|(2i+j \leq 1) \wedge (i+3j-k \leq -2)\}$.
(b) $P = \{(i,j,k)|(2i+j \leq 1) \wedge (i+3j-k \leq -2) \wedge (i \geq 0) \wedge (i \geq j-k)\}$.
(c) $P = \{(i,j)|(2i+j \leq 1) \wedge (i=j)\}$.

Exercise 2. Calculate the volume of each of the following N-polyhedra
(a) $P\{i,j)|(0 \leq i,j \leq n) \wedge (i \leq j)\}$.
(b) $P\{i,j,k)|(0 \leq i,j,k \leq n) \wedge (i+j+k \leq n)\}$.

Exercise 3. Consider the polyhedron
$P = \{(i,j)|(0 \leq i \leq 5) \wedge (0 \leq j \leq 7) \wedge (\frac{4}{3}i+3 \leq j \leq \frac{5}{2}i+\frac{15}{2})\}$.
1) Draw the envelop of P in a cartesian space.
2) List the peaks of P.
3) Calculate the volume of P.

The polyhedral model can be used to model or express a specific subregion of a n-dimensional array. This can serve as a basis for

- direct array reference
- data partitioning
- tiling
- data locality
- data dependence

Example

The non zero entries of the matrix are located at $P = \{(i,j) \in \{1,...,n\}^2 : j-1 \leq i \leq j+1\}$, which can be written in a canonical form as $P = \{(i,j) : (j-i \leq 1) \wedge (i-j \leq 1))\}$.
This model can be extended to $(p,q)-$banded matrices by
$P = \{(i,j) : (j-i \leq p) \wedge (i-j \leq q))\}$

$$\begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & 0 \\ 0 & 0 & 0 & a_{54} & a_{55} & a_{56} \\ 0 & 0 & 0 & 0 & a_{65} & a_{66} \end{bmatrix}$$

Exercise. Express the polyhedron corresponding to a upper (resp. lower) triangular matrix.

This is one of the major application of the polyhedral model. A given loop is is modeled by

- a polyhedral domain $D = \{x : Ax \leq b\}$.
  Also called iteration space, it is usually expressed with
  - the bounds of the enclosing loops
  - the elementary displacement vector
- a list of access function $\ell(x) = Bx + c$.
  This describes array references.
- a timing function (or schedule) $t(x) = u^T x + v$.
  An affine function assigning logical execution dates to iterations.

Remark

- within the body of the loop, additional constraints (i.e. control statements) may encountered.
- the bounds of the loops can be constant, parameterized, or dynamic (while loops).

- Domain:

$$\{(i,j) : \begin{pmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 1 & -1 \\ 1 & 3 \\ 2 & -1 \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix} \leq \begin{pmatrix} n \\ -1 \\ n \\ 0 \\ n \\ 0 \end{pmatrix} \}$$

```
for(i=1; i<= n; i++)
  for(j=i; j<=n; j++)
    if((i+3j<=n)&&(2i<=j))
S:    M[i+3j][2i-j+n] = 1;
```

- Access function:

$$\ell(i,j) = \begin{pmatrix} 1 & 3 \\ 2 & -1 \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} 0 \\ n \end{pmatrix}$$

- Timing function:

$$\ell(i,j) = \begin{pmatrix} n+1 & 1 \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix} - n$$

Exercise 1. Let consider the loop below

```
for(i=1; i<= n; i++)
  for (j=i; j<=i+3; j++)
    if (j<=n)
S:       M[j-i][j] = i+j;
```

1) Express the domain of the loop.
2) Express the access function of the loop.
3) Express the timing function of the loop.
4) Calculate the volume of the domain (complexity of the loop!).

Exercise 2. Let consider the loop below

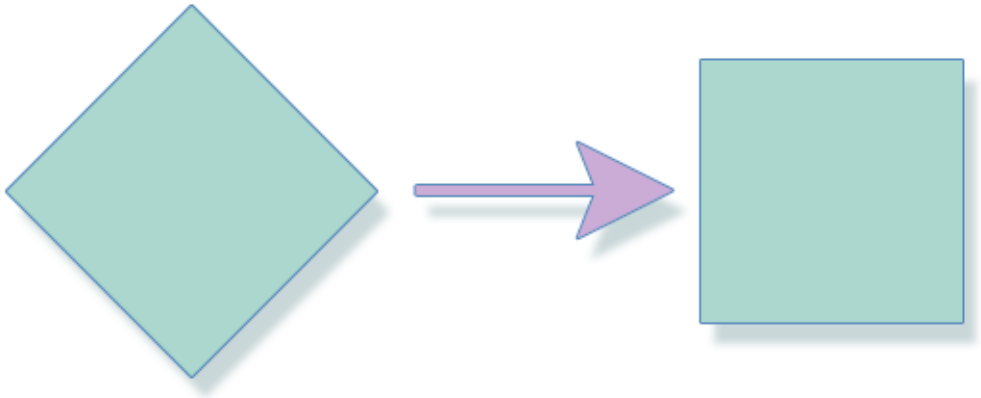
```
for(i=1; i<= n; i++)
   for(j=i; j<=n; j++)
     for(k=i+j; k<=n; k++)
S: M[i][j-i][k-j] = i+j+2k;
```

1) Express the domain of the loop.
2) Express the access function of the loop.
3) Express the timing function of the loop.
4) Calculate the volume of the domain (complexity of the loop!).

Isomorphism between polyhedra

Depending on the context, it can be useful to transform a given polyhedron into another one with the same volume but a different shape. This is done through a point-to-point correspondence $f : P \longmapsto f(P)$. Such isomorphisms are used for

- mapping a given data to a specific memory region
- loop transformations
- program simplifications
- shuffling to improve data locality
- reindexation
- reshaping



Linearisation.

The goal of the *linearisation* is to map a two-dimensional polyhedron into a corresponding one-dimension one.

Example

Consider the polyhedron $P = \{(i,j) : (1 \leq i \leq n) \wedge (1 \leq j \leq m)\}$ defining a two-dimensional array. If we choose $f(i,j) = (i-1)n + j$, we obtain another polyhedron $f(P)$, which is one-dimensional.

Exercise. Exhibit one linearization for $P = \{(i,j) : 1 \leq i \leq j \leq n\}$.

Reindexation.

The *linearisation* mainly applies on *access functions* in order to adapt the data pattern access to a specific algorithm. This is similar to a classical *change of variable*. For example

```
for(i=1; i<= n; i++)
      M[2i-1] = 0;
```

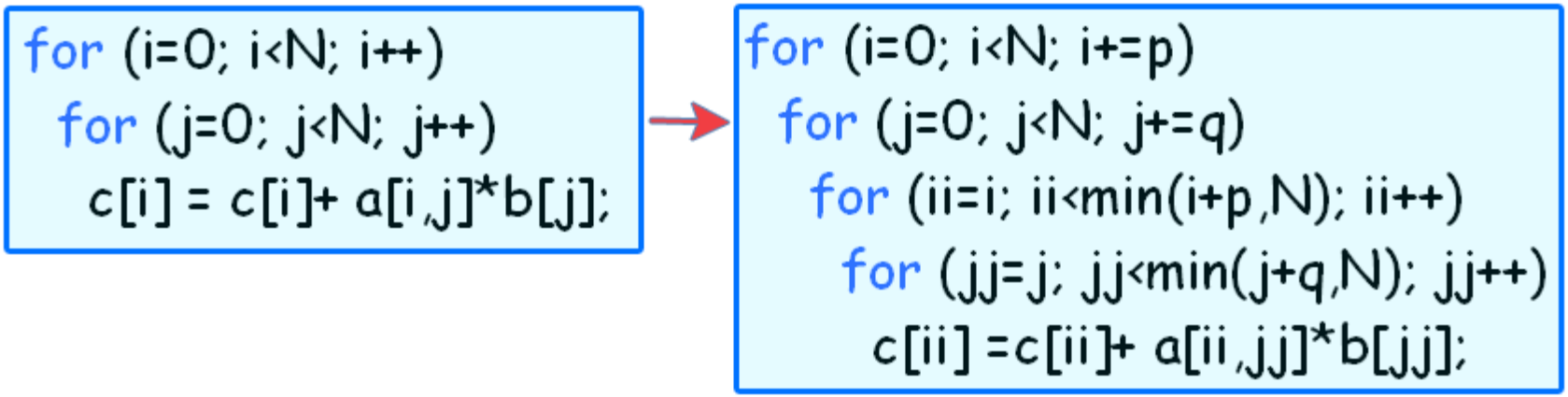
becomes

```
for(i=1; i<= 2n-1; i+=2)
      M[i] = 0;
```

Loop tiling or loop blocking is a loop optimization used by compilers to make the execution of certain types of loops more efficient. It can also apply to data, hence referred as data partitioning or clustering. There are mainly two possible purposes:

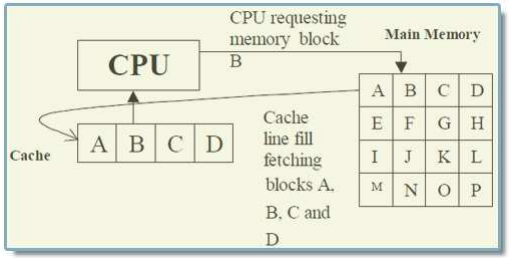
- data locality for cache misses reduction
- data packing for coarse grained interprocessor communication

A classical tiling looks like the following figure (matrix-vector product):



Exercise 1. Write a tile form of the standard matrix multiplication.
Exercise 2. Explain how tiling helps for a better data reuse.

It is important to understand that data are loaded (resp. stored) from (resp. to) the memory by blocks (cache line). The following figure illustrates the mechanism.



From that point:

- any consecutive memory blocks requested by the CPU within this spatial region will result in a cache hit
- otherwise, we got the so-called cache miss, which result in a time delay penalty.

The main goal of data locality studies is to reduce cache misses. Among existing techniques, we have

- loop reordering
- tiling
- data prefetch strategy
- use of registers

C. Tadonki – The polyhedral model

Some references

1. U. Bondhugula, A. Hartono, J. Ramanujam, and P. Sadayappan, *A Practical and Automatic Polyhedral Program Optimization System*, Proc. ACM SIGPLAN 2008 Conference on Programming Language Design and Implementation (PLDI 08), Tucson, June 2008.
2. U. Bondhugula, M. Baskaran, S. Krishnamoorthy, J. Ramanujam, A. Rountev, and P. Sadayappan, *Automatic Transformations for Communication-Minimized Parallelization and Locality Optimization in the Polyhedral Model*, in Proc. CC 2008 - International Conference on Compiler Construction, Budapest, Hungary, March-April 2008.
3. Wolfe, M. More, *Iteration Space Tiling*. Supercomputing'89, pages 655 – 664, 1989.
4. Wolfe, M. E. and Lam, M. *A Data Locality Optimizing Algorithm*. PLDI'91, pages 30 – 44, 1991.
5. Irigoin, F. and Triolet, R., *Supernode Partitioning*. POPL'88, pages 319 – 329, 1988.
6. Xue, J. *Loop Tiling for Parallelism*. Kluwer Academic Publishers. 2000.
7. Sanjay Rajopadhye, Tanguy Risset, and Claude Tadonki, *The algebraic path problem revisited*, European Conference on Parallel Computing Europar99, Toulouse (France), Lncs Sringer-Verlag, N 1685, p. 698-707, August 1999.
8. C. Bastoul, *Code generation in the polyhedral model is easier than you think*. In PACT'13 IEEE International Conference on Parallel Architecture and Compilation Techniques, pages 7-16, Juan-les-Pins, France, Sept. 2004.
9. L.-N. Pouchet, C. Bastoul, A. Cohen, and N. Vasilache, *Iterative optimization in the polyhedral model: Part I, one-dimensional time*. In ACM International Conference on Code Generation and Optimization (CGO'07), San Jose, California, pages 144-156, March 2007.
10. Sebastian Pop, Albert Cohen, Cedric Bastoul, Sylvain Girbal, P. Jouvelot, G.-A. Silber, and N. Vasilache. *GRAPHITE: Loop optimizations based on the polyhedral model for GCC*. In Proc. of the 4th GCC Developper's Summit, Ottawa, Canada, pages 179-198, June 2006.
11. Gautam, DaeGon Kim and Sanjay Rajopadhye. *Scheduling in the Z-Polyhedral Model*. In IPDPS07: International Parallel Distributed Programming Symposium, March 2007.
12. Gautam and Sanjay Rajopadhye. *The Z-Polyhedral Model*. In PPoPP07: Symposium on Principles and Practice of Parallel Programming, Mar 2007.
13. P. Quinton, S. Rajopadhye, T. Risset. *On Manipulating Z-polyhedra using a Canonical Representation*. Parallel Processing Letters, 1997.
14. *Data Layout Transformation for Enhancing Data Locality on NUCA Chip Multiprocessors*
`http://perso.ens-lyon.fr/christophe.alias/pub/pact2009.pdf`
15. Banerjee, Utpal, *Loop parallelization*, Kluwer Academic, 1994

...

Next (dependence analysis, parallelism, loop transformations)