A SYSTOLIC APPROACH
TO
LOOP PARTITIONING AND MAPPING
INTO
FIXED SIZE DISTRIBUTED MEMORY ARCHITECTURES

Ioannis Drositis, Nektarios Koziris, George Papakonstantinou and Panayotis Tsanakas

National Technical University of Athens
Department of Electrical and Computer Engineering
Division of Computer Science

Computing Systems Laboratory
`http://www.cslab.ece.ntua.gr`

Presentation Overview

- Loop Partitioning and Mapping - *The Systolic Approach*
- Some Terminology
- Communication Cost between Clusters
- The Main Procedure at a Glance
- Analyzing the Main Procedure
- Inductive Definition of h-length
- An Example
- Summarization
- Future Work

Loop Partitioning and Mapping *(the systolic approach)*

Example loop:

```
for i1 = 1 to 4 do
  for i2 = 1 to 3 do
    for i3 = 1 to 3 do
      (loop body)
    end i3
  end i2
end i1
```

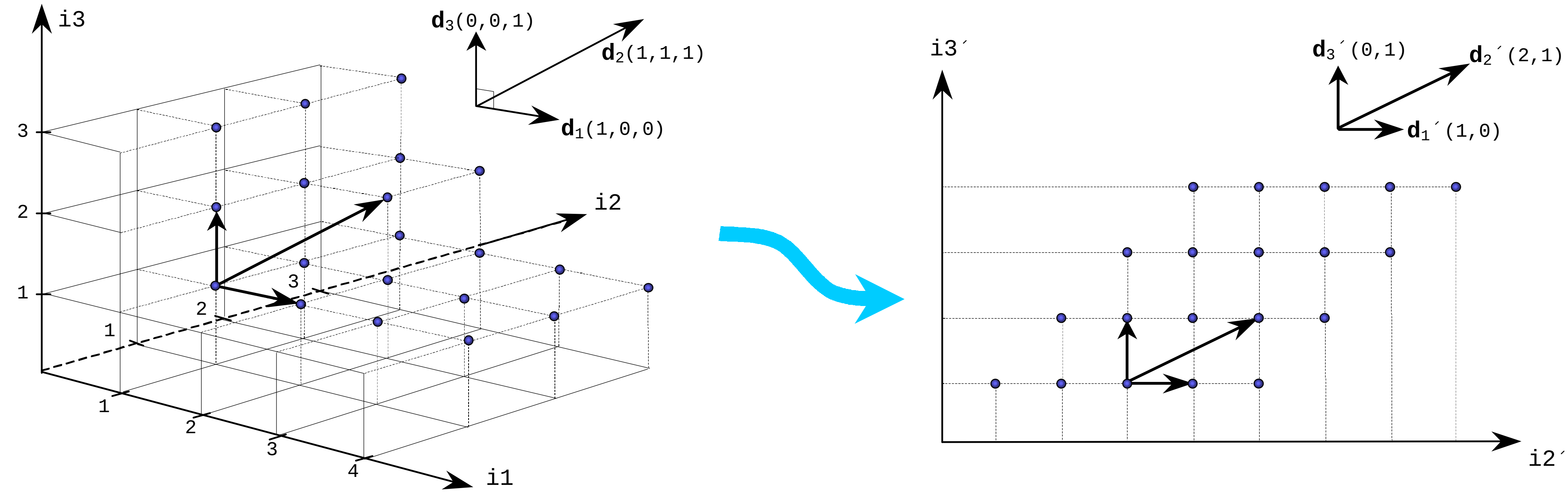
Through a linear transformation $T[n\times n]$:

$$T = \begin{bmatrix} \Pi \\ \hline S \end{bmatrix}, \text{ where } \Pi[1\times n] \text{ and } S[(n-1)\times n],$$

we obtain the array of virtual cells needed to compute the above (initial) index space.

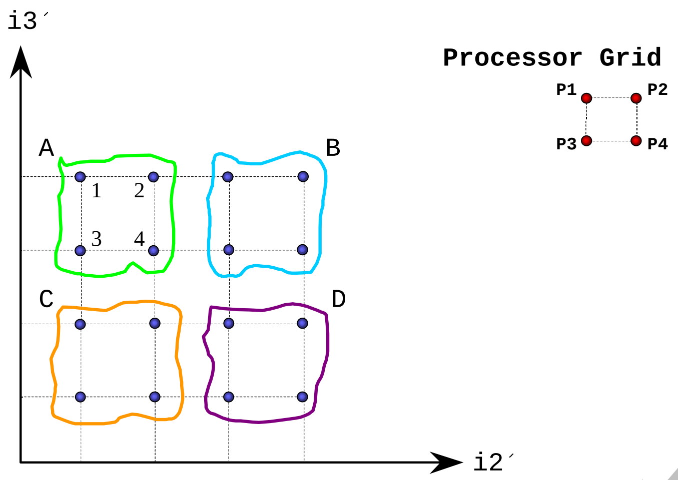
In other words:

$$(i2', i3')^T = S\cdot(i1, i2, i3)^T$$



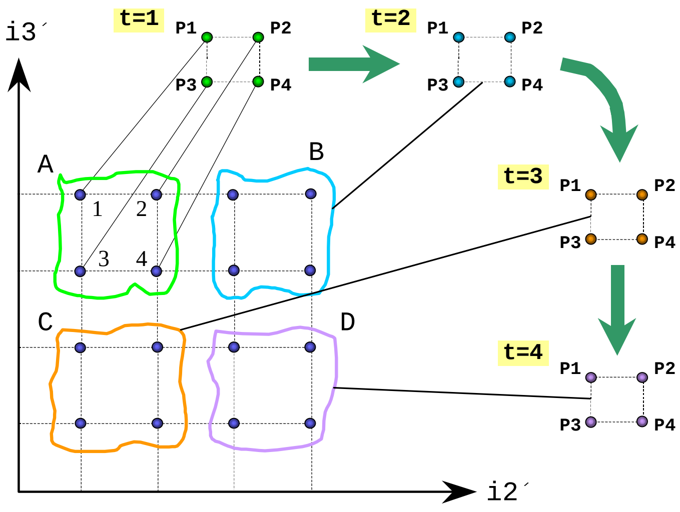
What needed to be done now: cutting the virtual space into ***clusters*** and assign each cluster to a different processor

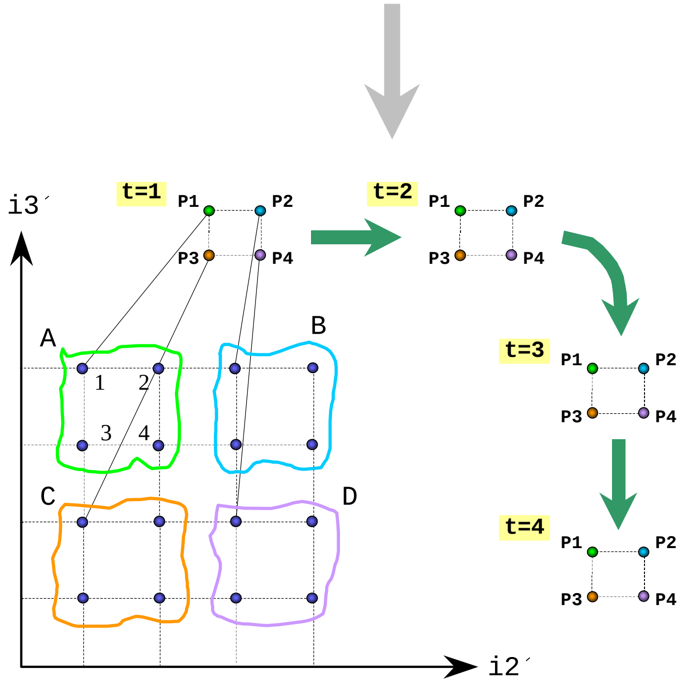
The Partitioning Method



Locally Parallel Globally Sequencial (LPGS)
where
cardinality of clusters = number of processors

Globally Parallel Locally Sequencial (GPLS)
where
number of clusters = number of processors



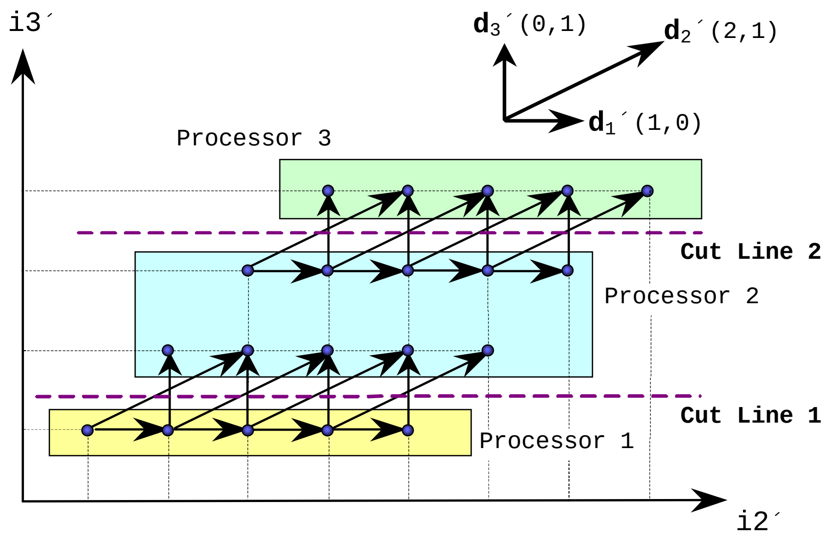


Cutting the Virtual Index Space: The consequences…

Available Processors: 3 $\rightarrow$ the Virtual (transformed) Space needs to be cut into 3 parts

First Attempt

Two horizontal lines, parallel to **horizontal boundary**

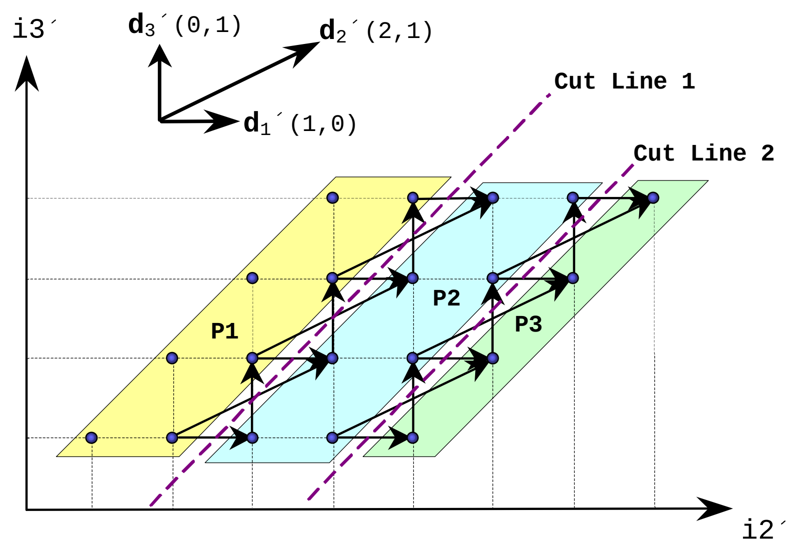


Result statistics:

- Communication cost = 8 + 8 = 16
- Processor utilization:
  Processor 1: 5 points
  Processor 2: 10 points
  Processor 3: 5 points

Second Attempt

Two lines, parallel to **side boundary**



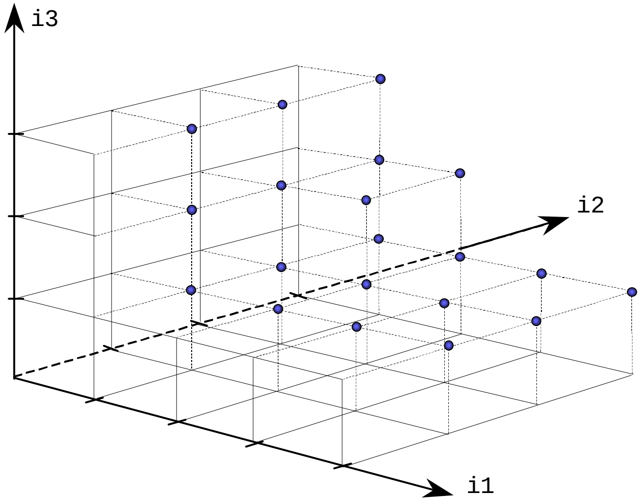
Result statistics:

- Communication cost = 10 + 10 = 20
- Processor utilization:
  Processor 1: 8 points
  Processor 2: 8 points
  Processor 3: 4 points

✓ Difference in ***communication cost*** as well as in ***processor utilization*** *'h-' stands for n-dimensional*

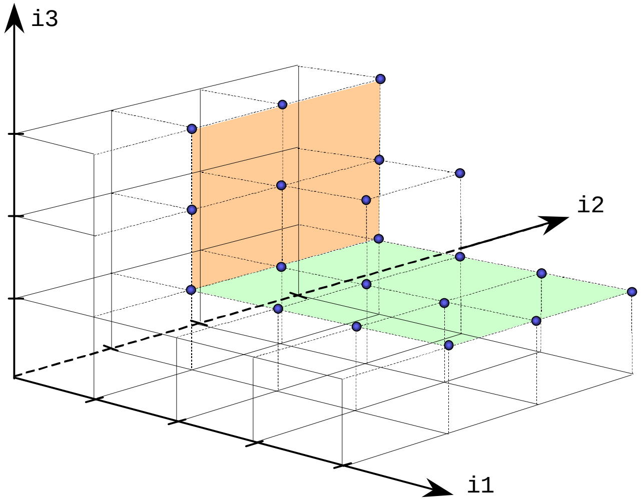
The h-terminology *(Part 1/2)*

- *h-space:* the *n*-dimensional space that corresponds to loop's indices (and depth)



For **n = 3**, a 3-dimensional (index) space is presented

- *h-plane:* a linear subspace of (*n*-1)-dimension *(a plane in the 3-dimensional space)*

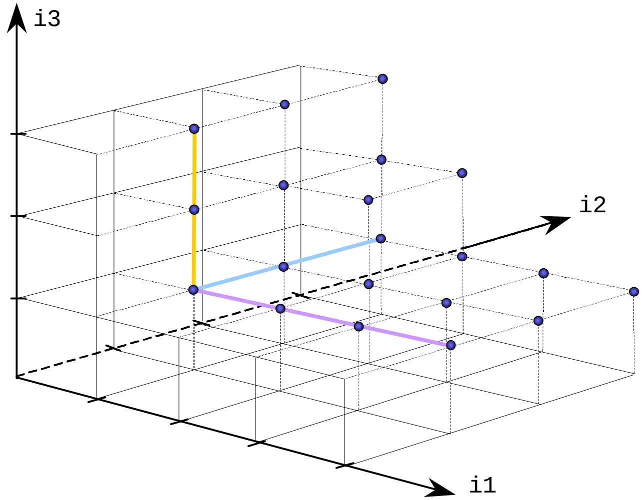


For **n = 3**, two 2-dimensional h-planes are presented here, the one perpendicular to the other

'h-' stands for n-dimensional

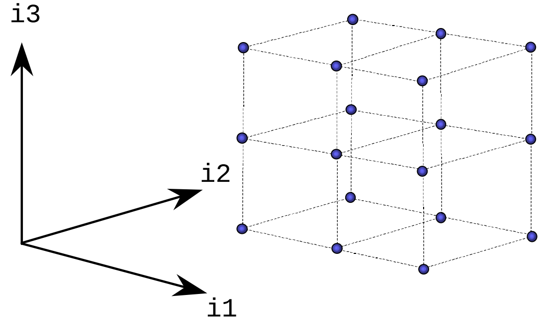
The h-terminology *(Part 2/2)*

- *h-line:* a linear subspace of (n-2)-dimension
  (a line in the 3-dimensional space)



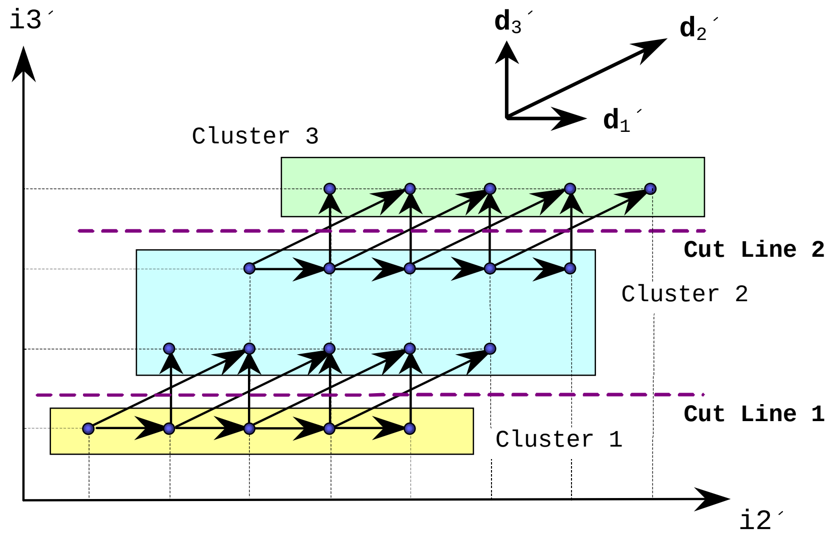
For **n = 3**, three 1-dimensional h-lines are presented, each one perpendicular to other two

- *h-mesh:* a mesh (of processors usually) in the (n-1)-dimensional space
  (an array of cells connected in a mesh topology)



For **n = 3**, a 3-dimensional mesh (3×2×3) of processors is presented

Communication Costs between Clusters *(Introduction)*



CUT

cost of a cut = { *number of transformed dependence vectors that traverse the cut's h-line* }
= { *density of dependence vectors* } × { *length of cut* }

MAPPING

cost of a mapping = ∑ { *cost values of its individual cuts* }

So:

cost of a single cut = { *length of the cut* } × { *overall density (of all dependence vectors) at the direction that is perpendicular to the cut* }

or:

cost of a single cut = { *length of the cut* } × ∑ { *density of each dependence vector on the specified direction* }

Communication Cost between Clusters *(continuing…)*

COST OF A SINGLE CUT

cost of a single cut = { *length of the cut* } × ∑ { *density of each dependence vector on the specified direction* }

$$\text{cost of a single cut: } c = l \cdot \sum_{i=1}^{m} \frac{|\mathbf{p} \cdot \mathbf{d}'_i|}{\|\mathbf{p}\|}$$

where:

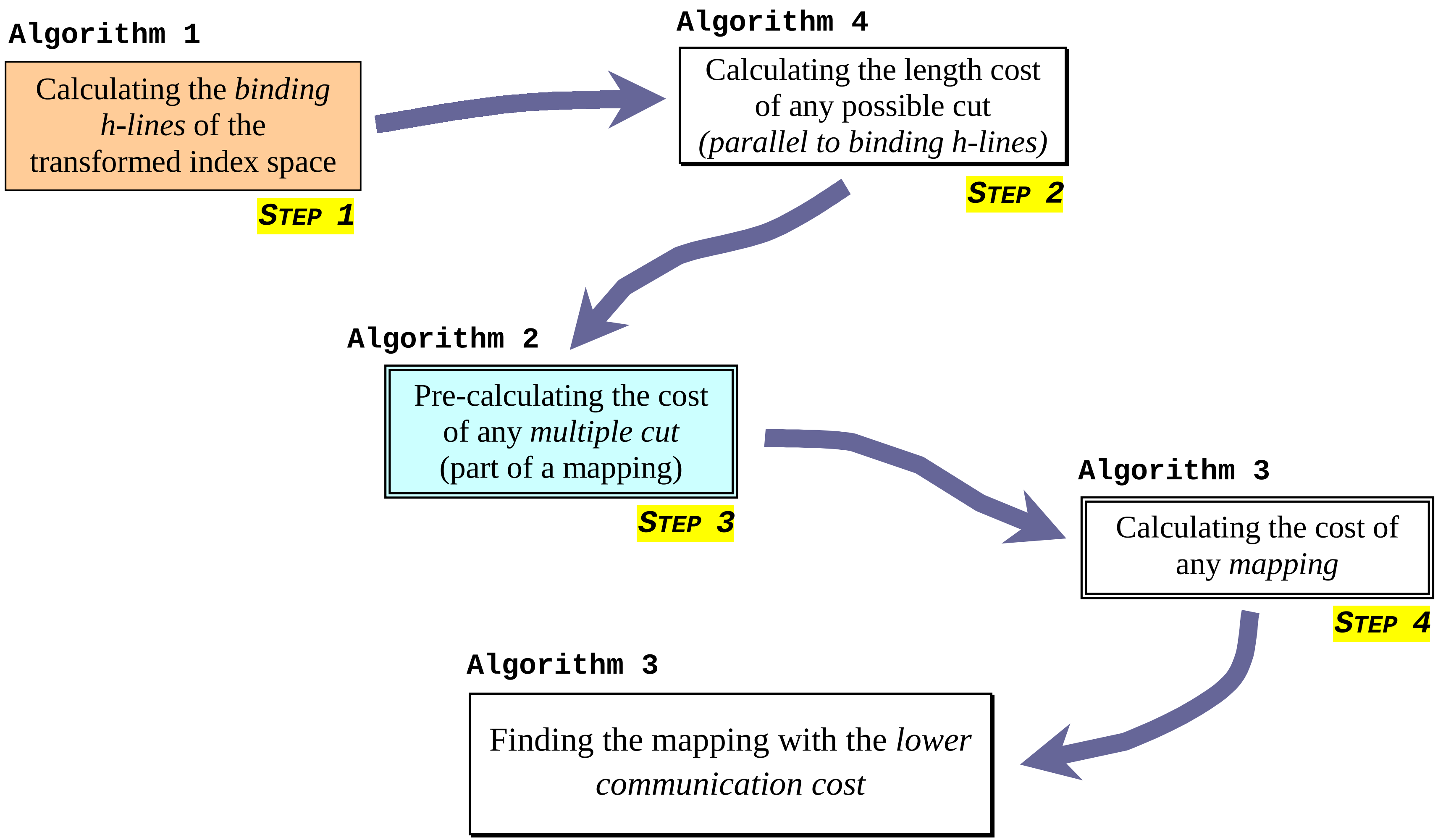
- m is the number of distinct dependence vectors, $\mathbf{p}$ is the vector that is perpendicular to the cut,
- $\mathbf{d}'_i$ is a single transformed dependence vector, $\|\mathbf{u}\|$ is the Euclidean norm of vector $\mathbf{u}$,
- l is the h-length of the segment of the h-line that corresponds to the cut and is *within the bounds* of the transformed h-space.

COST OF A MAPPING

cost of a mapping = { *sum of costs of all cuts that comprise the mapping* }

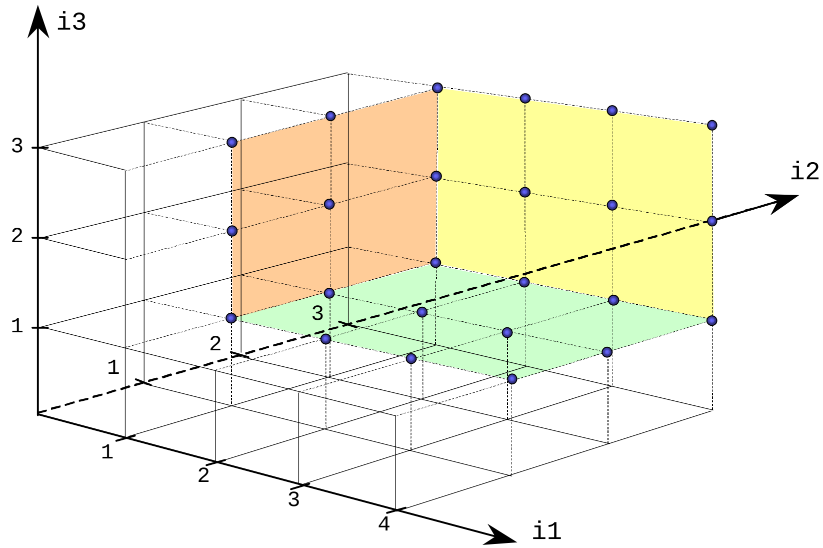
$$\text{cost of a mapping} = \sum_{\text{for all cuts}} \{\text{cost of a single cut}\} = \sum_{\text{for every cut } k} \left\{ l_k \cdot \sum_{i=1}^{m} \frac{|\mathbf{p} \cdot \mathbf{d}'_i|}{\|\mathbf{p}\|} \right\}$$

The Procedure at a Glance



Analyzing the Procedure *(Part 1/4)*

```
For i1 = 1 to 4 do
   for i2 = 1 to 3 do
     for i3 = 1 to 3 do
       (loop body)
     end i3
   end i2
 end i1
```



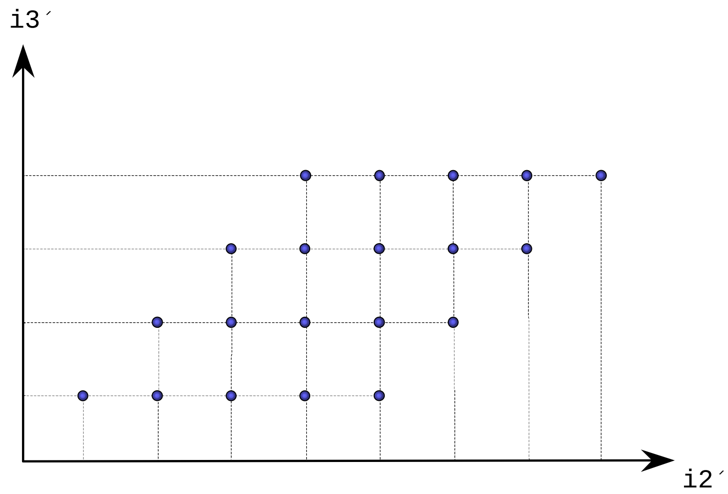
Boundary points in n-dimensional index space

Algorithm 1

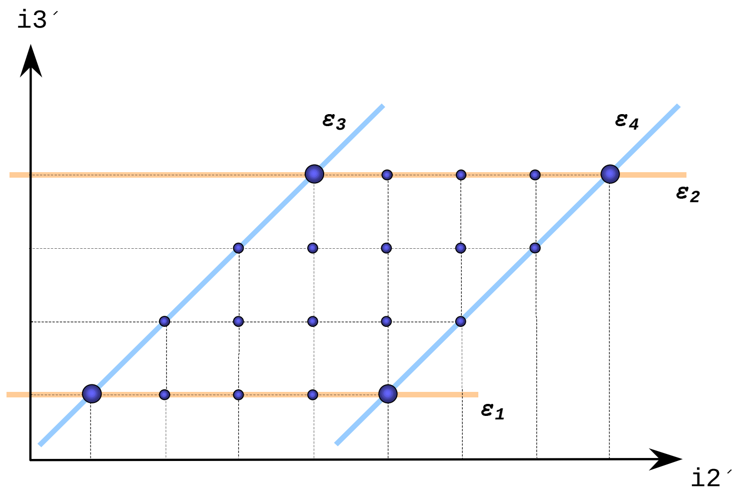
Calculate the *binding h-lines* of the transformed index space

Determining possible cut directions

Find transformed points and calculate the convex hull of them;



from the convex hull boundaries, calculate virtual space's binding h-lines.



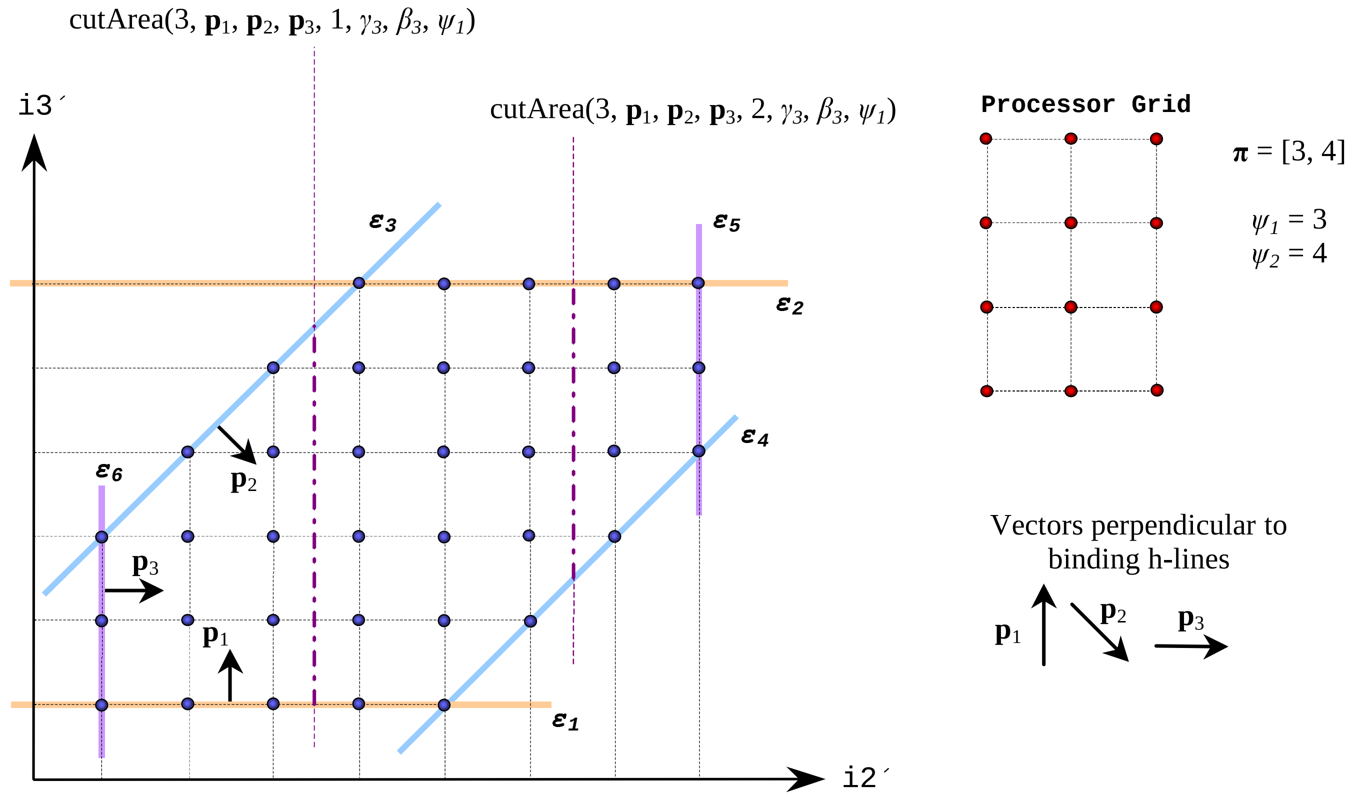
Analyzing the Procedure *(Part 2/4)*

Algorithm 4

> Calculate the length cost of any possible cut *(parallel to binding h-lines)*

Implemented by function *cutArea()*:

$$cutArea(i, \mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_b, k, \gamma_i, \beta_i, \psi_j)$$



Analyzing the Procedure *(Part 3a/4)*

Algorithm 2

Pre-calculate the cost of any *multiple cut* (part of a mapping)

A. Evaluate *depCost*$_i$, which is the overall dependence vector density along direction of binding h-line pair *i*.

B. Call several times *cutArea()* function with ***properly*** specified parameters:

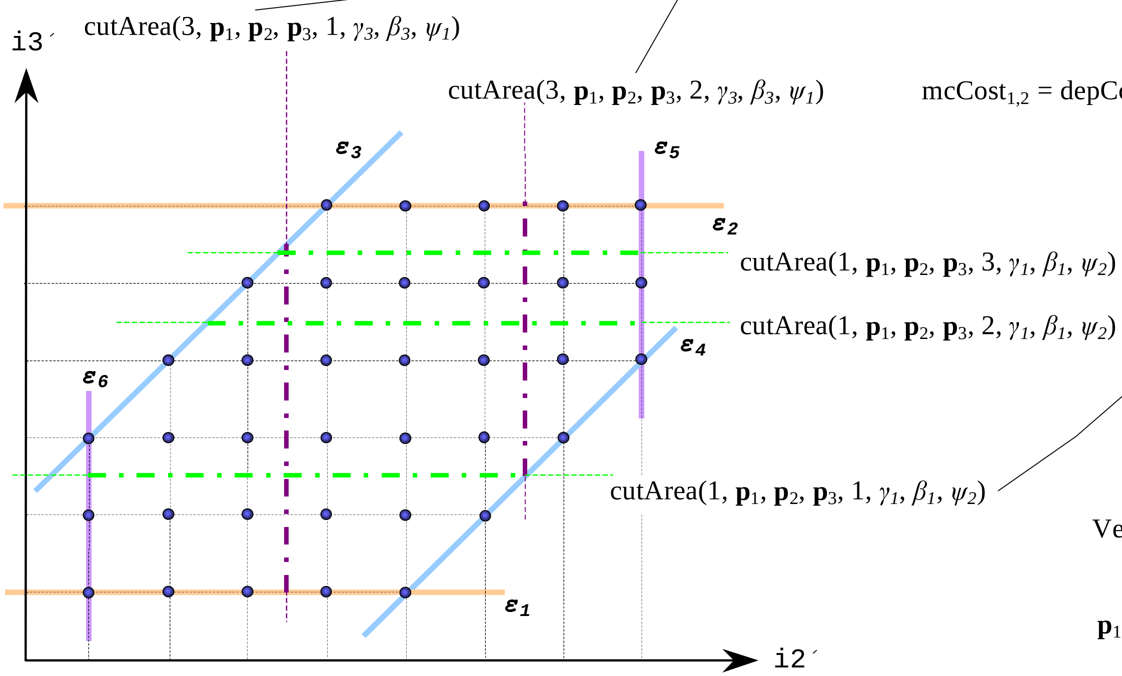
- **for all** pairs of binding h-lines (1)
- **for all** combinations of processor-grid arrangement (2)

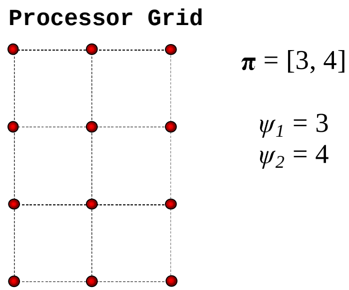
What we do in this step

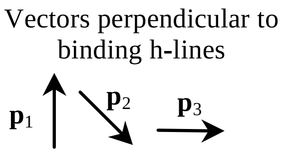
We computes multiple-cut cost for every multiple-cut possible *(by lines parallel to binding h-lines)*

$$mcCost_{3,1} = depCost_3 \times \{cutArea(3, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, 1, \gamma_3, \beta_3, \psi_1) + cutArea(3, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, 2, \gamma_3, \beta_3, \psi_1)\}$$

$$mcCost_{1,2} = depCost_1 \times \{cutArea(1, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, 1, \gamma_1, \beta_1, \psi_2) + cutArea(1, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, 2, \gamma_1, \beta_1, \psi_2) + cutArea(1, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, 3, \gamma_1, \beta_1, \psi_2)\}$$







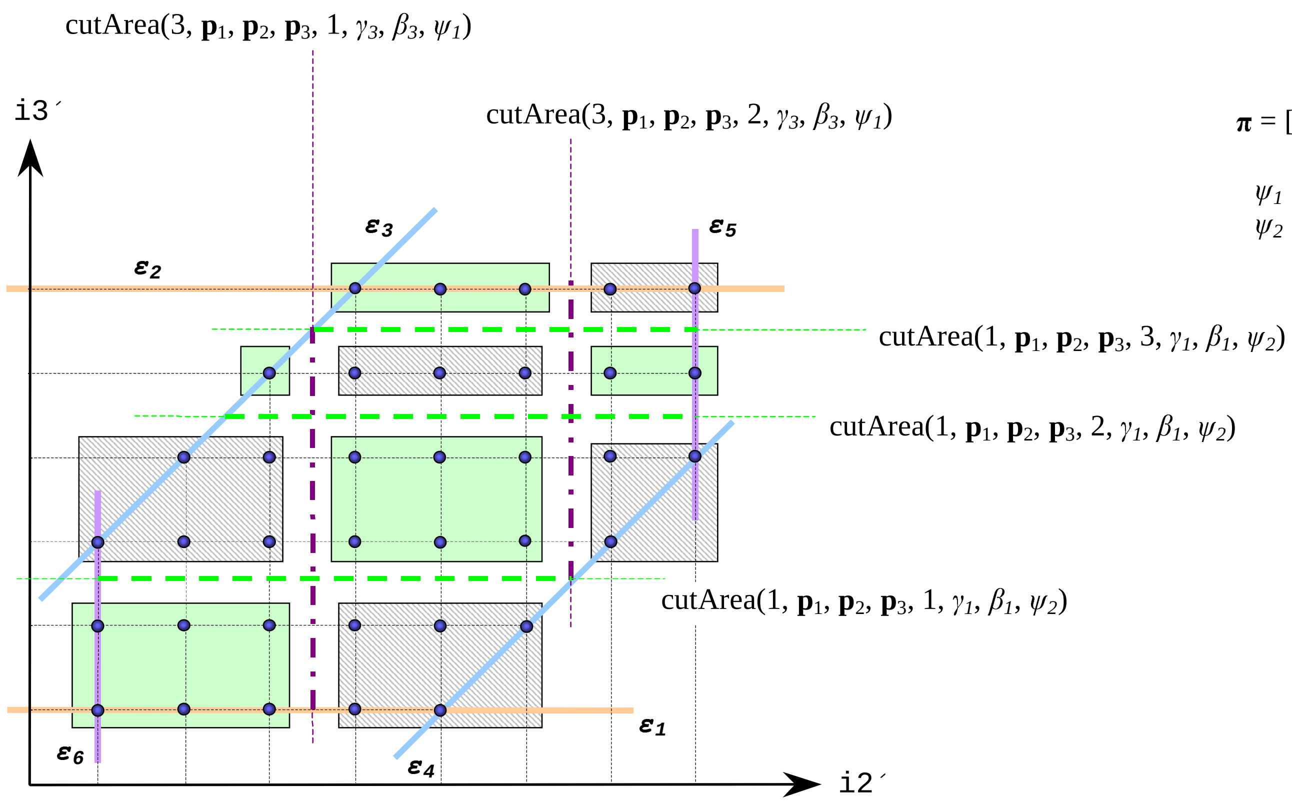
Analyzing the Procedure *(Part 3b/4)*

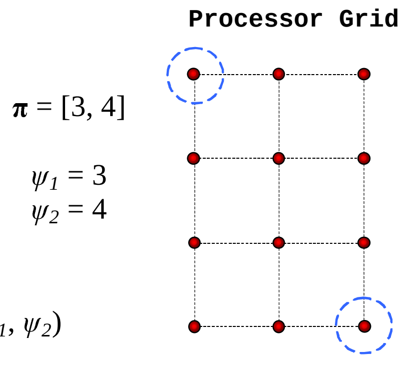
Algorithm 2

Pre-calculate the cost of any *multiple cut* (part of a mapping)

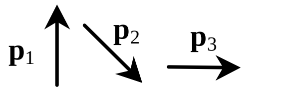
CLUSTERING #1

Cutting lines: **a.** parallel to binding h-line pairs 3 (lines ε_5 and ε_6) and 1 (lines ε_1 and ε_2) and
b. using three processors along first pair (grid 1st dimension) and four processors along second pair.





Vectors perpendicular to binding h-lines



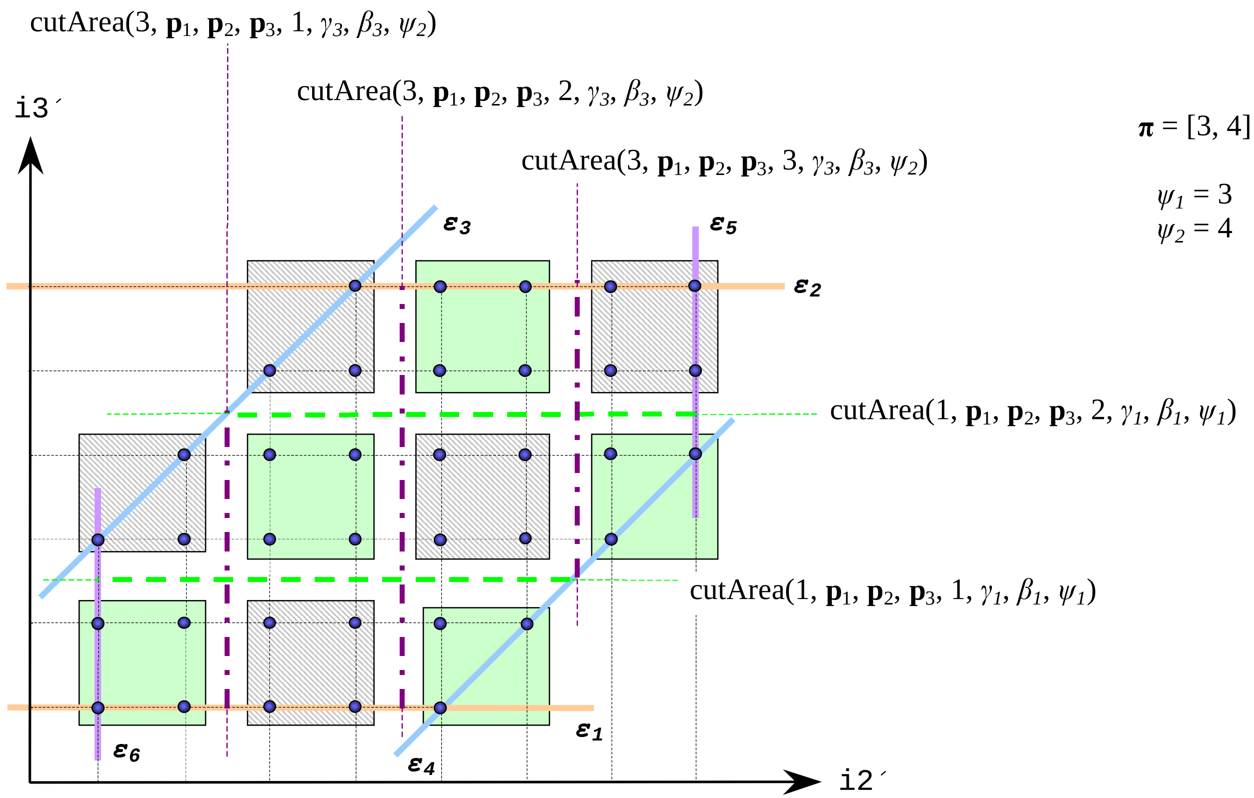
Analyzing the Procedure *(Part 3c/4)*

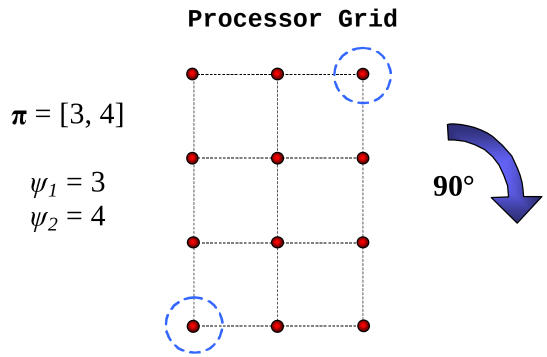
Algorithm 2

Pre-calculate the cost of any *multiple cut* (part of a mapping)

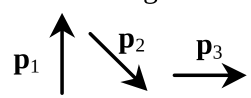
CLUSTERING #2

Cutting lines: **a.** parallel to binding h-line pairs 3 (lines ε_5 and ε_6) and 1 (lines ε_1 and ε_2) and
b. using *four* processors along first pair (grid 2nd dimension) and *three* processors along second pair.





Vectors perpendicular to binding h-lines



Analyzing the Procedure *(Part 4/4)*

Algorithm 3

Calculate the cost of any *mapping*

and

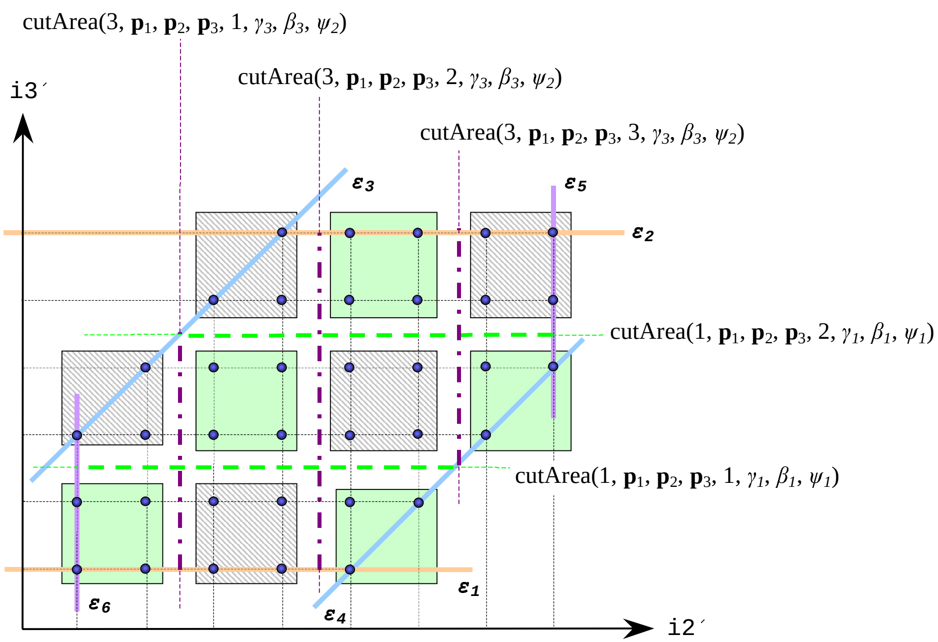
Find the mapping with the *lower communication cost*

For any valid mapping, find the mapping cost, by summing all multiple-cut costs that comprise the mapping and keep track of the lower cost.

Mapping #1

$$cost = mcCost_{3,2} + mcCost_{1,1}$$

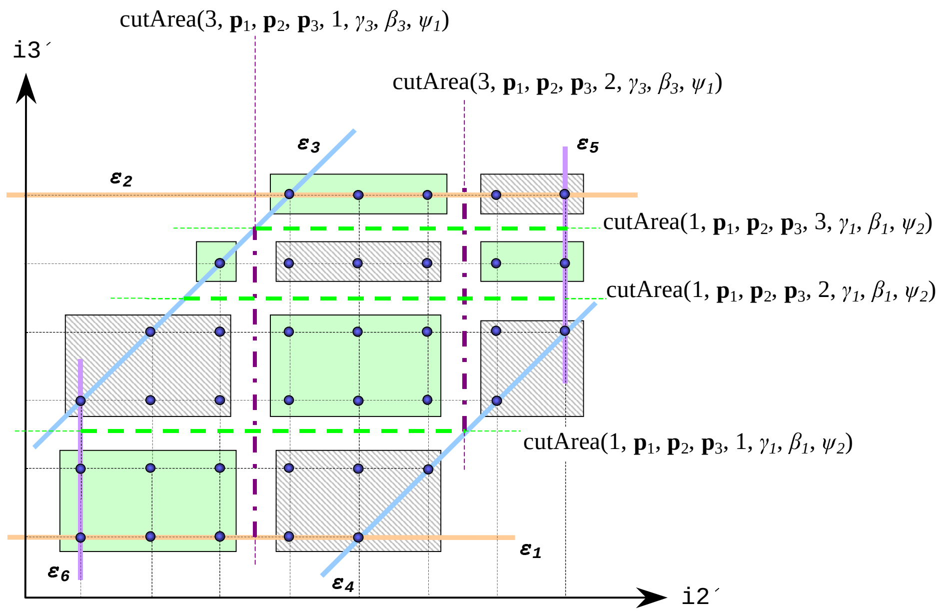
$$cost = depCost_3 \times \{cutArea(3, \ldots, 1, \ldots, \psi_2) + cutArea(3, \ldots, 2, \ldots, \psi_2) + cutArea(3, \ldots, 3, \ldots, \psi_2)\} + depCost_1 \times \{cutArea(1, \ldots, 1, \ldots, \psi_1) + cutArea(1, \ldots, 2, \ldots, \psi_1)\}$$



Mapping #2

$$cost = mcCost_{3,1} + mcCost_{1,2}$$

$$cost = depCost_3 \times \{cutArea(3, \ldots, 1, \ldots, \psi_1) + cutArea(3, \ldots, 2, \ldots, \psi_1)\} + depCost_1 \times \{cutArea(1, \ldots, 1, \ldots, \psi_2) + cutArea(1, \ldots, 2, \ldots, \psi_2) + cutArea(1, \ldots, 3, \ldots, \psi_2)\}$$



Inductive Definition of h-length

Algorithm 5

Polygon triangulation to calculate its area

For $n = 3$, use Euclidean distance

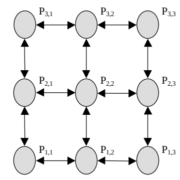
For $n > 3$:

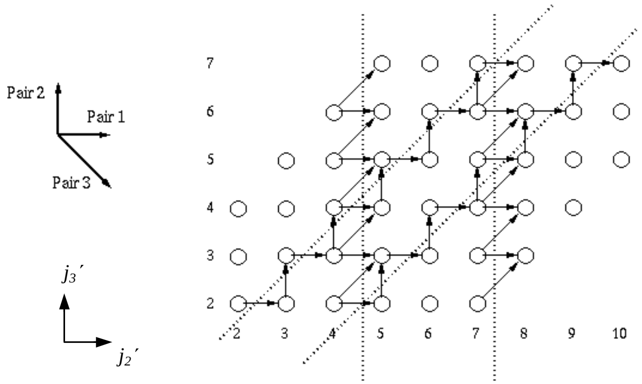
- exclude one point ***u*** arbitrarily
- use the same algorithm to calculate the h-length l' of the h-line segment that is defined by the remaining n-1 points, in an h-space of dimension n-2
- find the projection ***u*′** of ***u*** on the h-plane defined by the remaining n-1 points
- calculate the Euclidean distance d between ***u*** and ***u*′**; the result is the product of l and d.

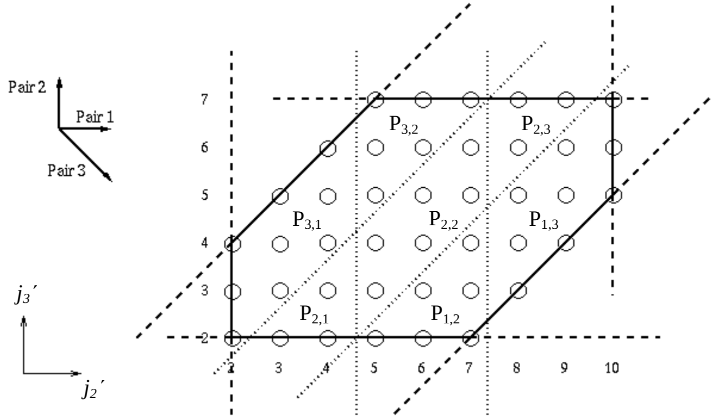
An Example

```
for i1 = 1 to 6 do
  for i2 = 1 to 4 do
    for i3 = 1 to 3 do
      a(i1,i2,i3) = a(i1,i2-1,i3) + a(i1-1,i2,i3) + a(i1,i2,i3-1)
    end i3
  end i2
end i1
```

$$D = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad T = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$







For this problem, optimal transformation methods for systolic arrays produce matrices:

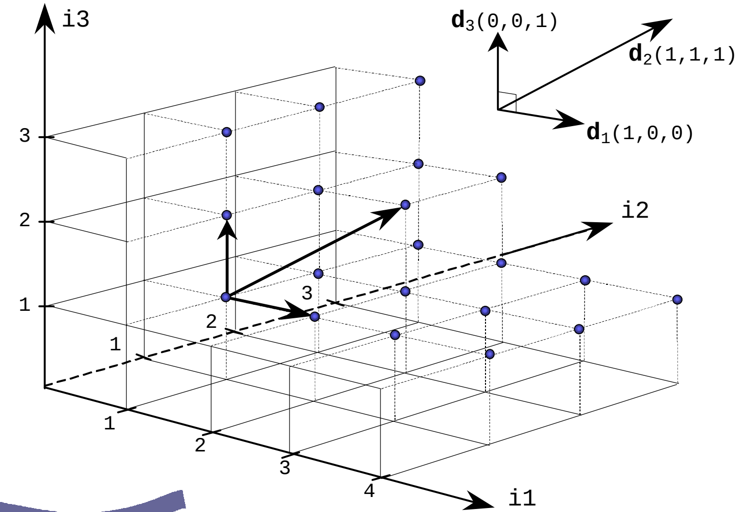
$$T_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}, T_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}, T_3 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}, T_4 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

These matrices result in systolic arrays of 42, 24, 12 and 12 cells respectively.

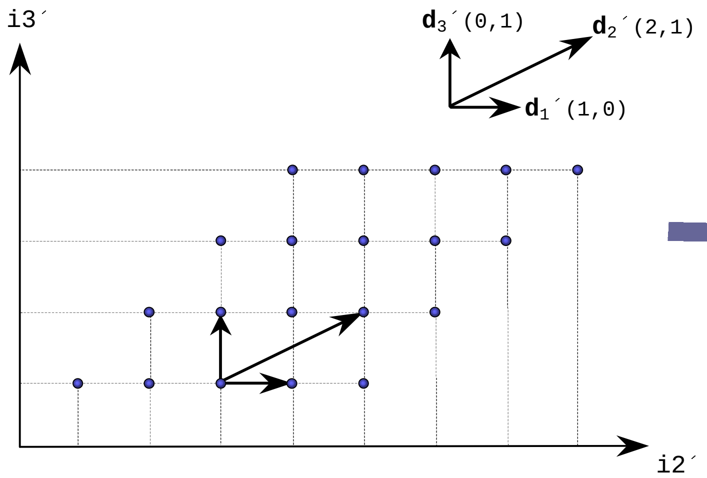
Summarization

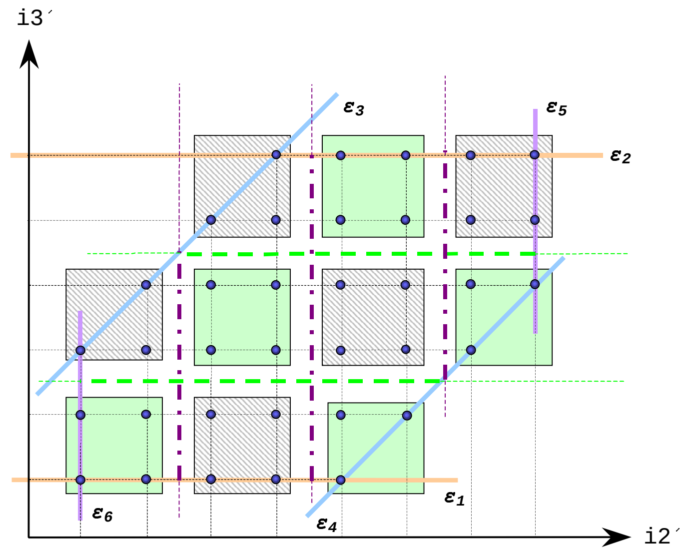
a FOR loop

```
for i1 = 1 to 4 do
  for i2 = 1 to 3 do
    for i3 = 1 to 3 do
      (loop body)
    end i3
  end i2
end i1
```



The method presented: finds the ***lower cost mapping*** for a given processor grid, using cuts that are *parallel* to virtual space *boundaries*





Future Work

- Intra-processor scheduling

  Mapping that different points correspond to the same time instance and same processor.

  How they are executed?