An Irredundant Decomposition of Data Flow with Affine Dependences

Corentin Ferry – Colorado State University & Univ Rennes, CNRS, Inria, IRISA
Steven Derrien – Univ Rennes, CNRS, Inria, IRISA
Sanjay Rajopadhye – Colorado State University
IMPACT 2024 – January 17, 2024 – Munich, Germany

Outline

Context

- HPC Applications: Large volumes of data, low number of operations
  → **I/O intensive, low operational intensity**
- **Memory communication** is extremely **expensive** (time + energy)
- **Sub-optimal utilization** of memory
  - Too many accesses
  - Limited contiguity
- Polyhedral compilation → locality-improving techniques
  - Mostly temporal locality (e.g., tiling – lots of work e.g. [1, 2])
- Spatial locality → (re-)allocate data, change access schedule

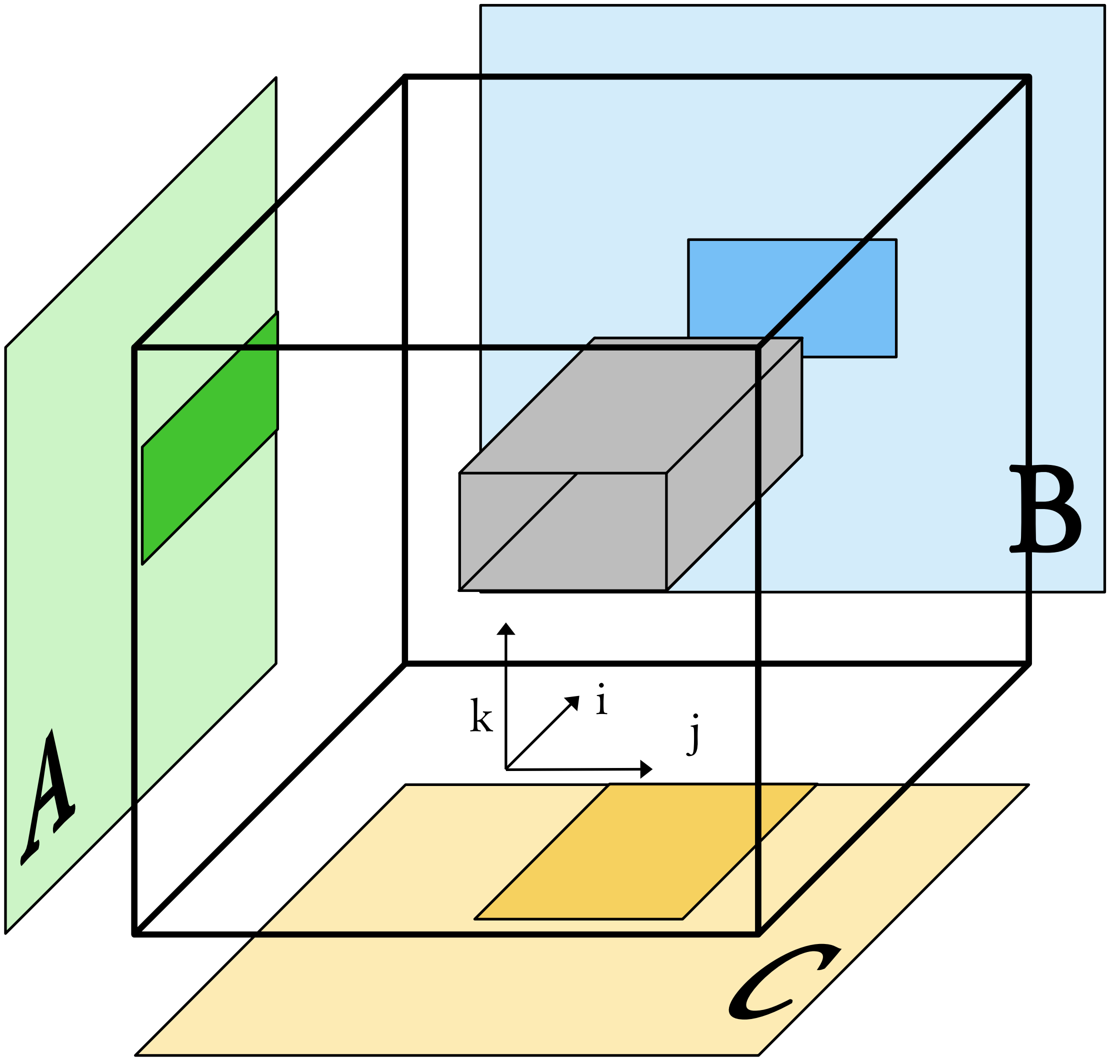
Exploit polyhedral model's information to get spatial locality?

[1] A. Agarwal, D.A. Kranz, and V. Natarajan. 1995. Automatic Partitioning of Parallel Loops and Data Arrays for Distributed Shared-Memory Multiprocessors. IEEE Transactions on Parallel and Distributed Systems 6, 9 (1995), 943–962.

[2] Jie Zhao and Peng Di. 2020. Optimizing the Memory Hierarchy by Compositing Automatic Transformations on Computations and Data. In 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE.

Tiling as a locality optimization

Matrix-Matrix Product (C += A * B)



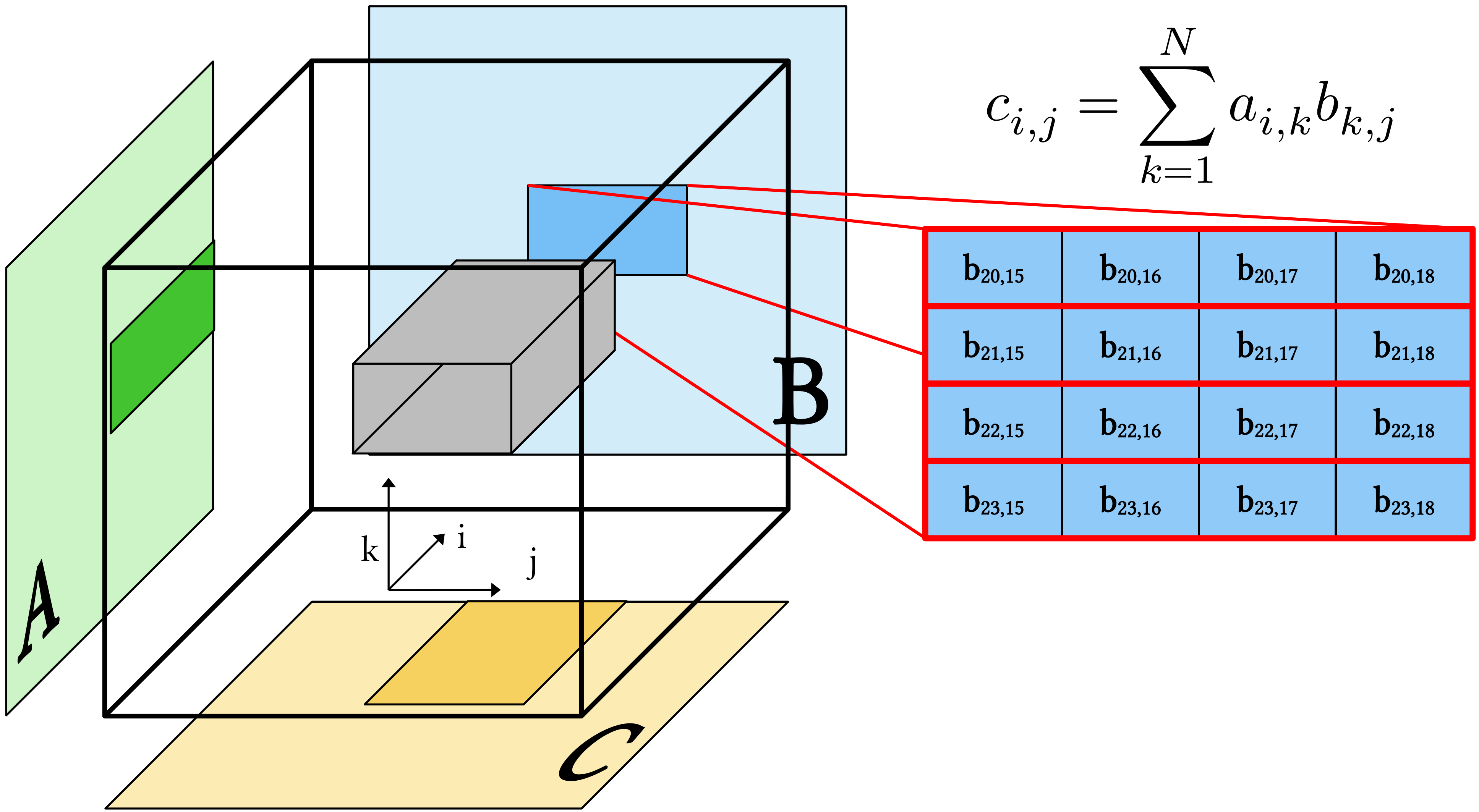
$$c_{i,j} = \sum_{k=1}^{N} a_{i,k} b_{k,j}$$

Tiling improves reuse
(temporal locality)
e.g. patch of A reused
in a row of tiles

Tiling at the expense of spatial locality...

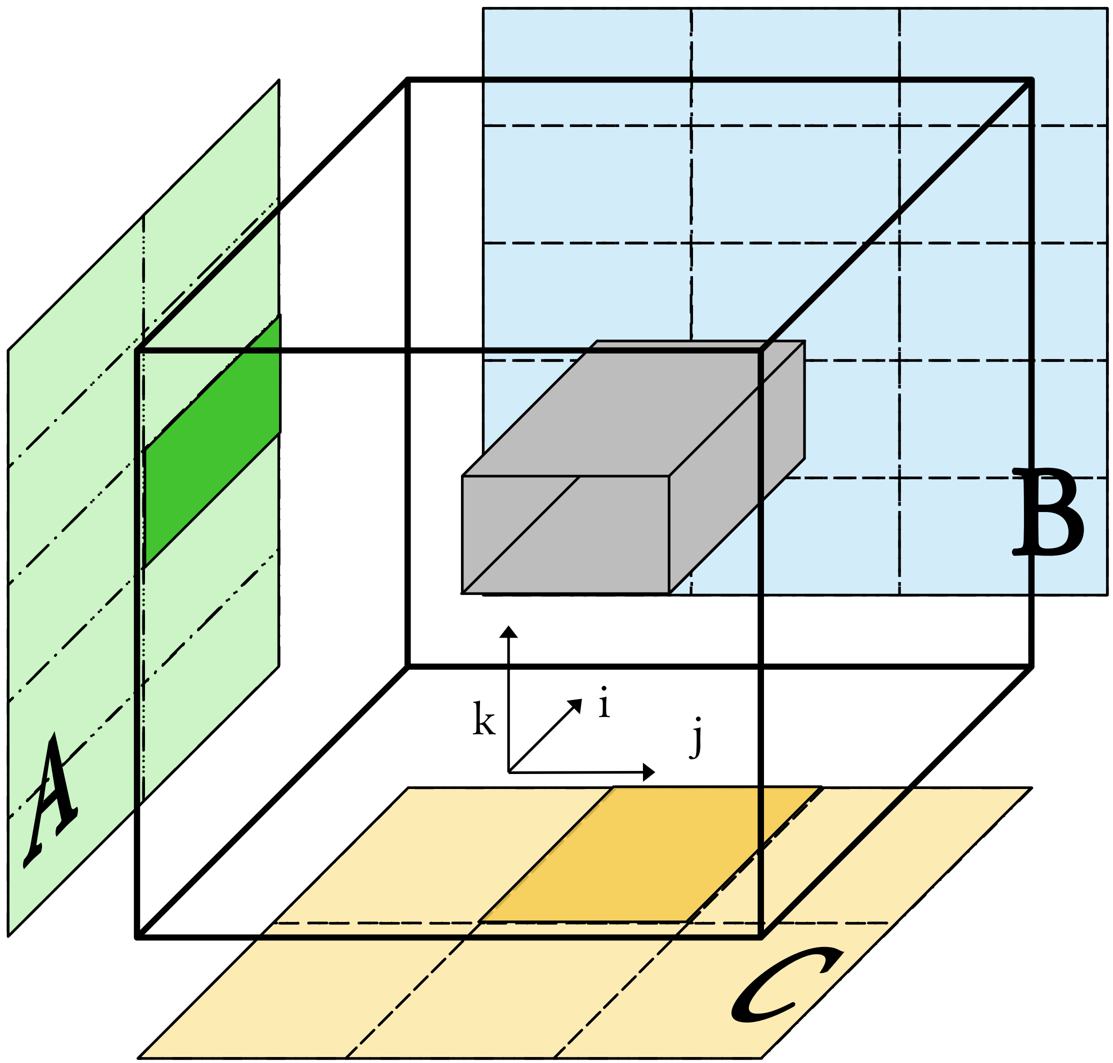
Patches are not contiguous blocks of data

$$c_{i,j} = \sum_{k=1}^{N} a_{i,k} b_{k,j}$$



Data Tiling for Spatial Locality

All data within a patch is now contiguous

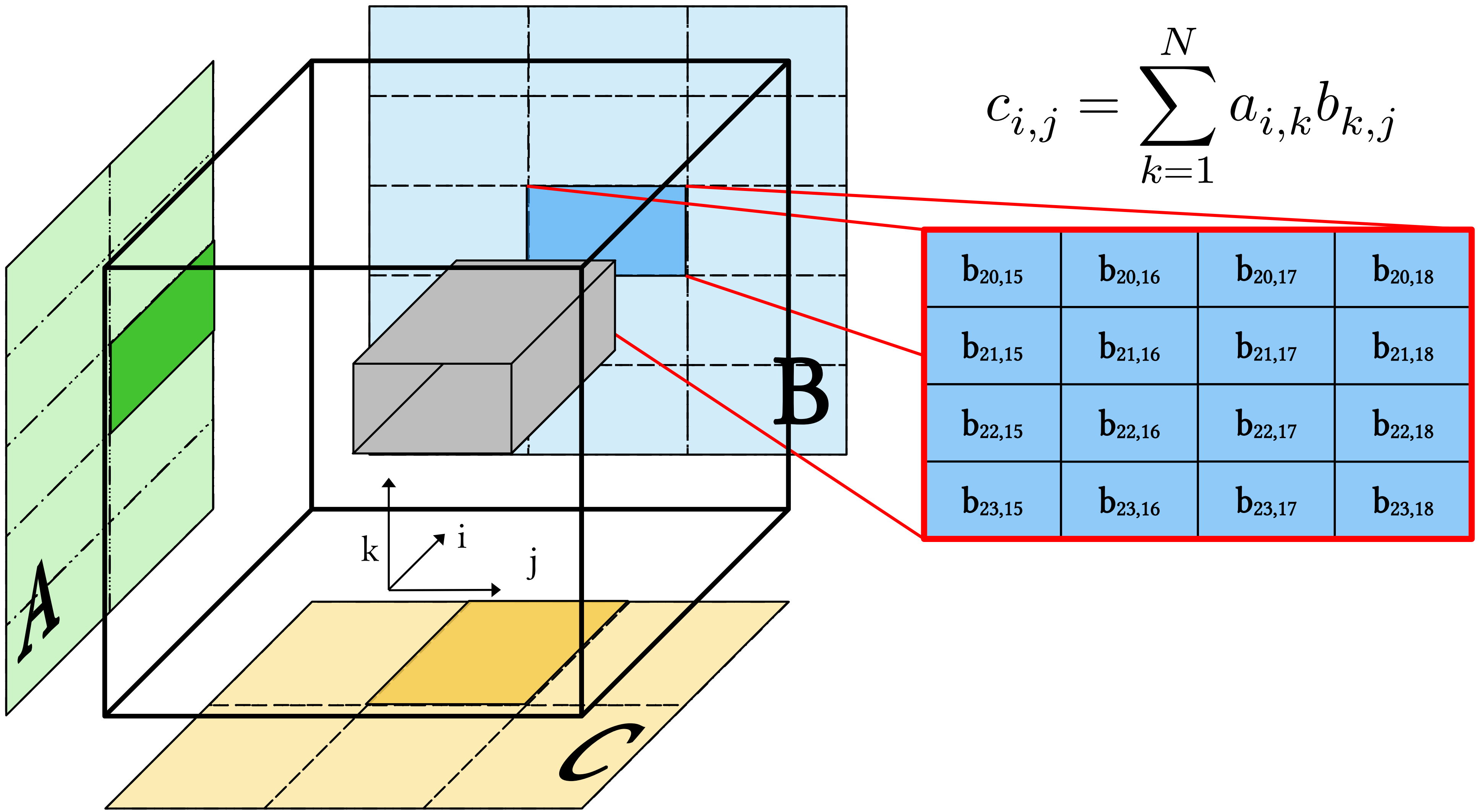


$$c_{i,j} = \sum_{k=1}^{N} a_{i,k} b_{k,j}$$

Data Tiling for Spatial Locality

All data within a patch is now contiguous

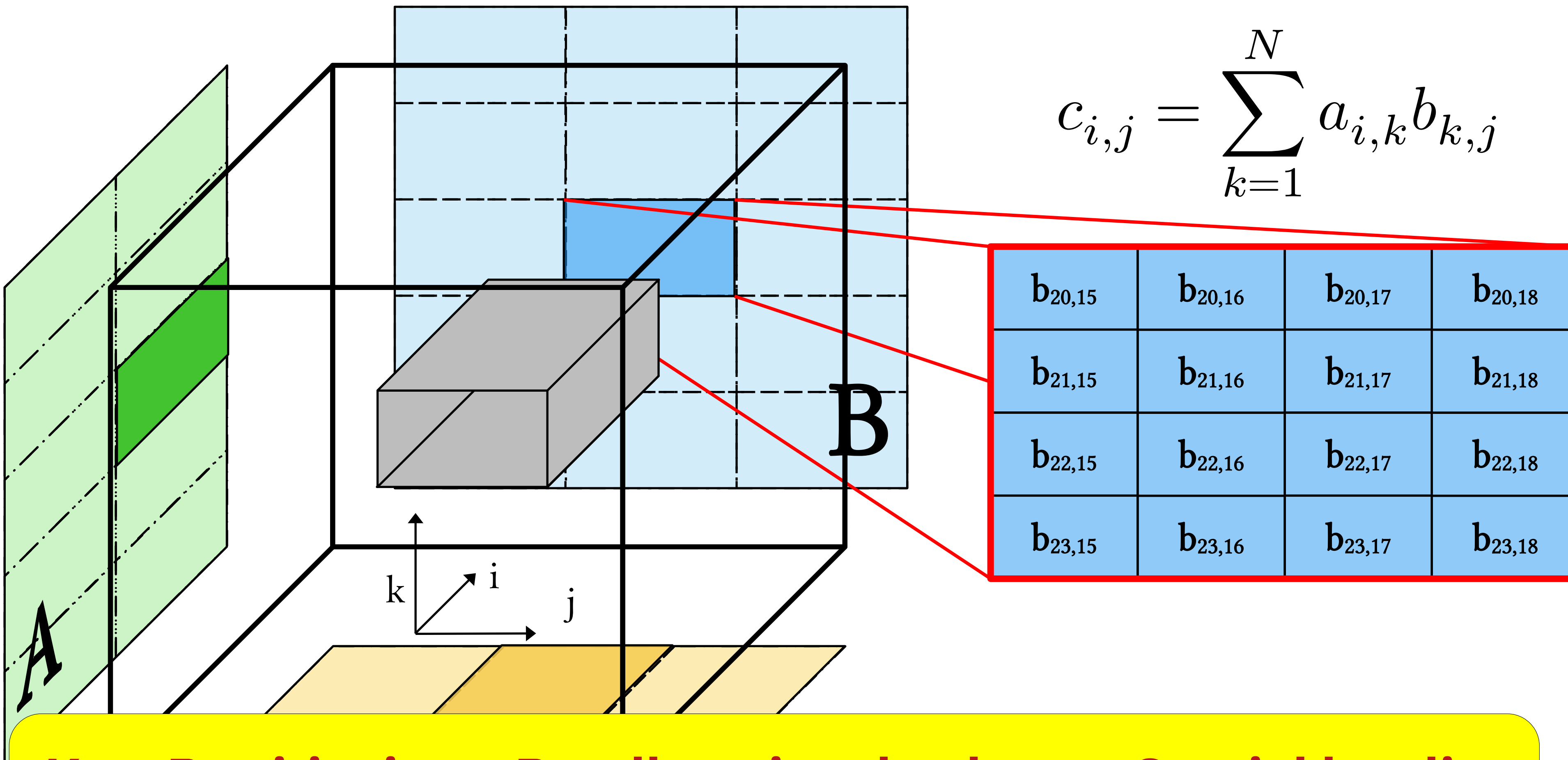
$$c_{i,j} = \sum_{k=1}^{N} a_{i,k} b_{k,j}$$



Data Tiling for Spatial Locality

All data within a patch is now contiguous

$$c_{i,j} = \sum_{k=1}^{N} a_{i,k} b_{k,j}$$



Key: Partitioning + Re-allocating the data → Spatial locality
Similar partitioning data with arbitrary footprints?

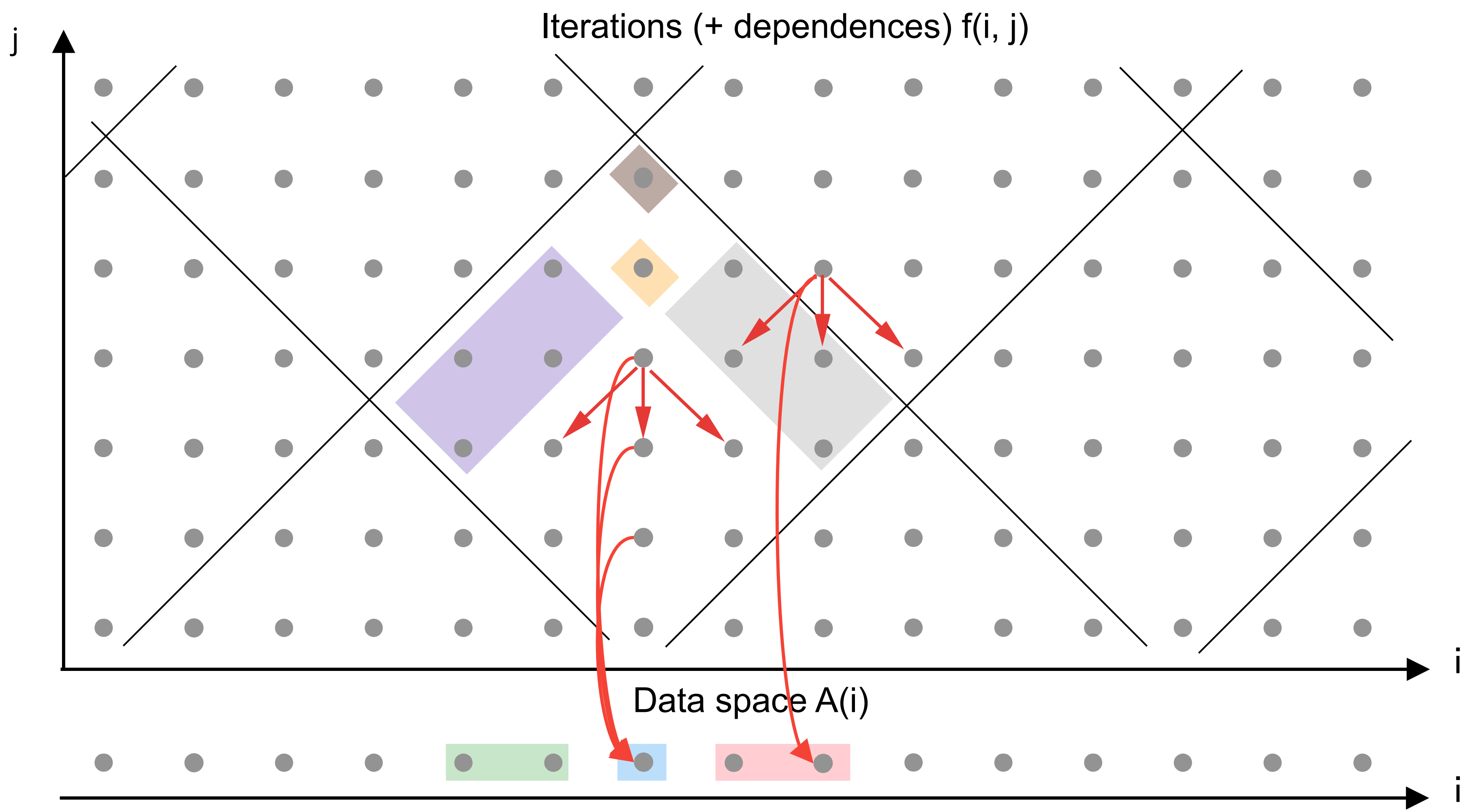
Outline

1. **Introduction**
    1. Context
    2. Loop Tiling as a Locality Optimization
    3. MARS : Partitioning Data for Spatial Locality
2. **Partitioning Data with Affine Dependences**
    1. Single affine dependence
    2. Multiple uniformly intersecting dependences
    3. Multiple non-uniformly intersecting dependences
    4. Dependences to tiled iteration spaces
3. **Conclusions**

Our Idea : Use the Polyhedral Model

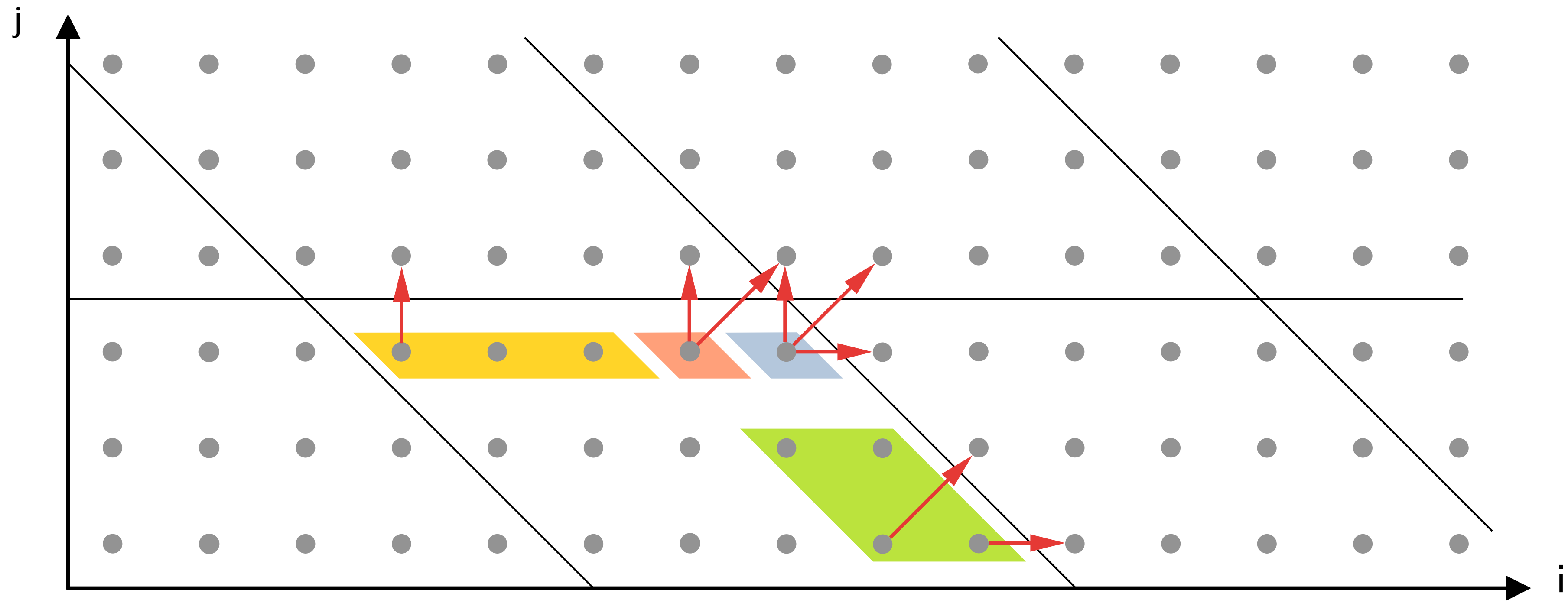
Partition iterations & data with dependence / data flow info

ex. f(i, j) = f(i-1, j-1) + f(i, j-1) + f(i+1, j-1) + A(i)



MARS : Iteration Space Partitioning

Largest sets of iterations within a tile (*results*) that
have the exact same consumer tiles.

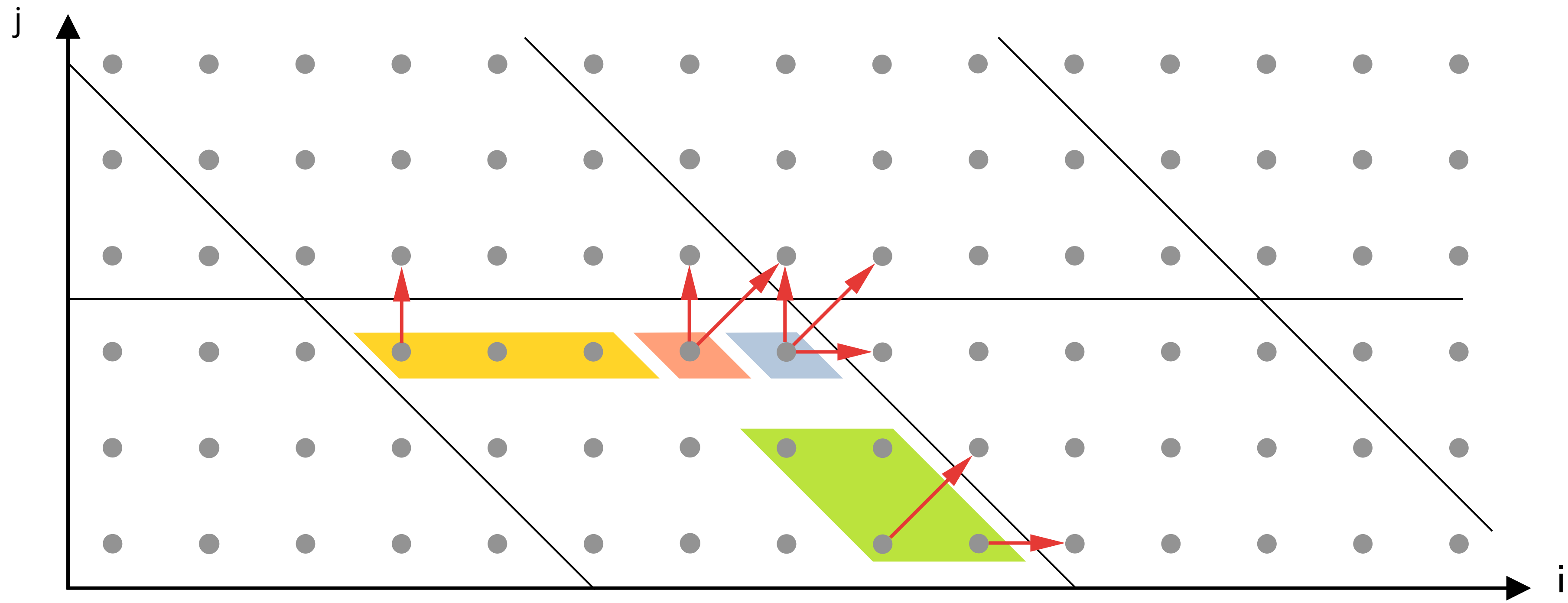


Corentin Ferry, Steven Derrien, and Sanjay Rajopadhye. 2023. Maximal Atomic irRedundant Sets: a Usage-based Dataflow Partitioning Algorithm. In 13th International Workshop on Polyhedral Compilation Techniques (IMPACT'23).

MARS : Iteration Space Partitioning

Largest sets of iterations within a tile (*results*) that

have the exact same consumer tiles.



MARS → Partitioning intermediate results

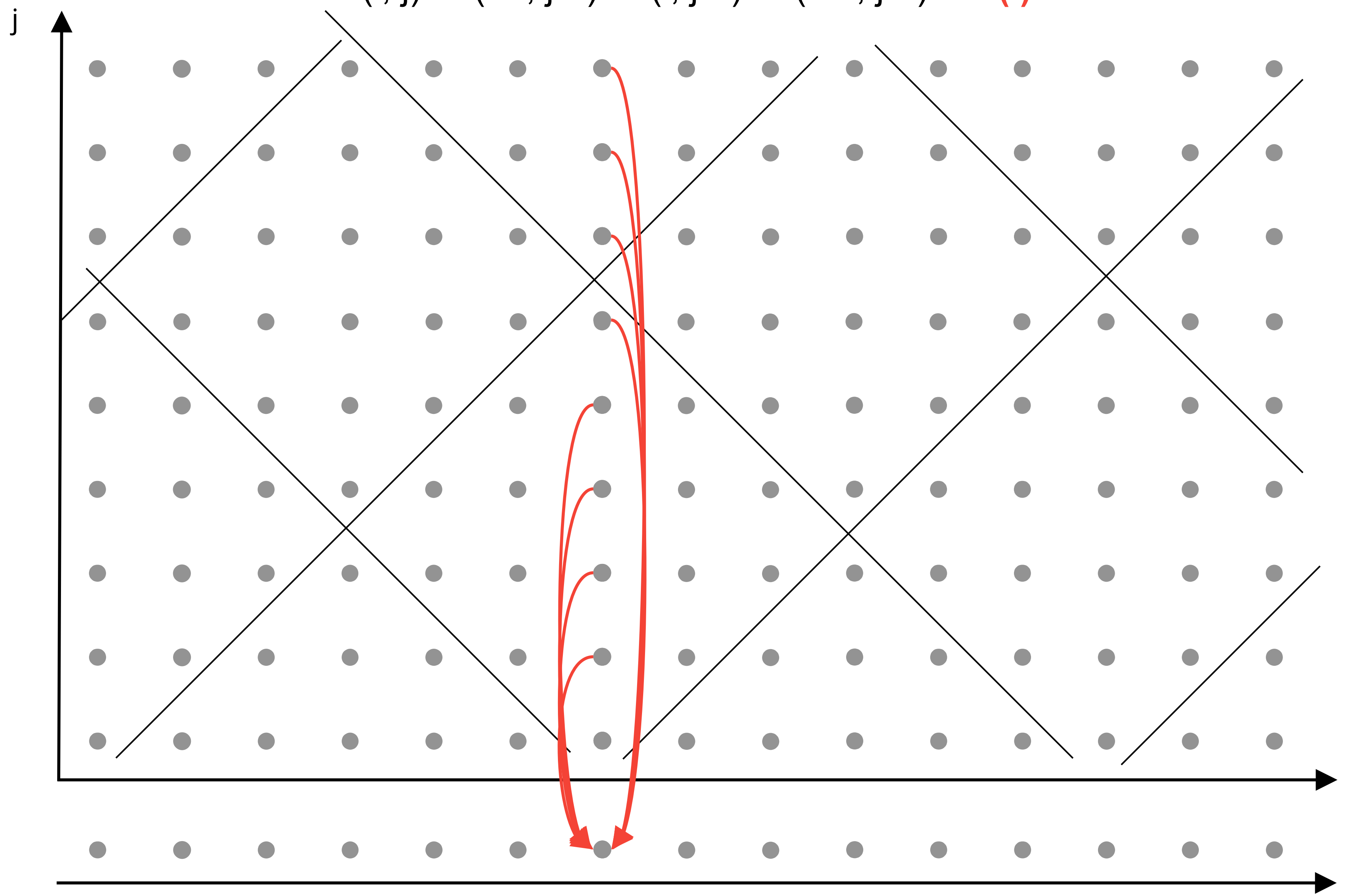
Can we partition input/output data as well?

Corentin Ferry, Steven Derrien, and Sanjay Rajopadhye. 2023. Maximal Atomic irRedundant Sets: a Usage-based Dataflow Partitioning Algorithm. In 13th International Workshop on Polyhedral Compilation Techniques (IMPACT'23).

The Problem with Affine Dependences

Number of consumer tiles potentially unbounded...

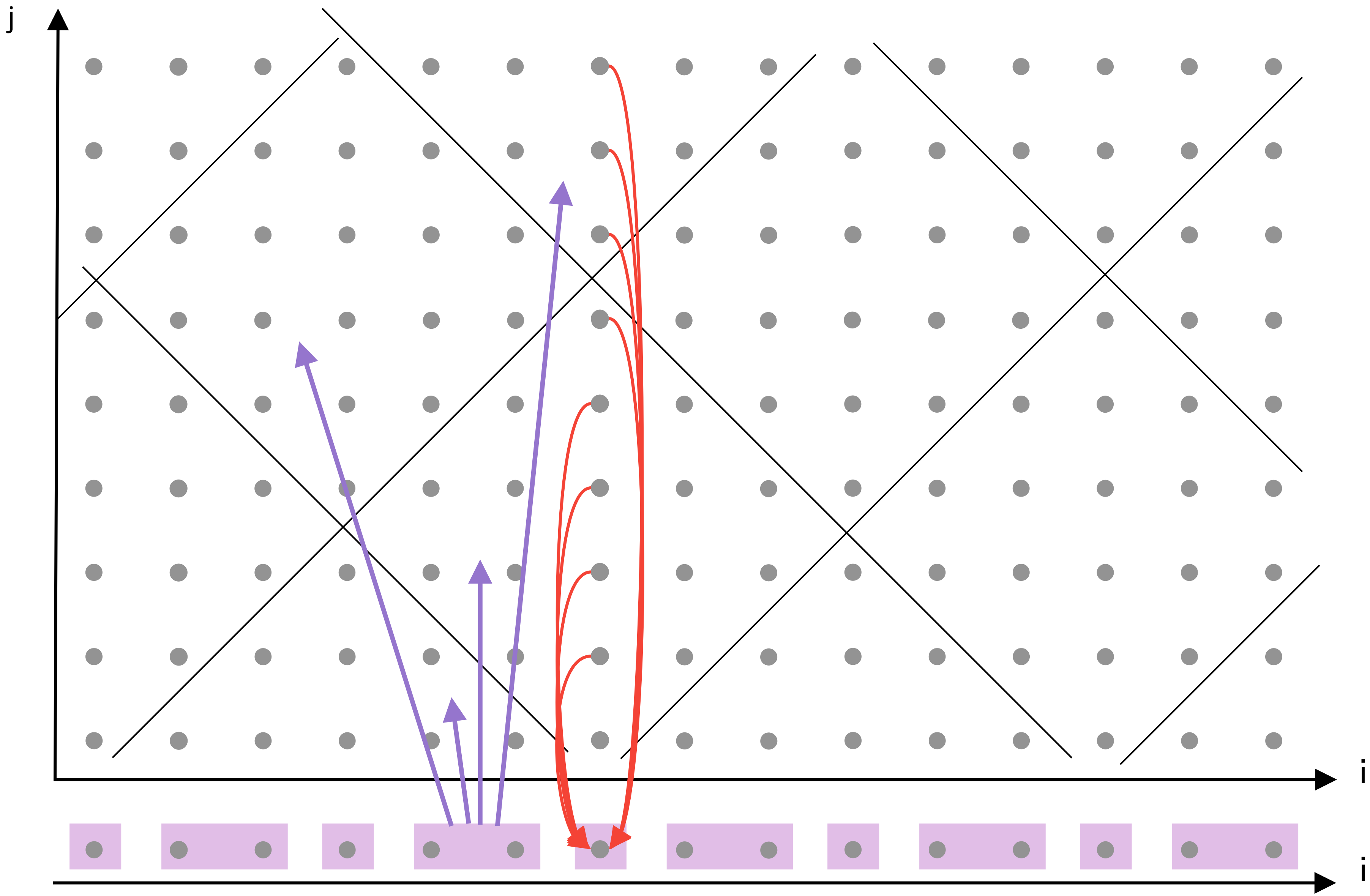
$f(i, j) = f(i-1, j-1) + f(i, j-1) + f(i+1, j-1) + \color{red}{\mathbf{A(i)}}$



The Problem with Affine Dependences

...but partitioning is still possible !

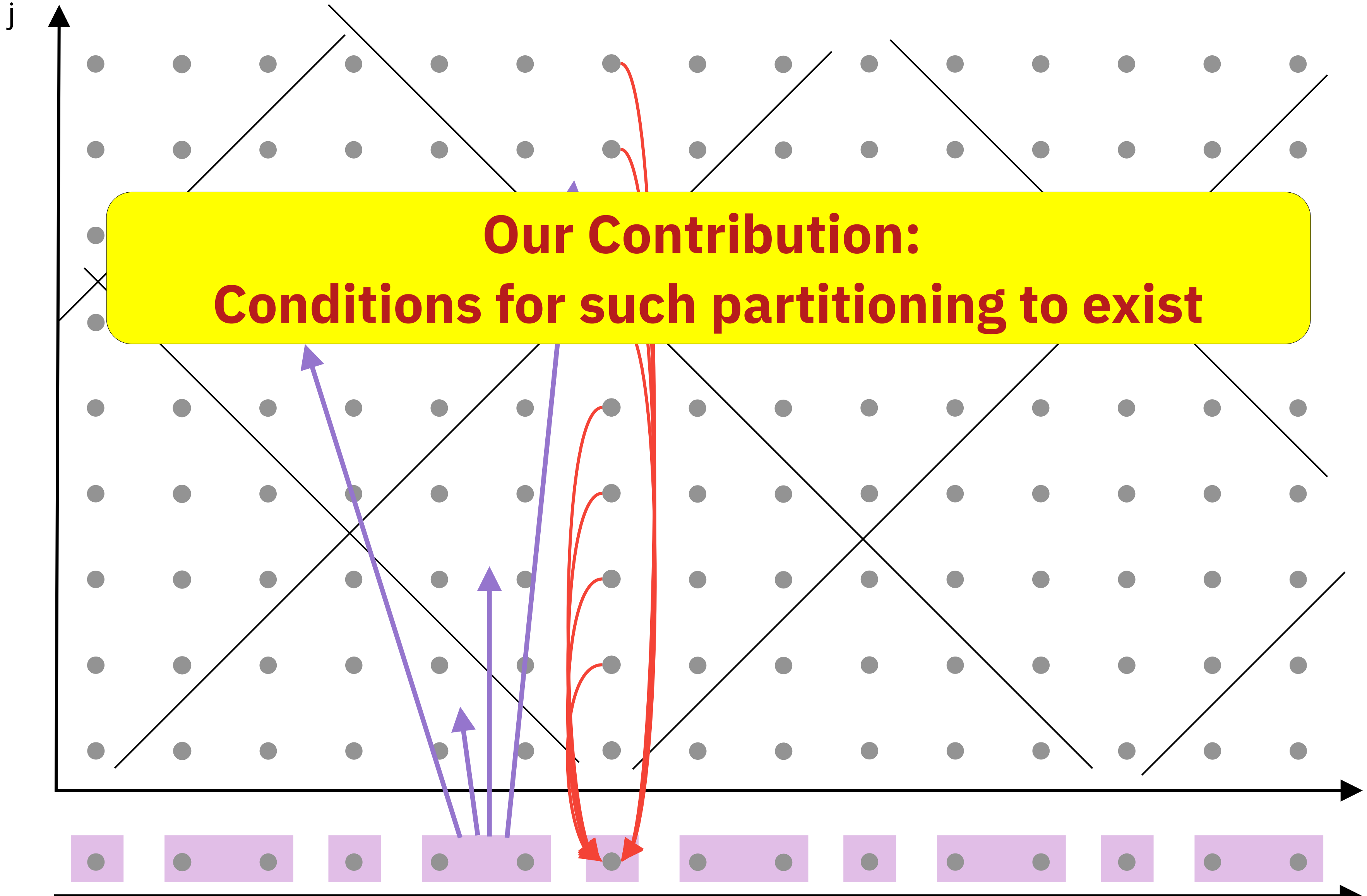
$$f(i, j) = f(i-1, j-1) + f(i, j-1) + f(i+1, j-1) + \mathbf{A(i)}$$



The Problem with Affine Dependences

...but partitioning is still possible !

$$f(i, j) = f(i-1, j-1) + f(i, j-1) + f(i+1, j-1) + \mathbf{A(i)}$$



Our Contribution:
Conditions for such partitioning to exist

Building MARS with affine dependences

- Consumer tiles must be represented in an **enumerable (finite) set**
  - There must be a finite number of elements in the partition
- The partition must (necessary?) be **invariant across tiles**
  - All tiles must have the same footprint shape

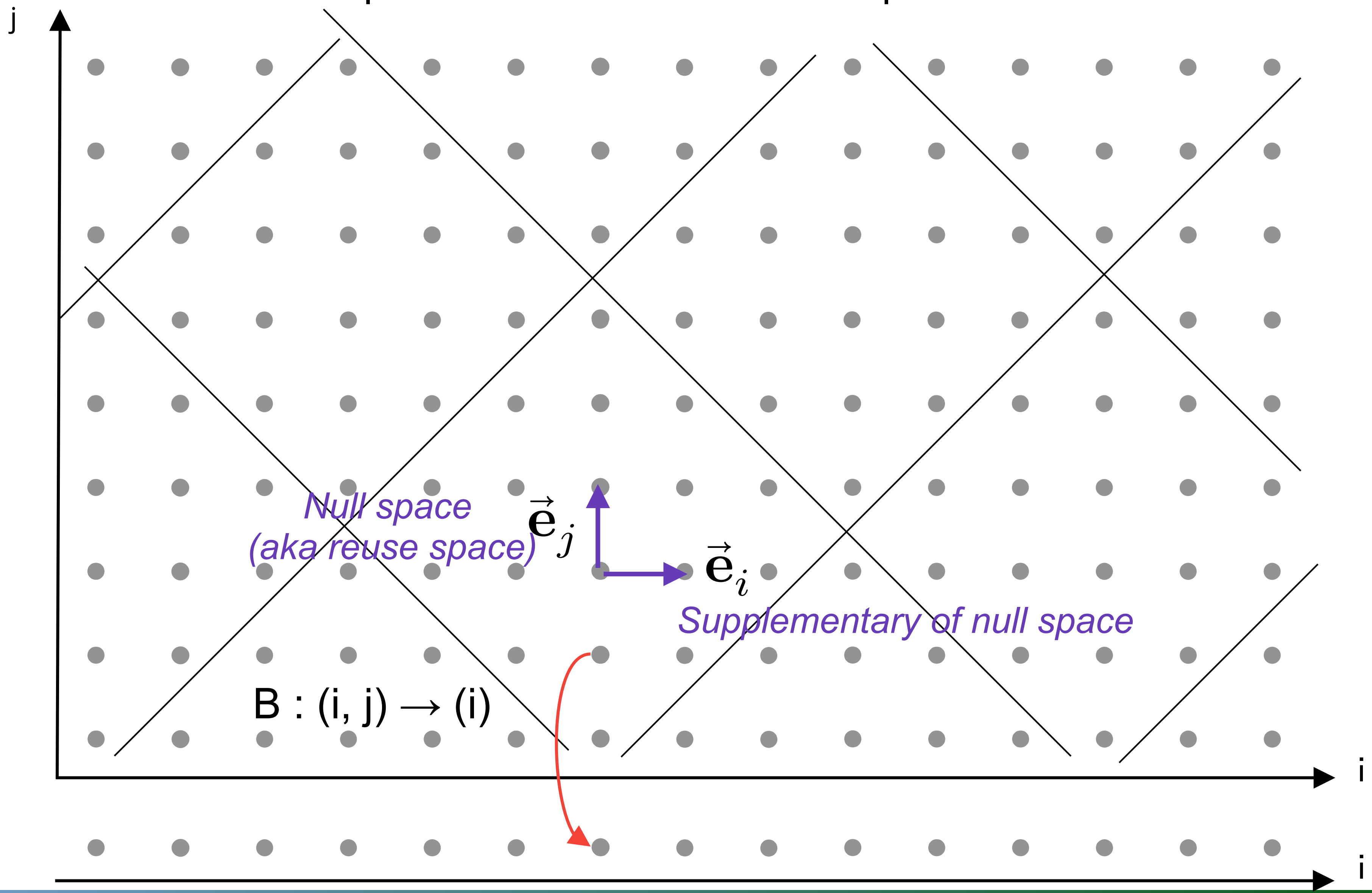
Dependences	**Consumers Enumerable**	**Partition Invariant**
Uniform (≥ 1)	Yes	Yes
Single Affine	Yes	Yes
Multiple Uniformly Intersecting	Yes	Yes
Multiple Affine Same null space	Conditional	Maybe (we don't know)
Multiple Affine Multiple null spaces	No	No

Outline

1. **Introduction**
    1. Context
    2. Loop Tiling as a Locality Optimization
    3. MARS : Partitioning Data for Spatial Locality
2. **Partitioning Data with Affine Dependences**
    1. Single affine dependence
    2. Multiple uniformly intersecting dependences
    3. Multiple non-uniformly intersecting dependences
    4. Dependences to tiled iteration spaces
3. **Conclusions**

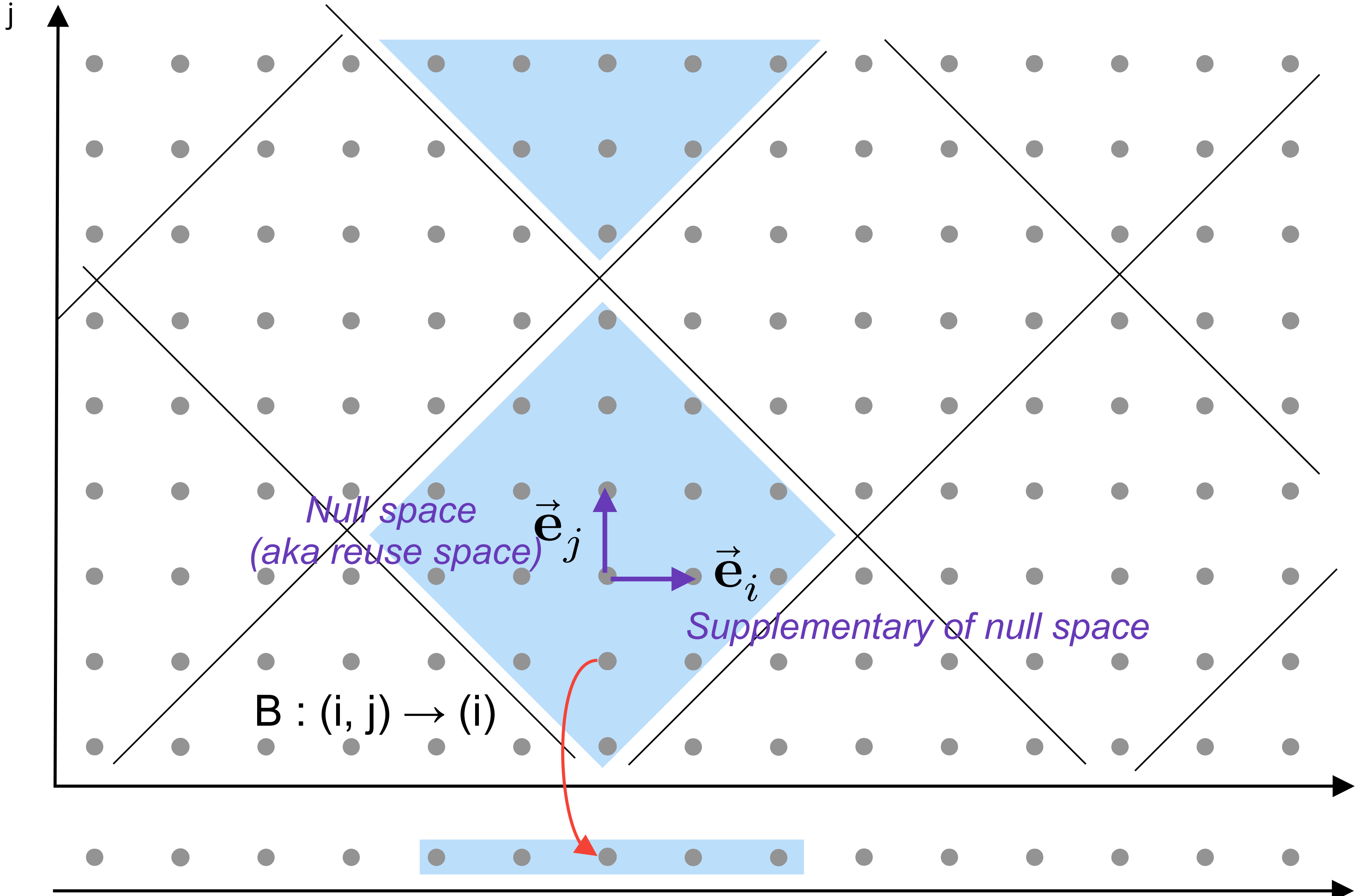
Single affine dependence

- **Goal :** make consumer tiles of every point enumerable
- **How :** use the dependence function's null space



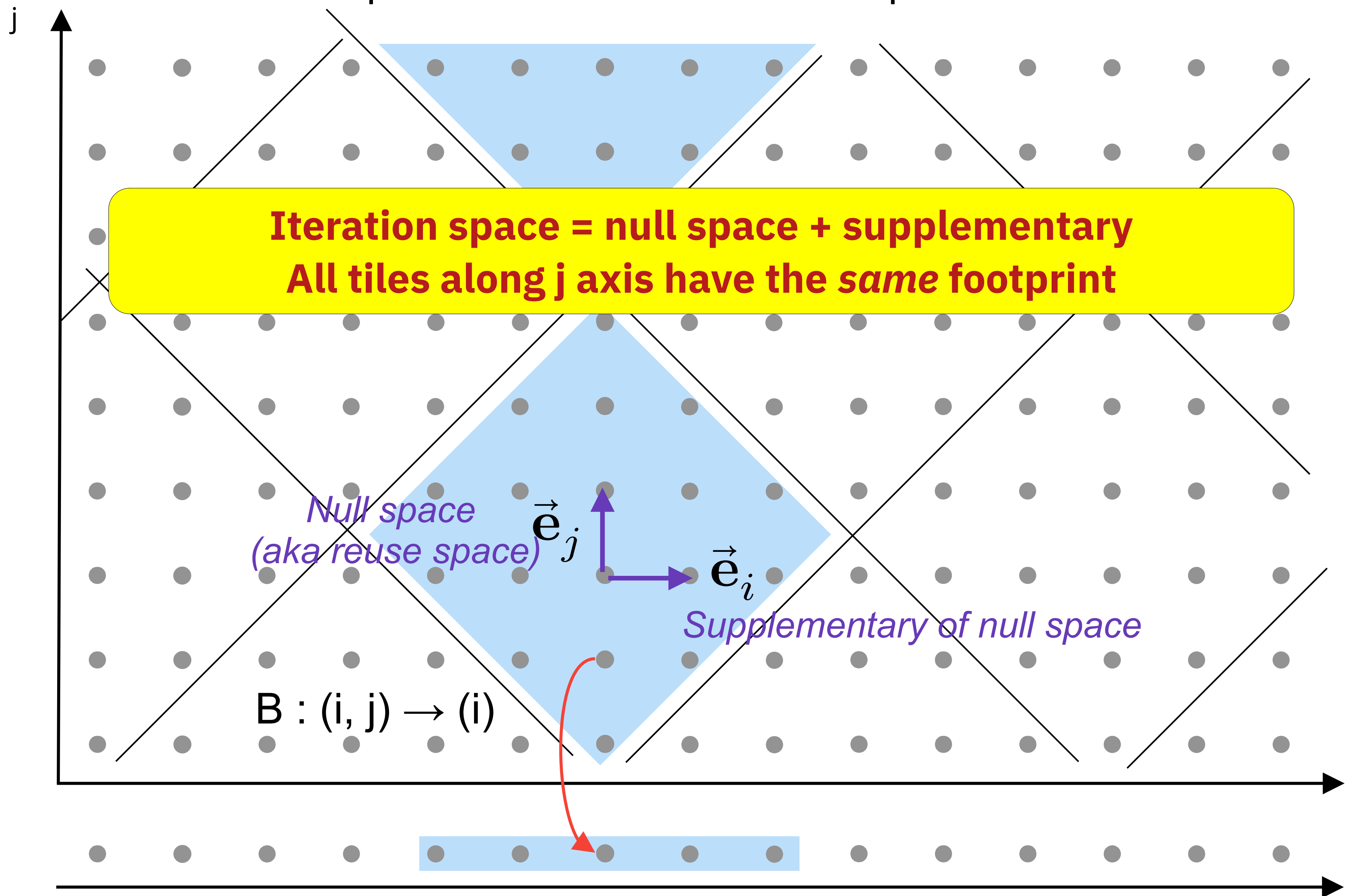
Single affine dependence

- **Goal :** make consumer tiles of every point enumerable
- **How :** use the dependence function's null space



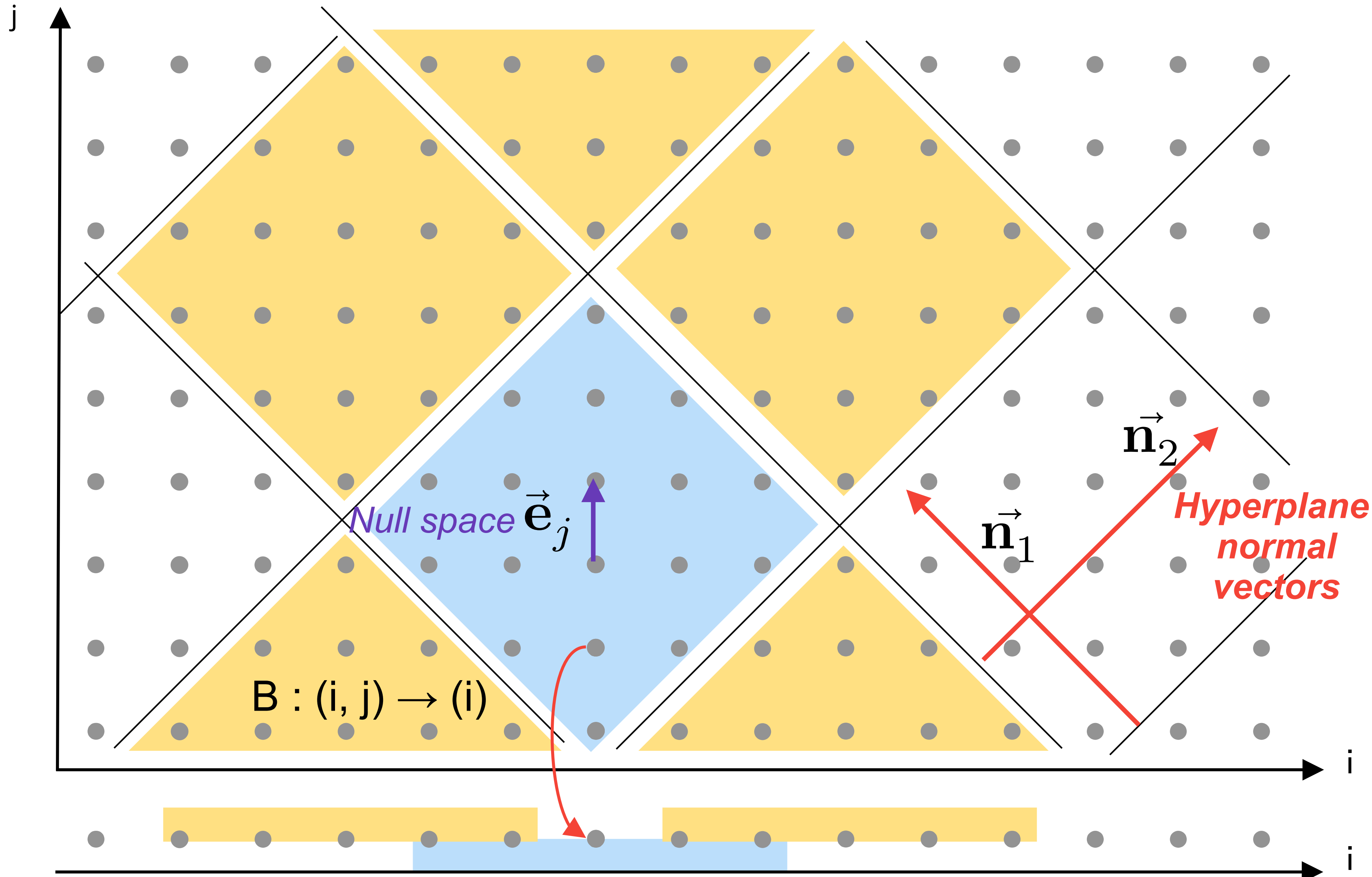
Single affine dependence

- **Goal :** make consumer tiles of every point enumerable
- **How :** use the dependence function's null space



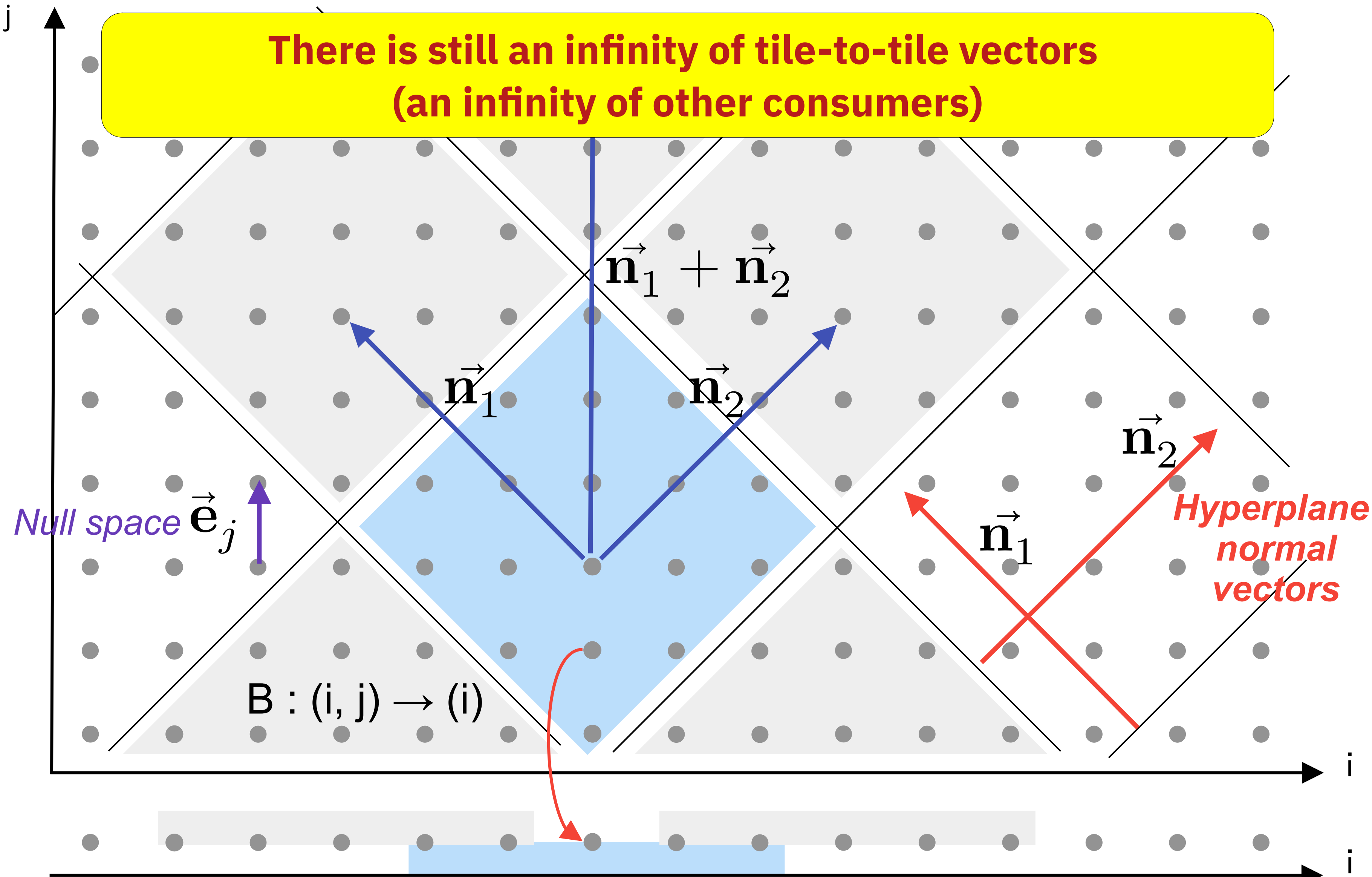
Single affine dependence

What We Need : normal vectors, reference tile (parameter), other tiles which footprint intersects with the reference tile (set)



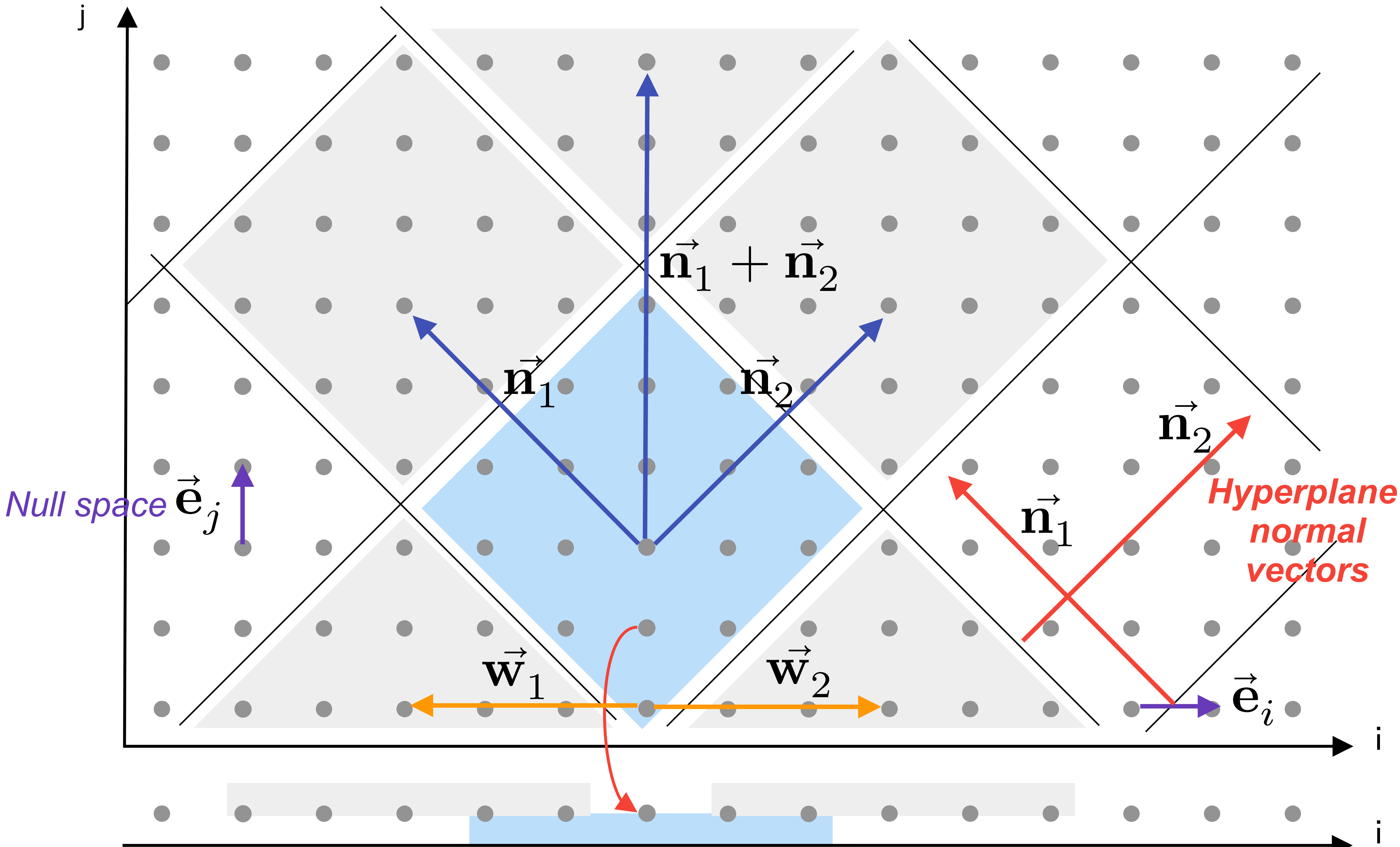
Single affine dependence

Transform « other consumers » into tile-to-tile vectors (translations)



Single affine dependence

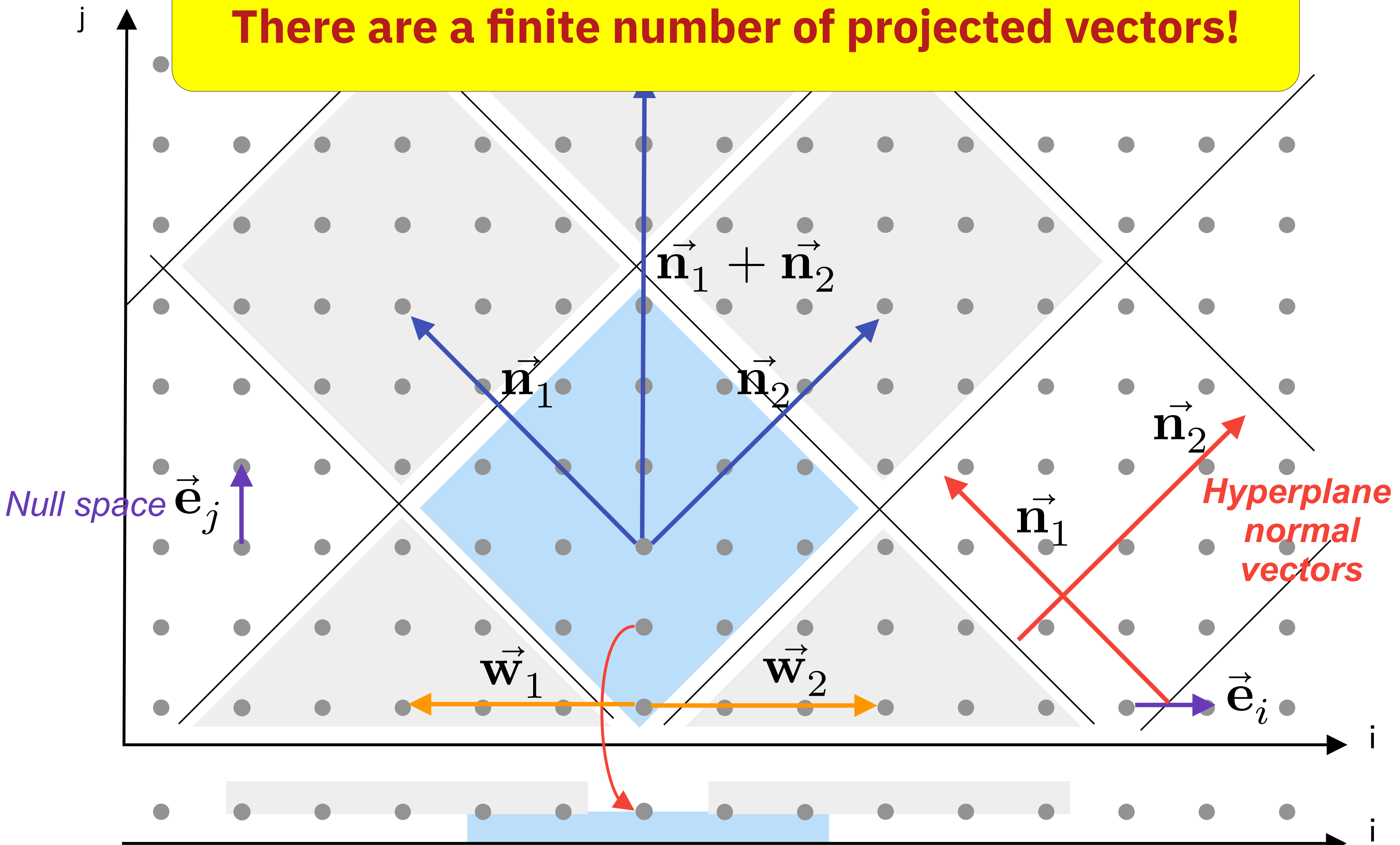
Project out the null space component of the translation vectors



Single affine dependence

Project out the null space component of the translation vectors

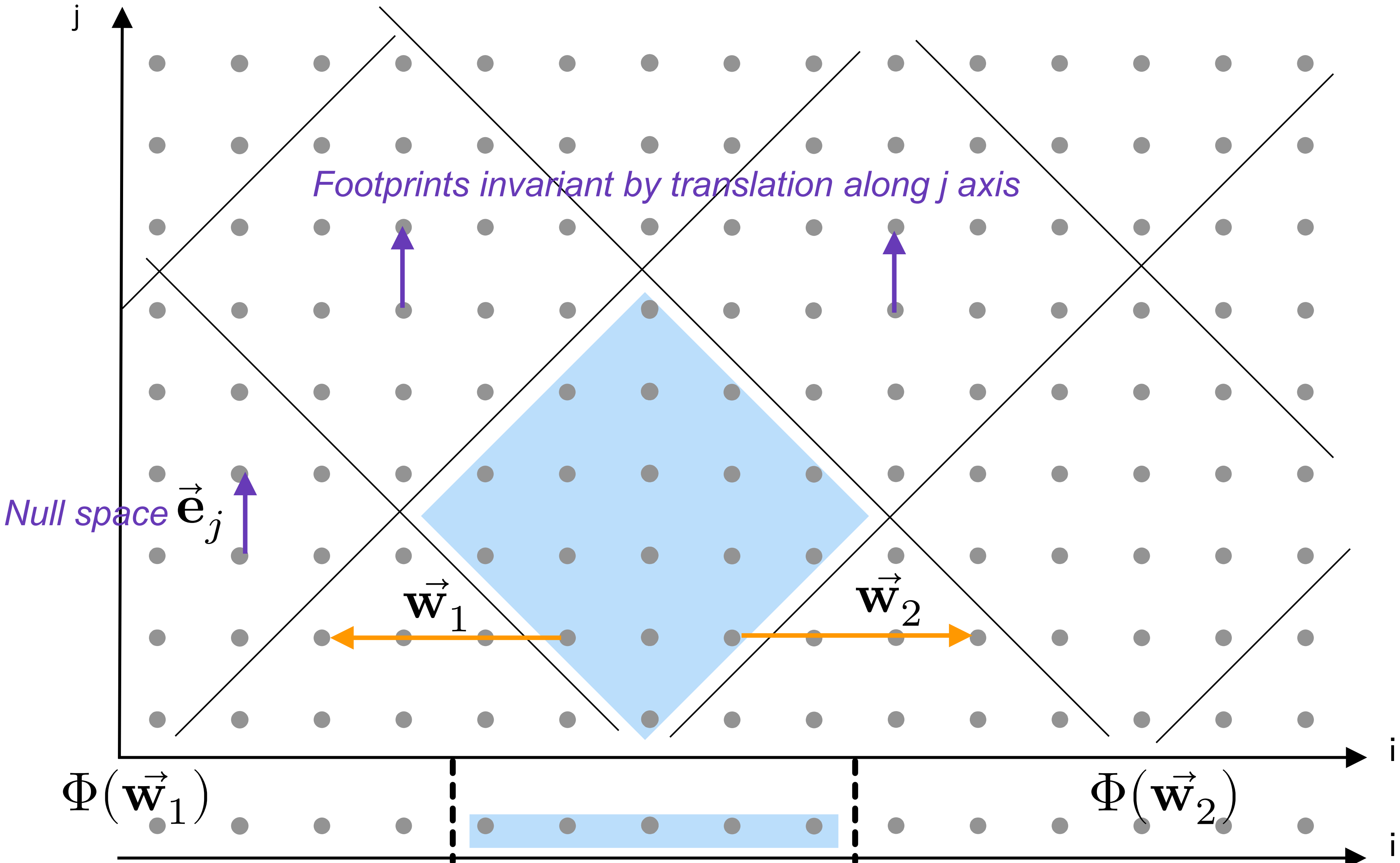
There are a finite number of projected vectors!



Constructing the MARS

For all w → Get footprint of tile translated by w

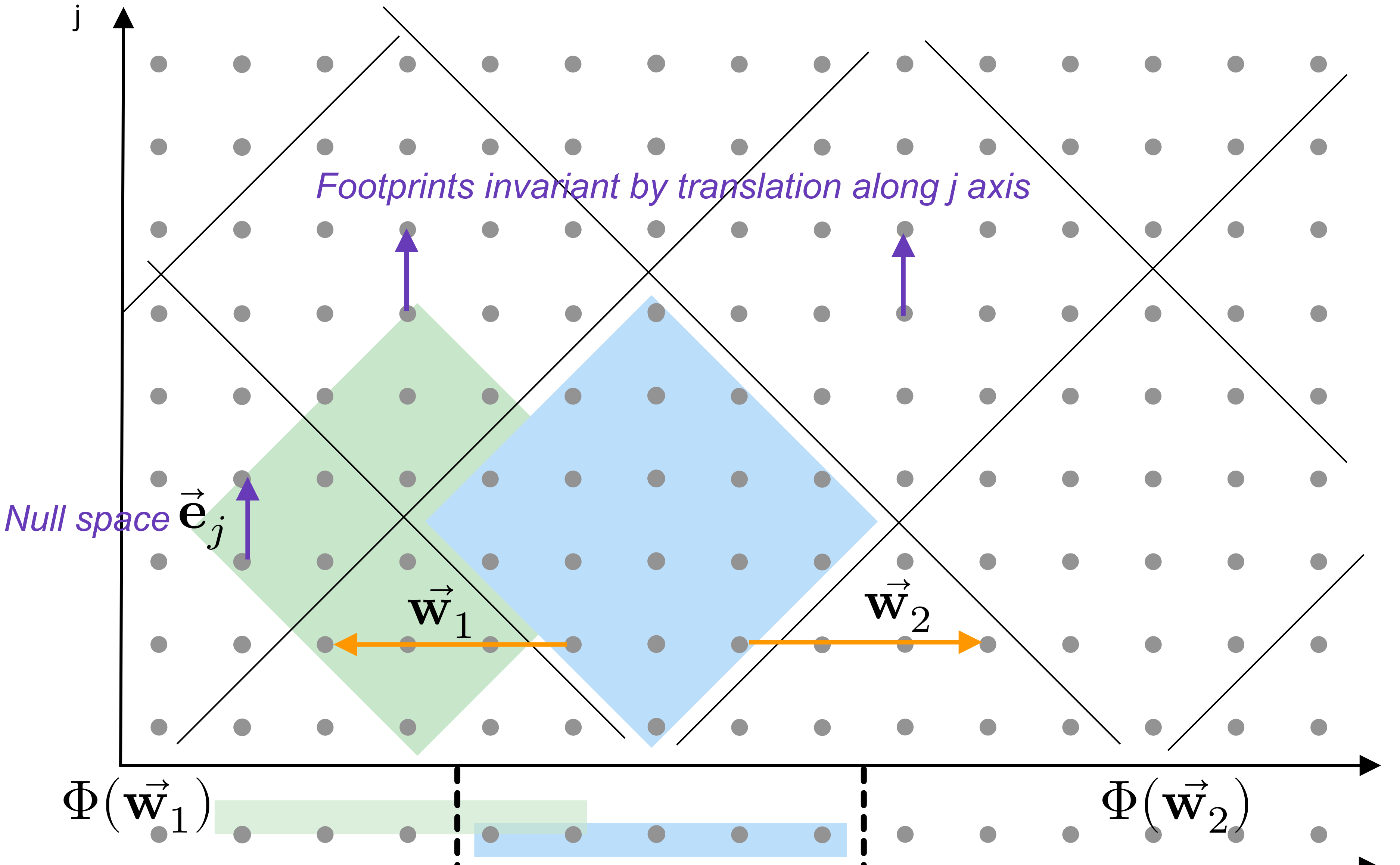
Intersect with current tile's footprint



Constructing the MARS

For all w → Get footprint of tile translated by w

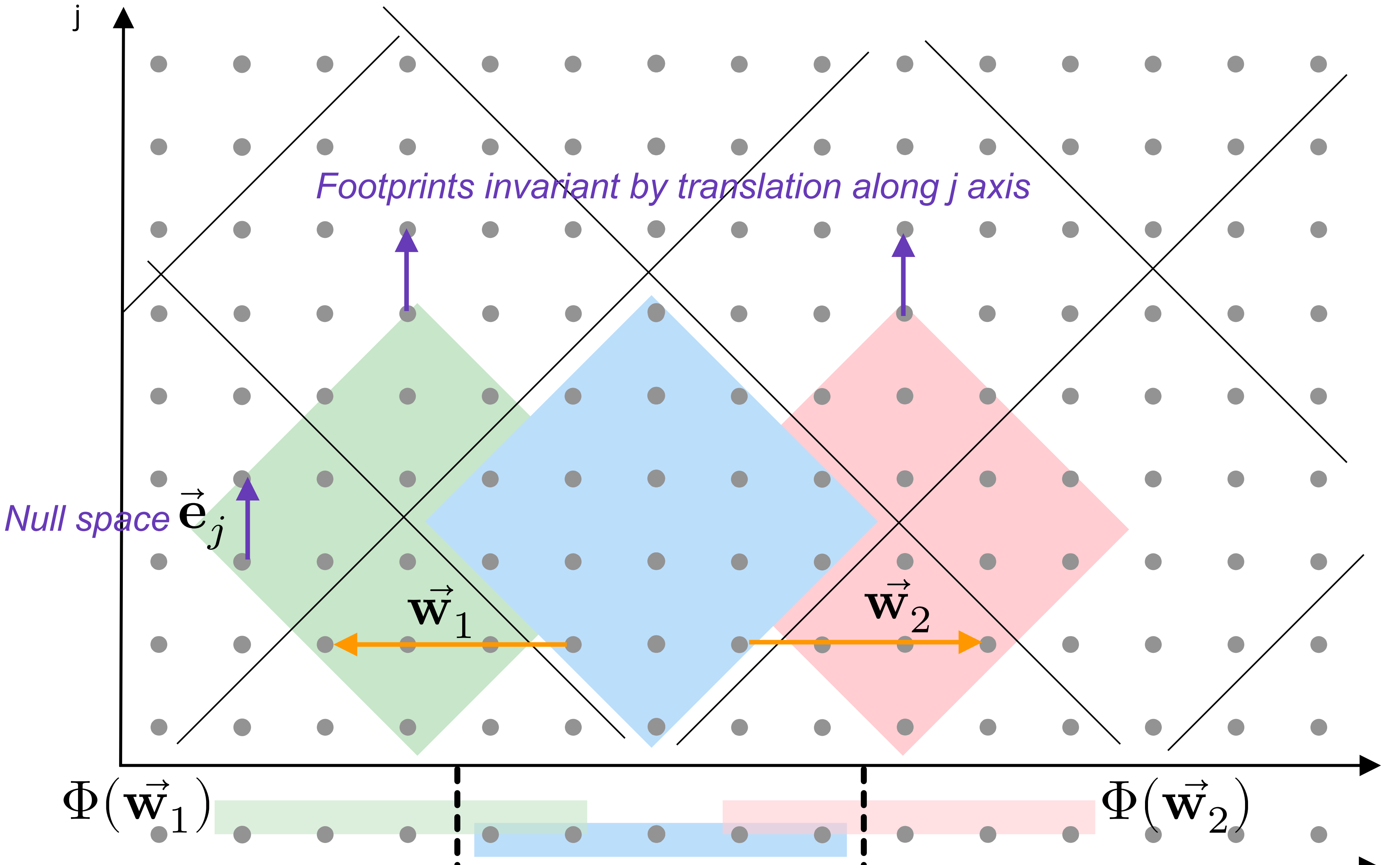
Intersect with current tile's footprint



Constructing the MARS

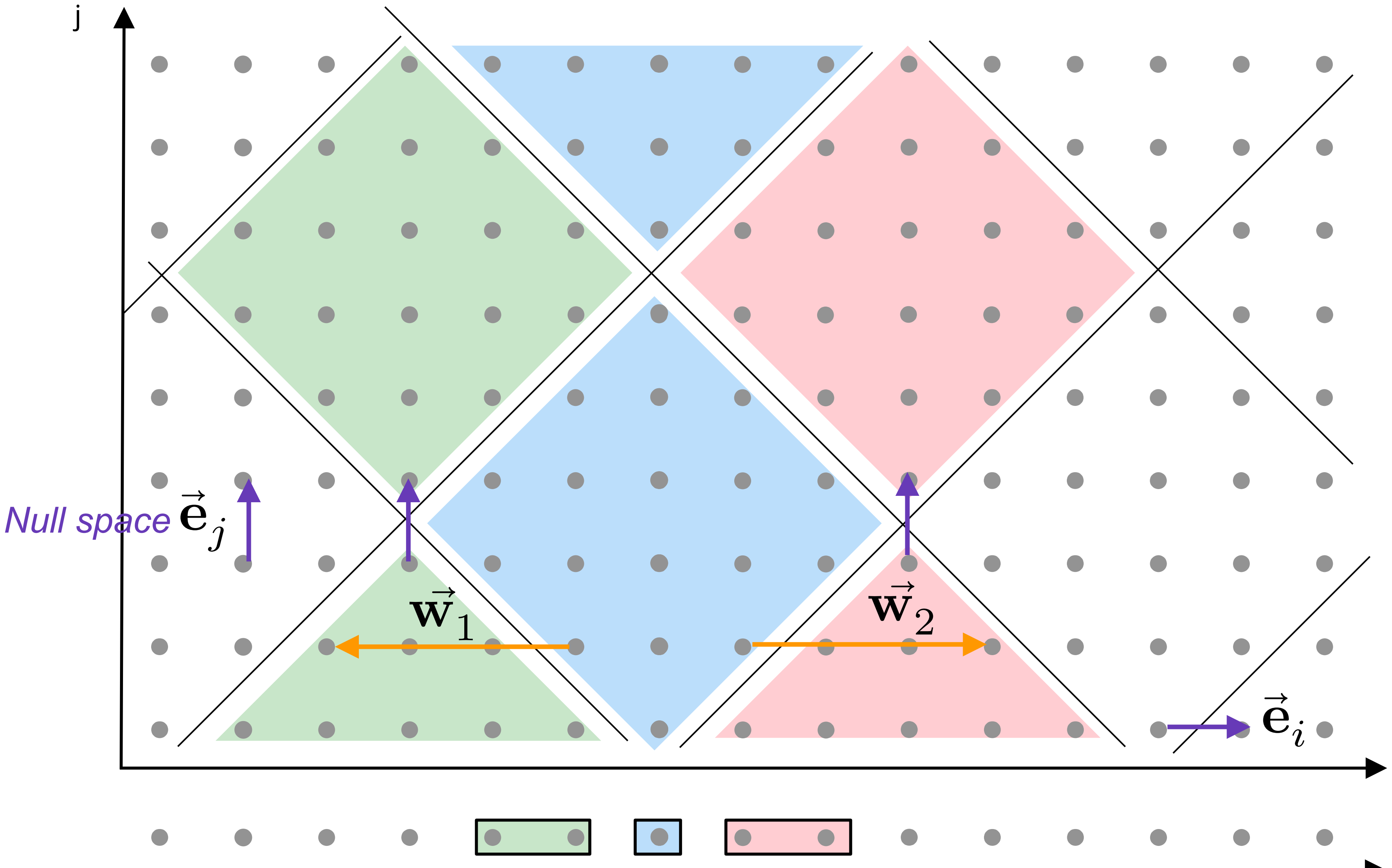
For all w → Get footprint of tile translated by w

Intersect with current tile's footprint



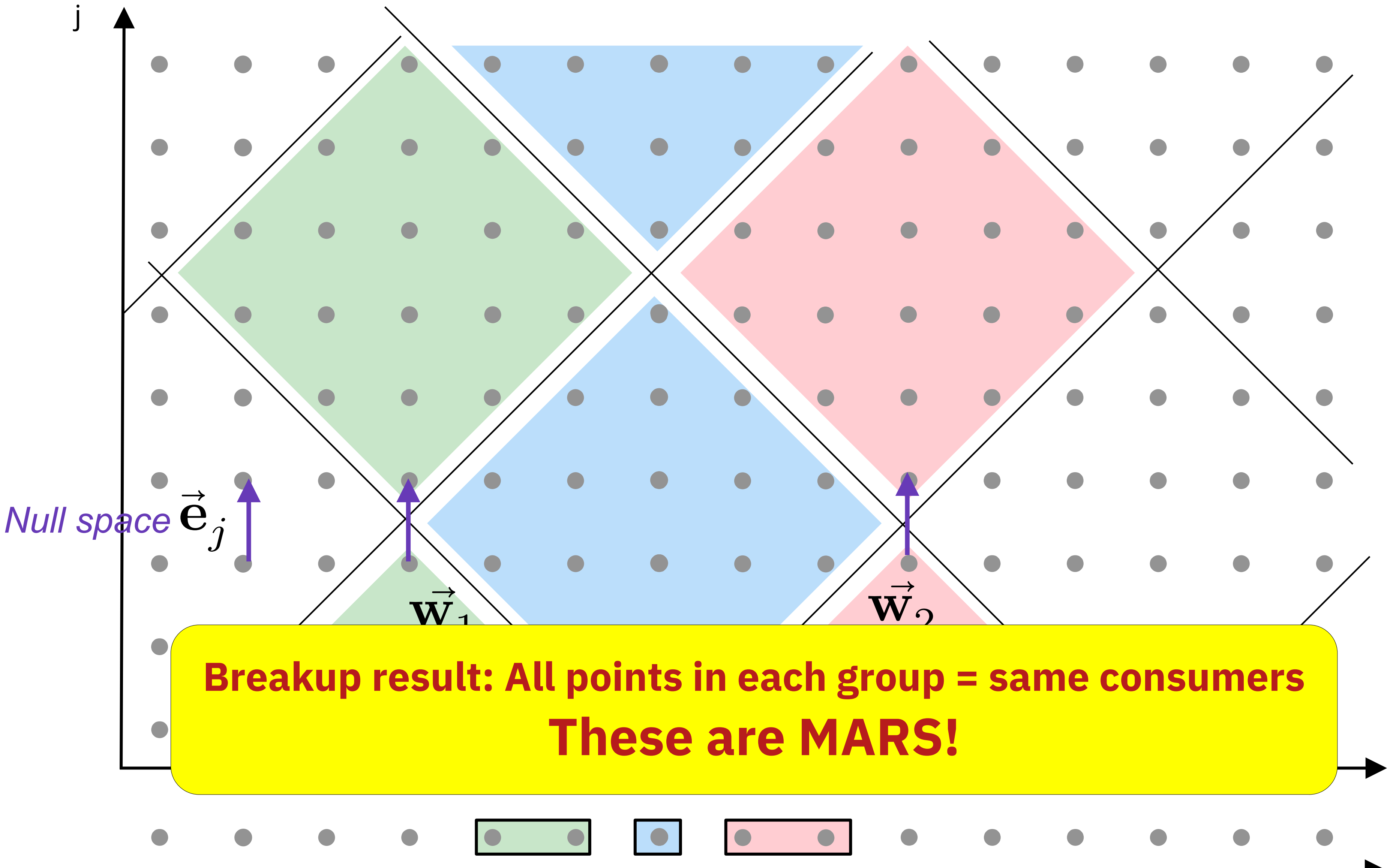
Constructing the MARS

Break up current tile's footprint per tuple of consumer tiles



Constructing the MARS

Break up current tile's footprint per tuple of consumer tiles

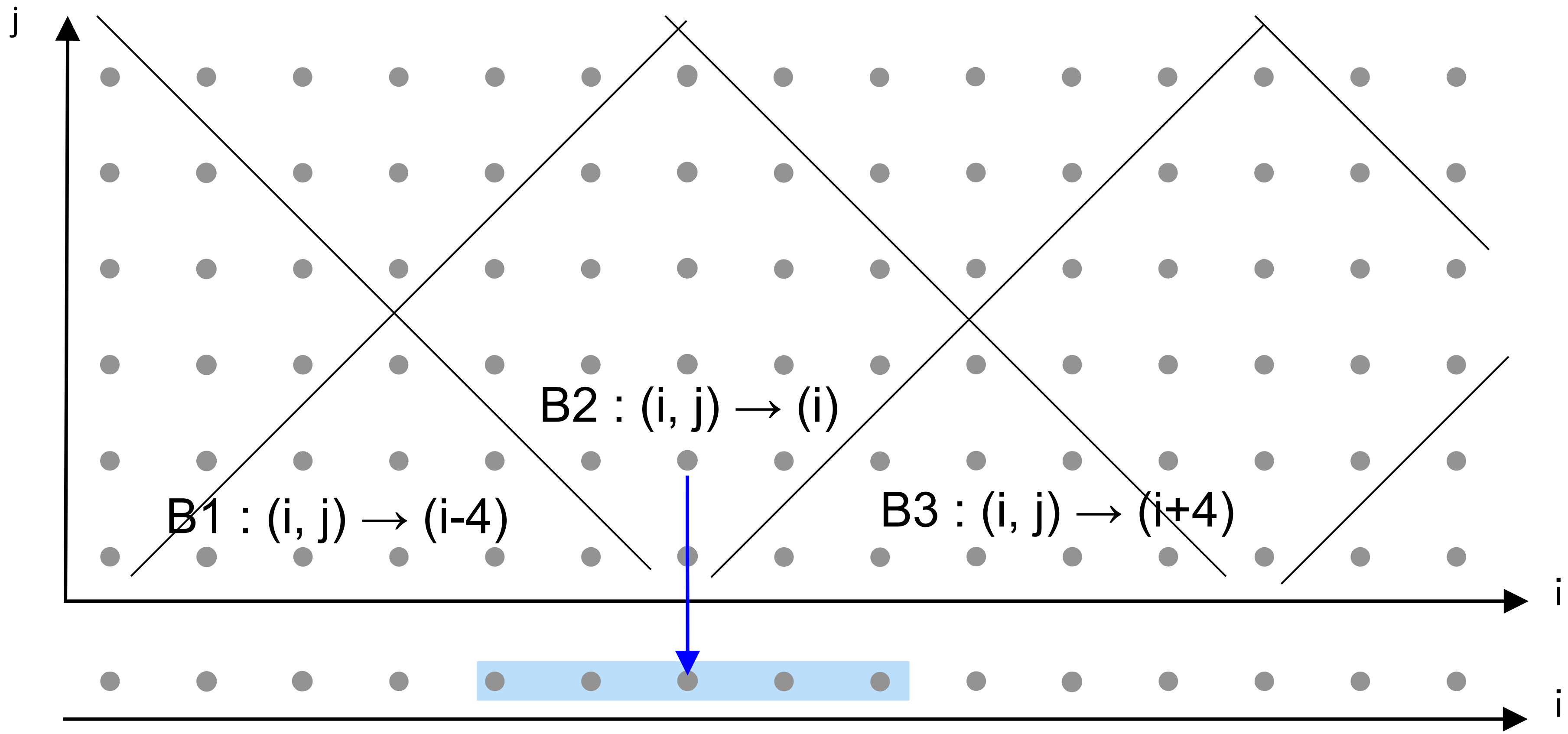


Breakup result: All points in each group = same consumers

These are MARS!

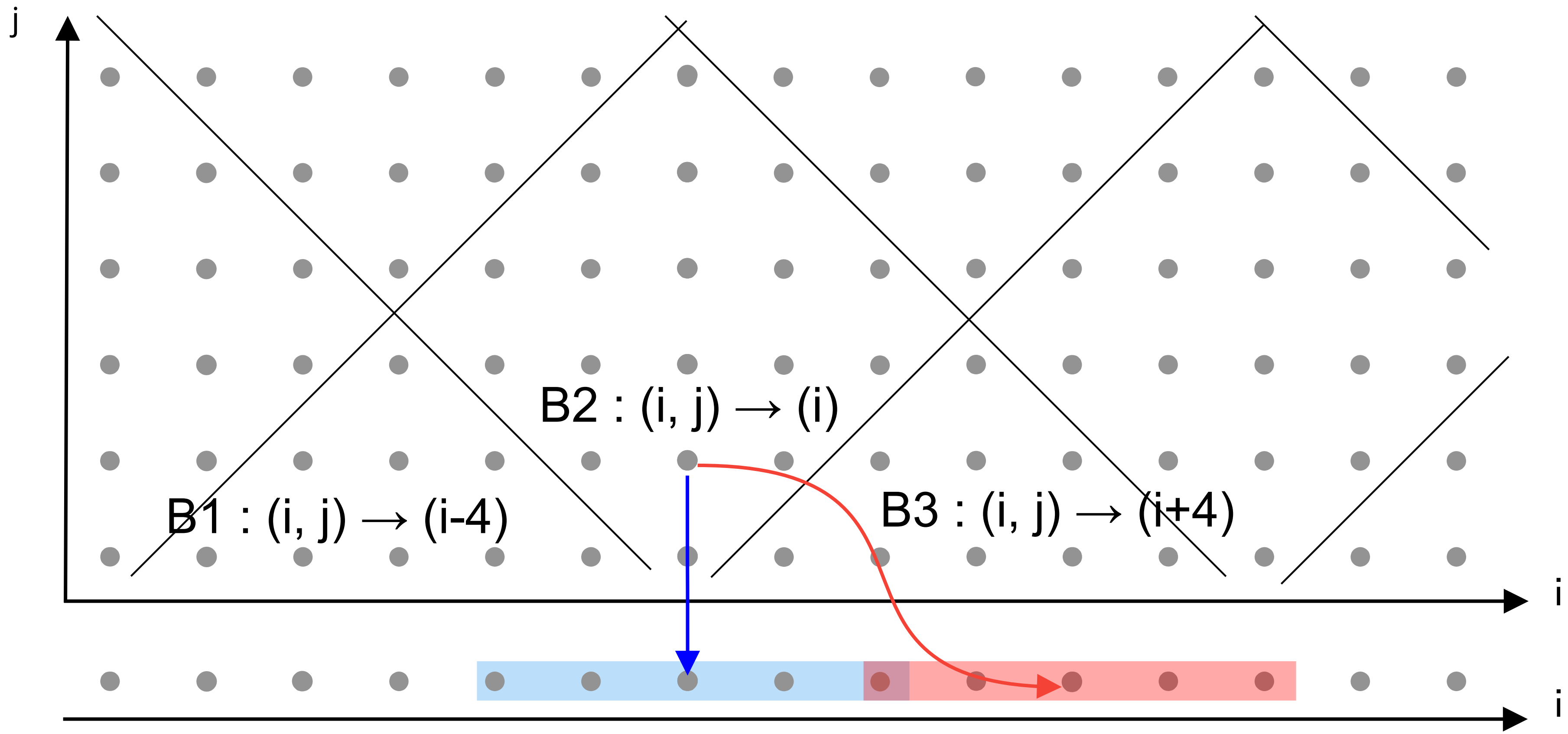
Uniformly intersecting dependences

- **Multiple dependences, same linear part** (i.e. same null space)
- Example : B1 : (i, j) → (i-4) ; B2 : (i, j) → (i) ; B3 : (i, j) → (i+4)



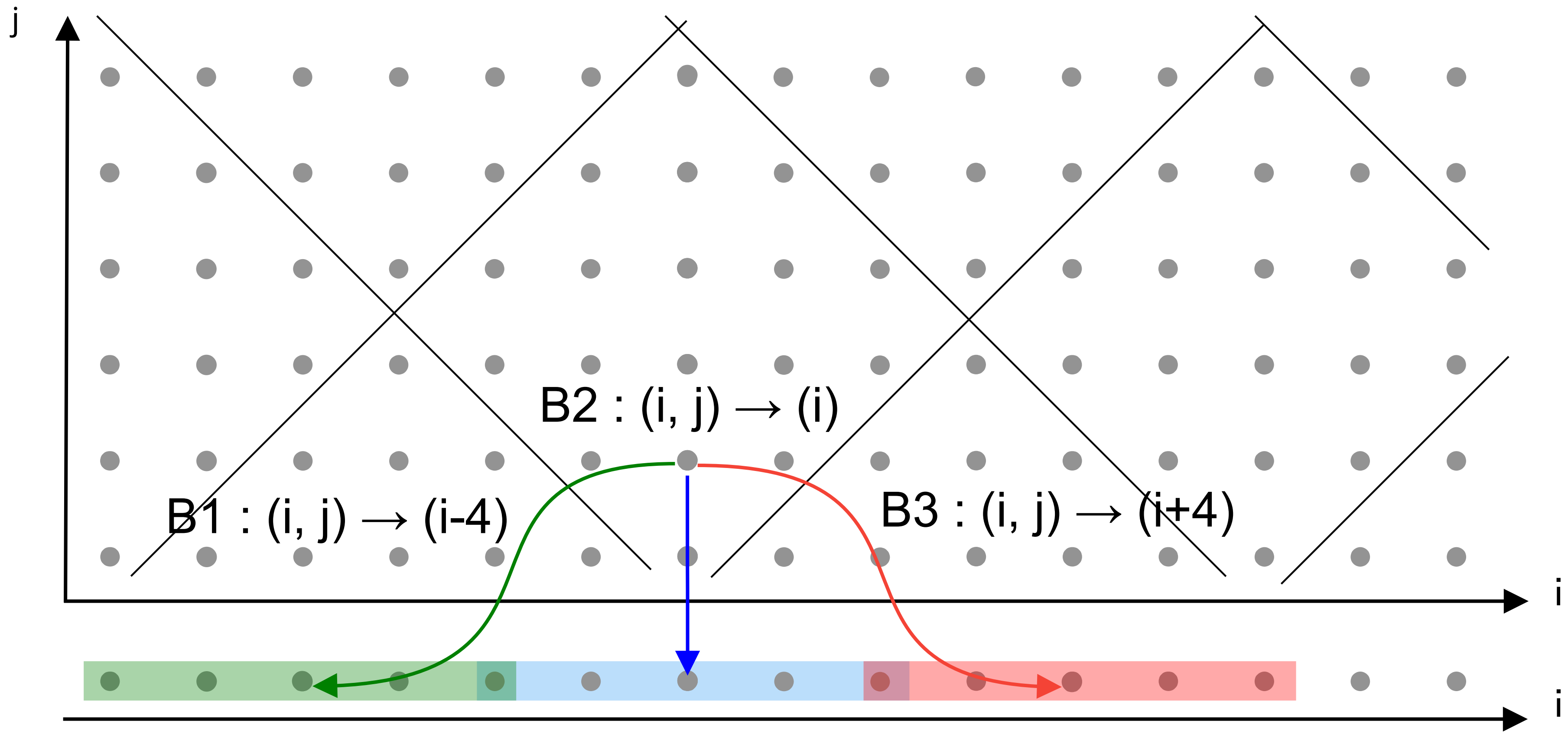
Uniformly intersecting dependences

- **Multiple dependences, same linear part** (i.e. same null space)
- Example : B1 : (i, j) → (i-4) ; B2 : (i, j) → (i) ; B3 : (i, j) → (i+4)



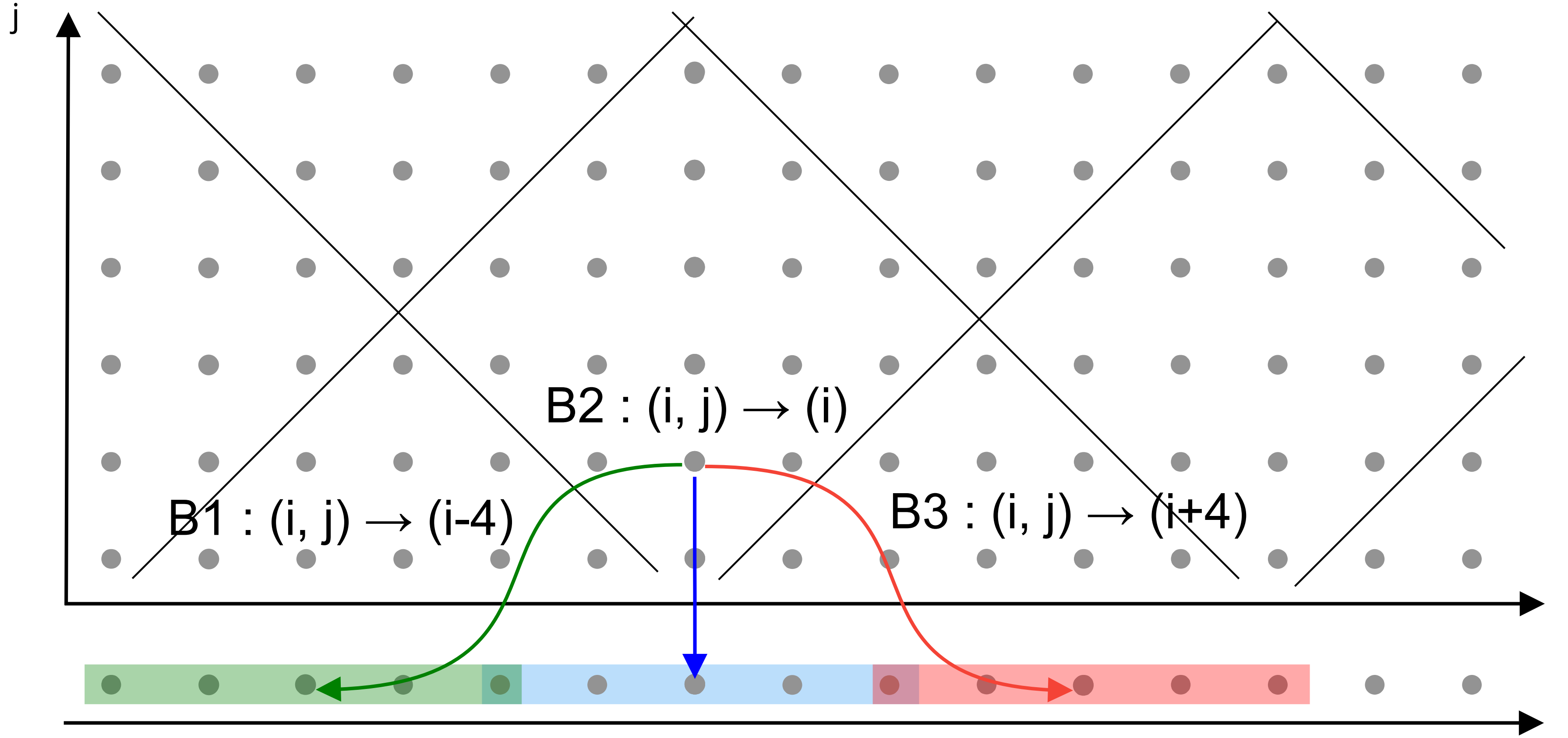
Uniformly intersecting dependences

- **Multiple dependences, same linear part** (i.e. same null space)
- Example : B1 : (i, j) → (i-4) ; B2 : (i, j) → (i) ; B3 : (i, j) → (i+4)



Uniformly intersecting dependences

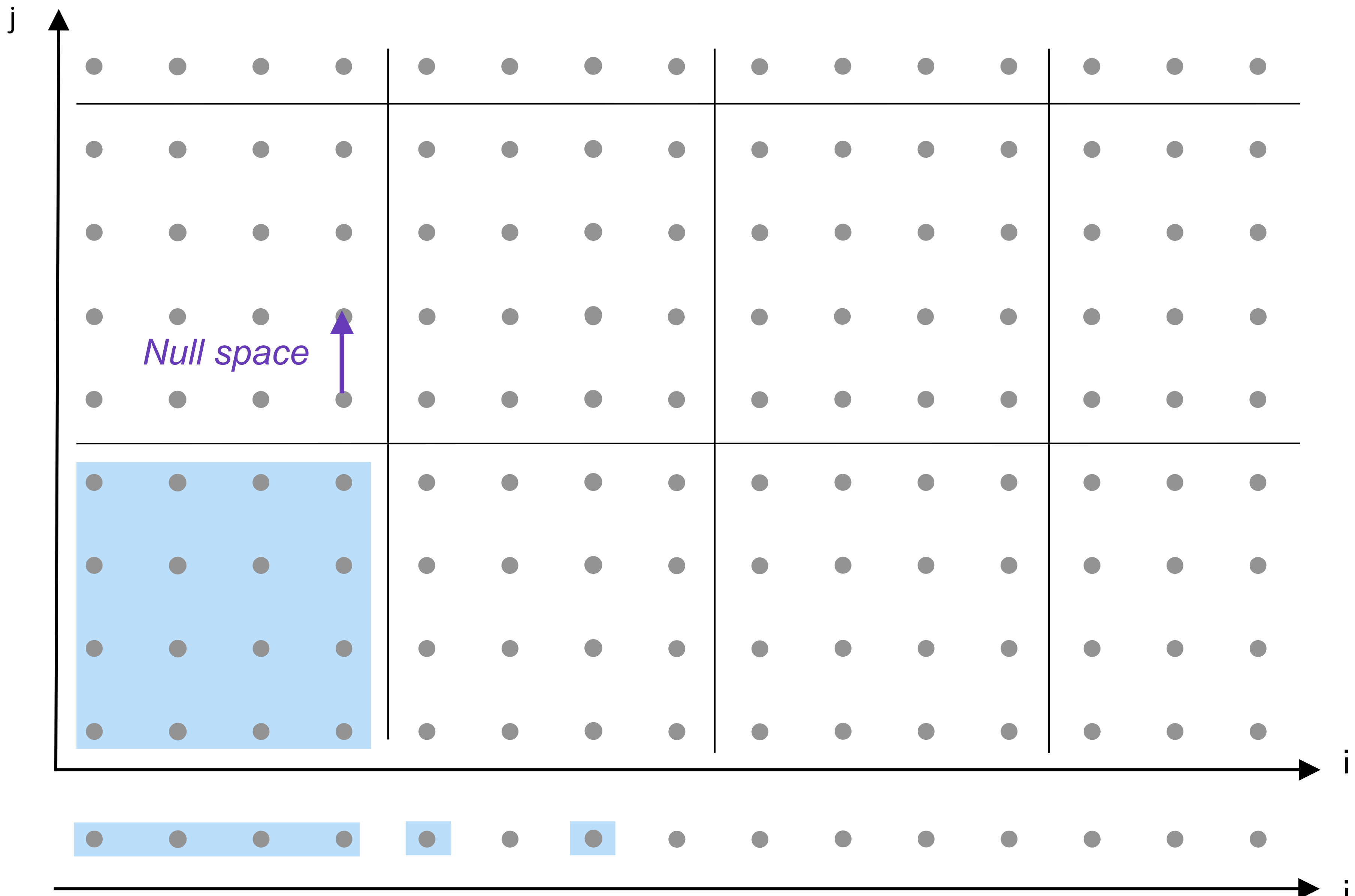
- **Multiple dependences, same linear part** (i.e. same null space)
- Example : B1 : (i, j) → (i-4) ; B2 : (i, j) → (i) ; B3 : (i, j) → (i+4)



No extra issue in enumerating all consumer tiles
→ Uniformly translate footprints

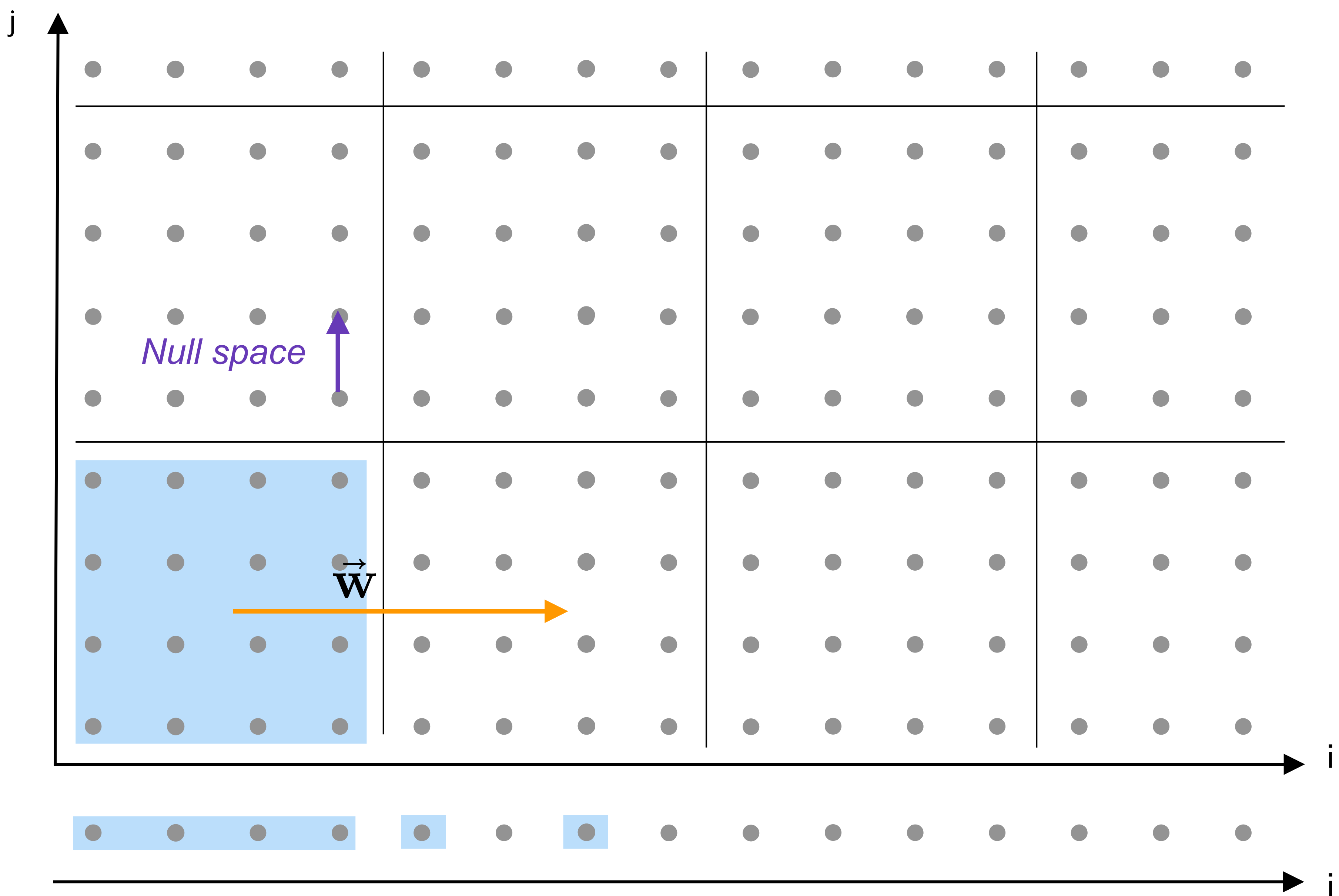
Non-uniformly intersecting dependences

Example w/ same null space: (i, j) → (i) ; (i, j) → (2i)



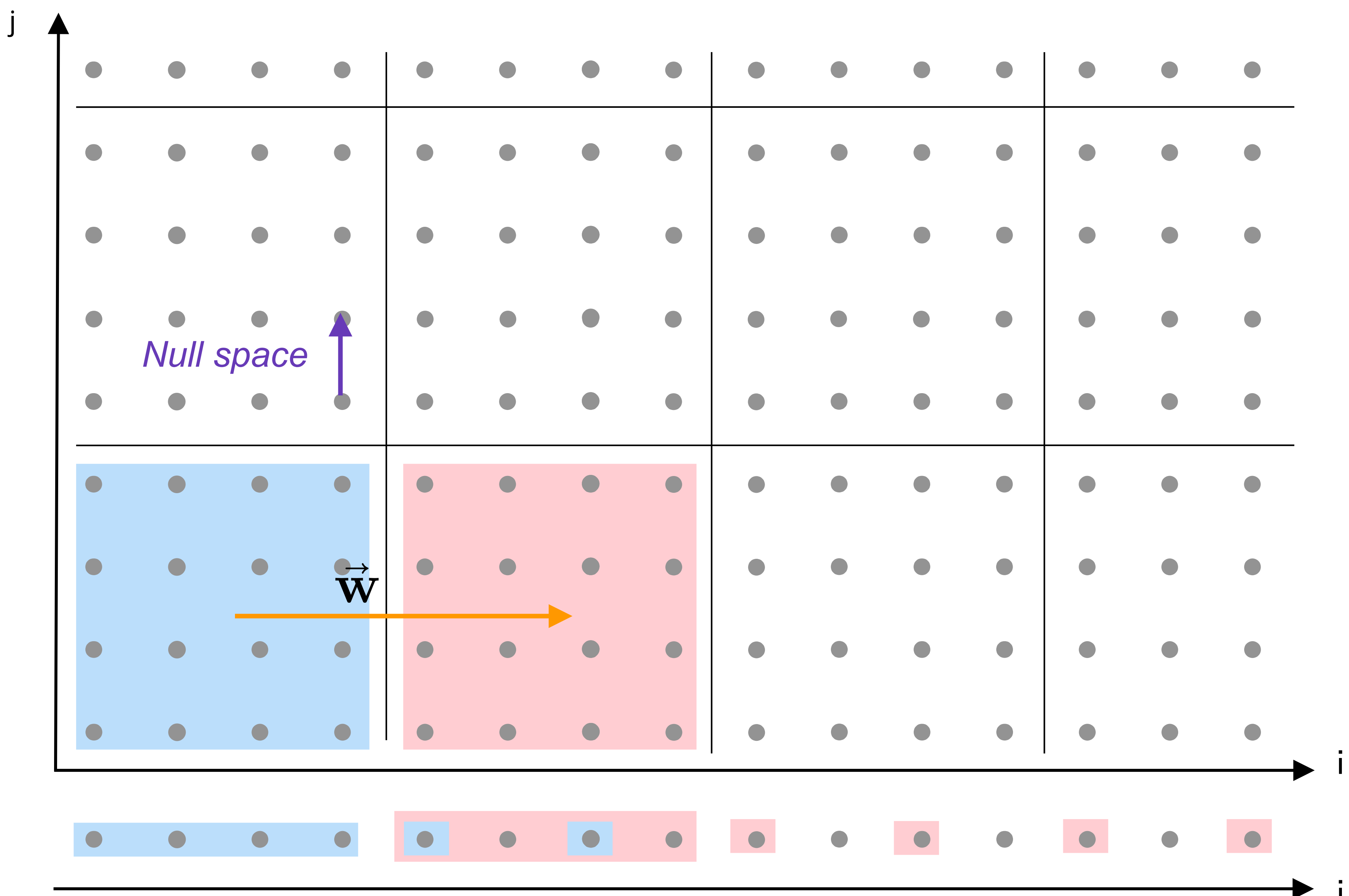
Non-uniformly intersecting dependences

Example w/ same null space: (i, j) → (i) ; (i, j) → (2i)



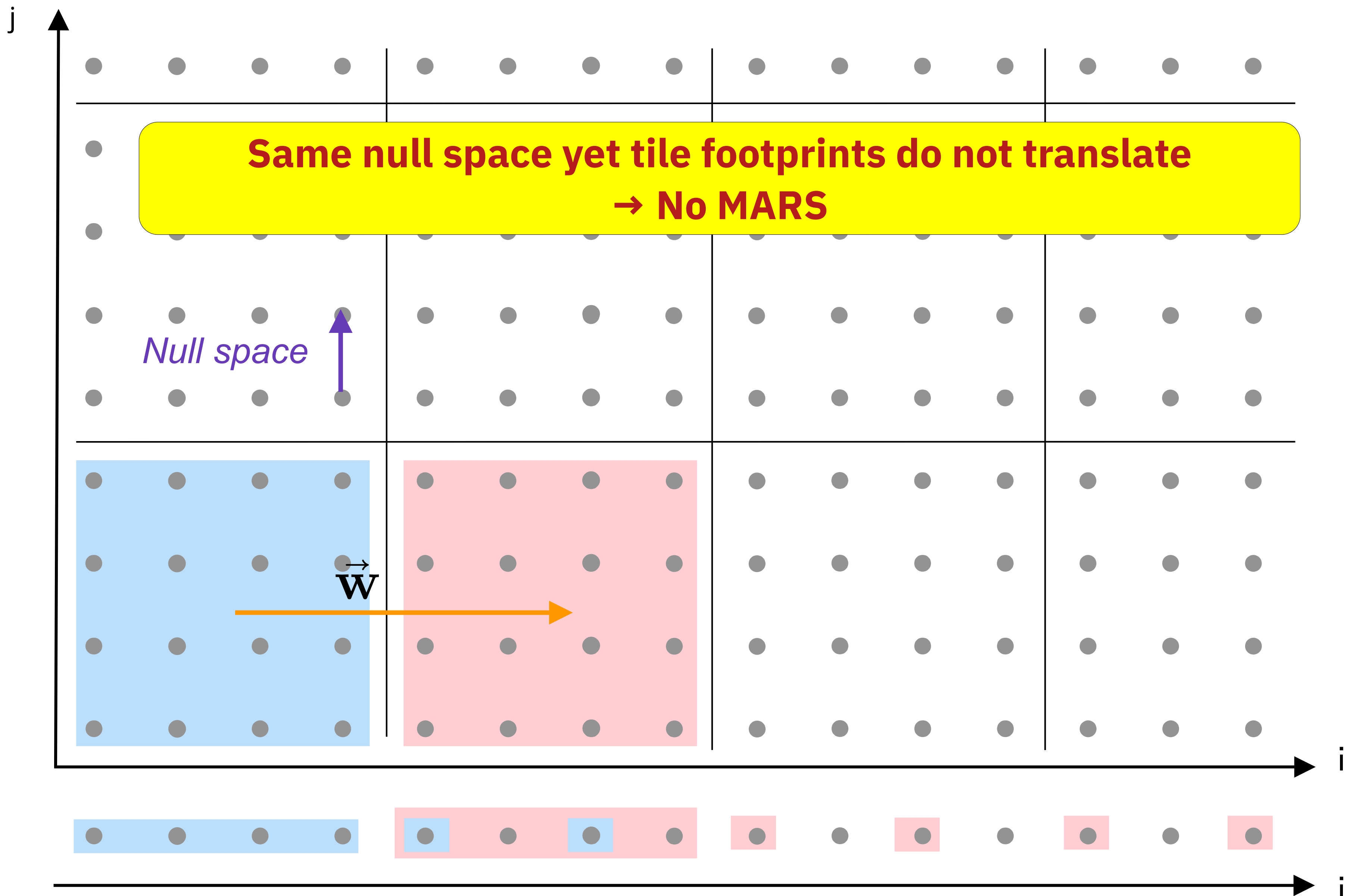
Non-uniformly intersecting dependences

Example w/ same null space: (i, j) → (i) ; (i, j) → (2i)



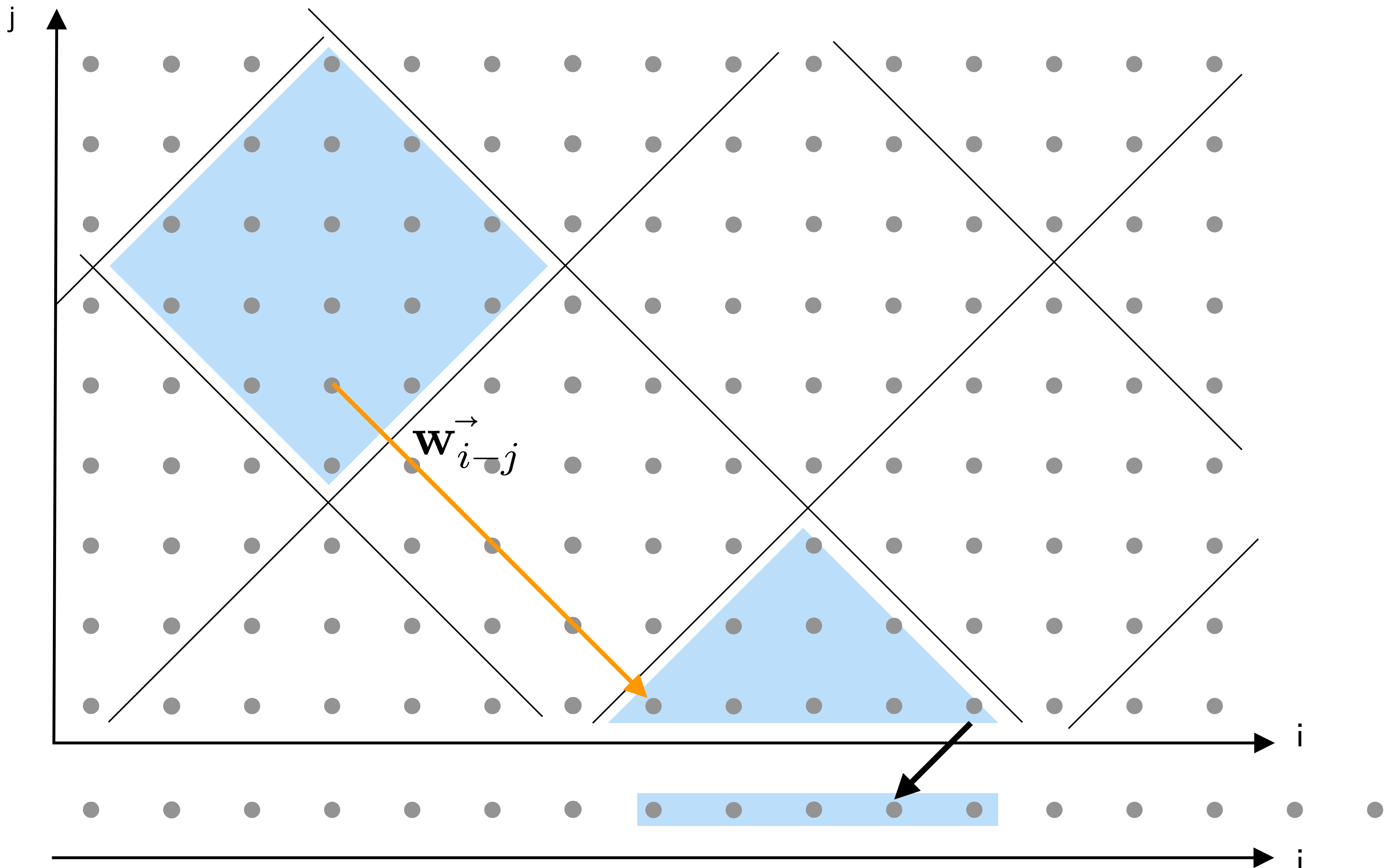
Non-uniformly intersecting dependences

Example w/ same null space: (i, j) → (i) ; (i, j) → (2i)



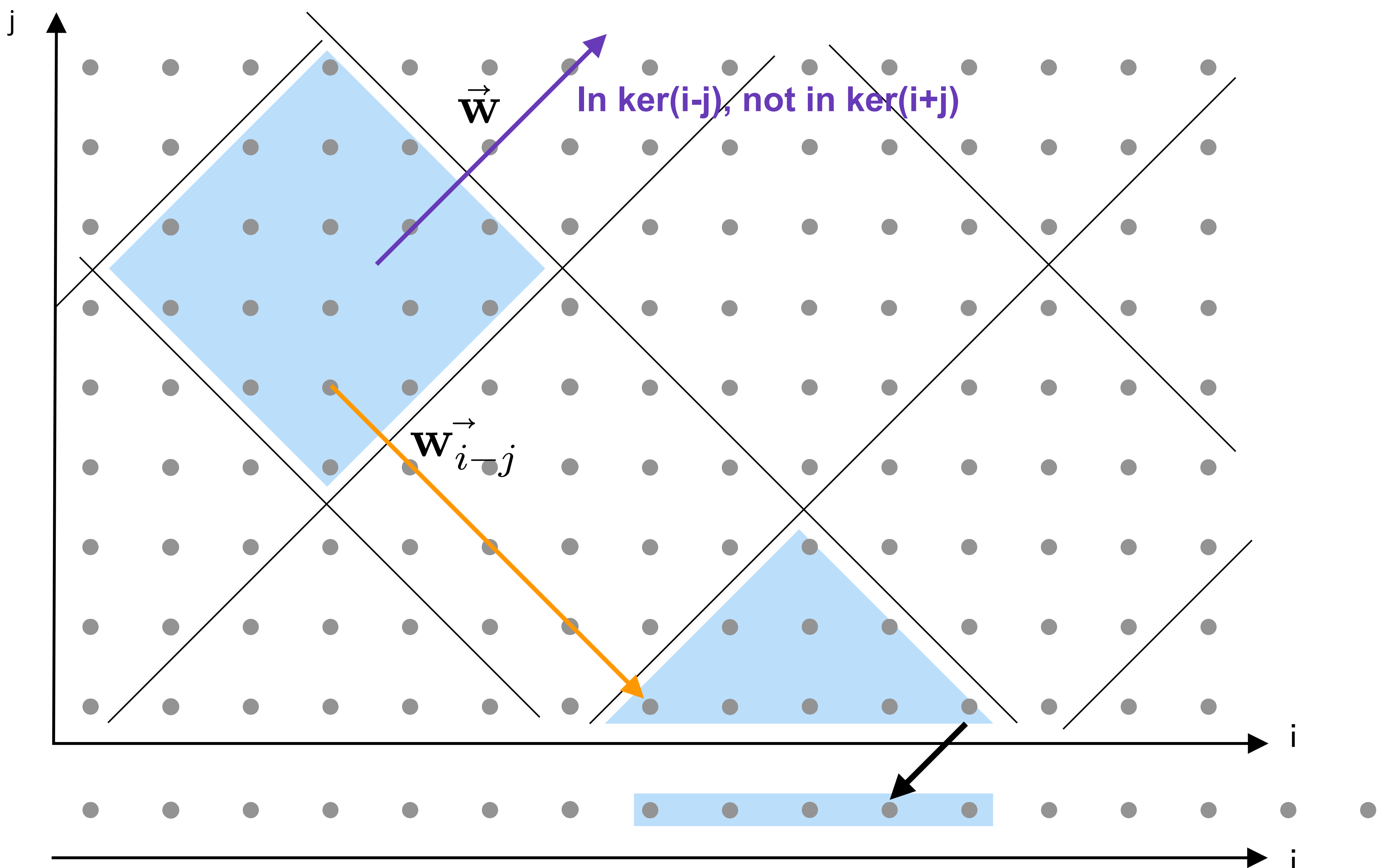
Non-uniformly intersecting dependences

Example : (i, j) → (i+j) ; (i, j) → (i-j)



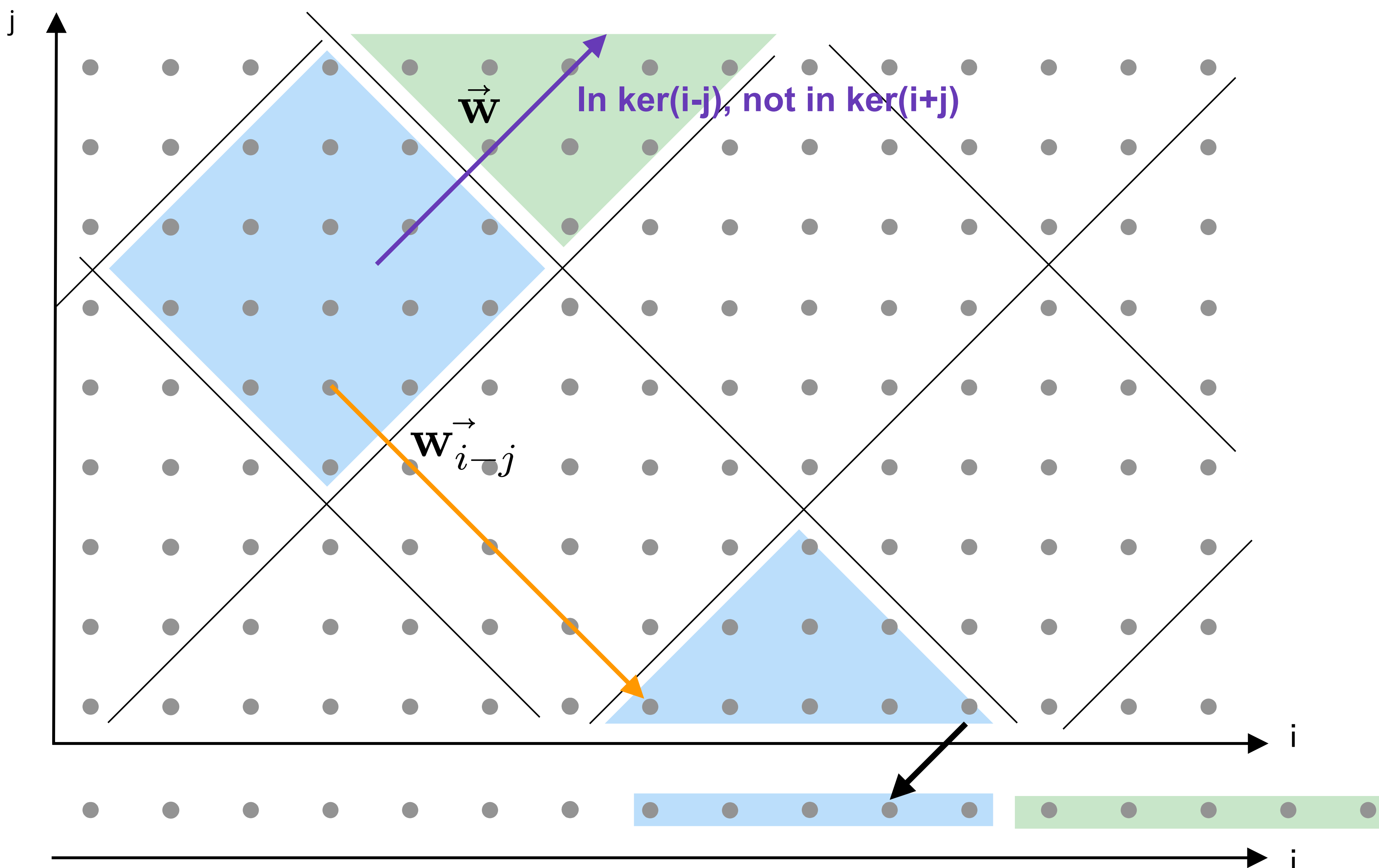
Non-uniformly intersecting dependences

Example : (i, j) → (i+j) ; (i, j) → (i-j)



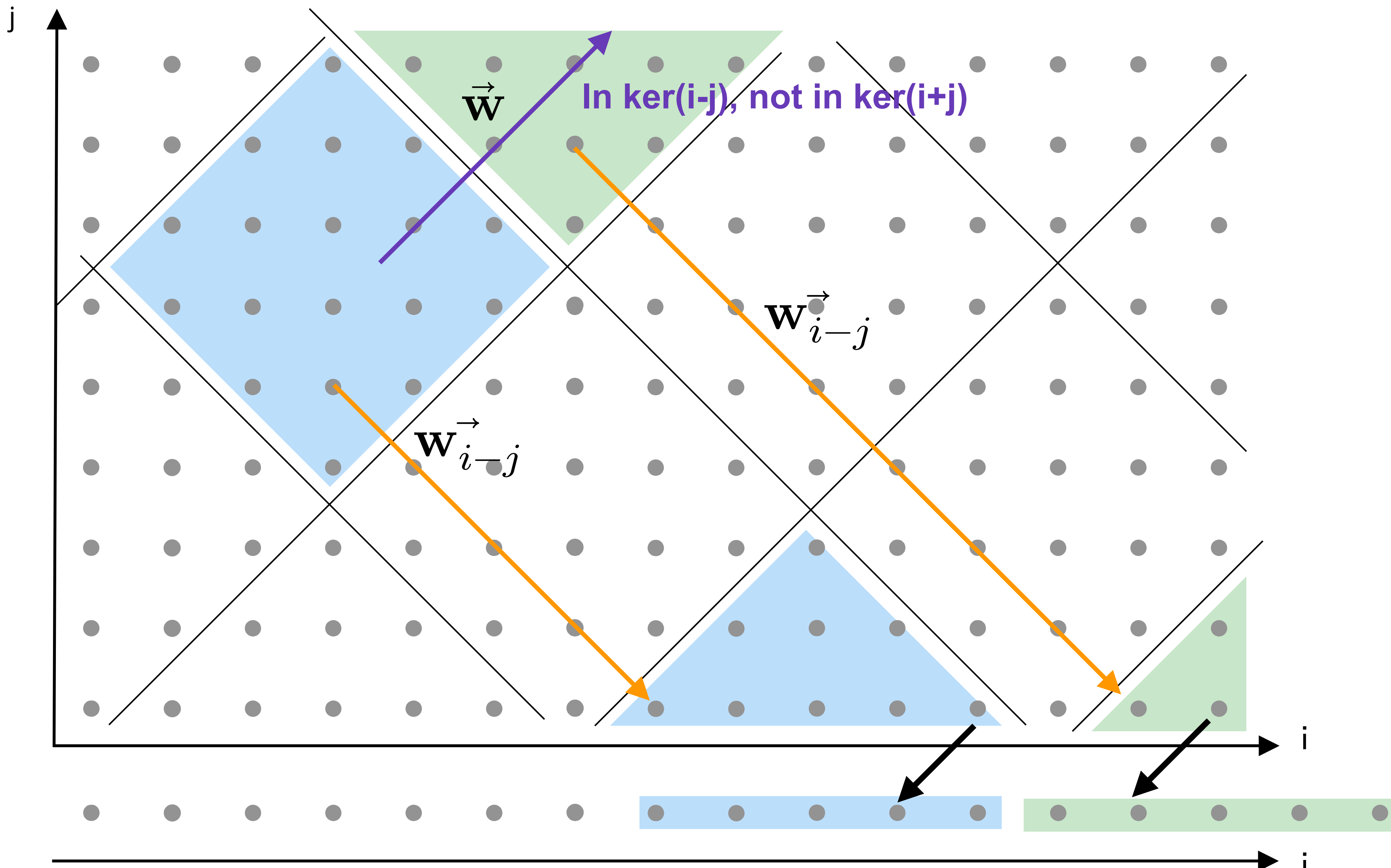
Non-uniformly intersecting dependences

Example : (i, j) → (i+j) ; (i, j) → (i-j)



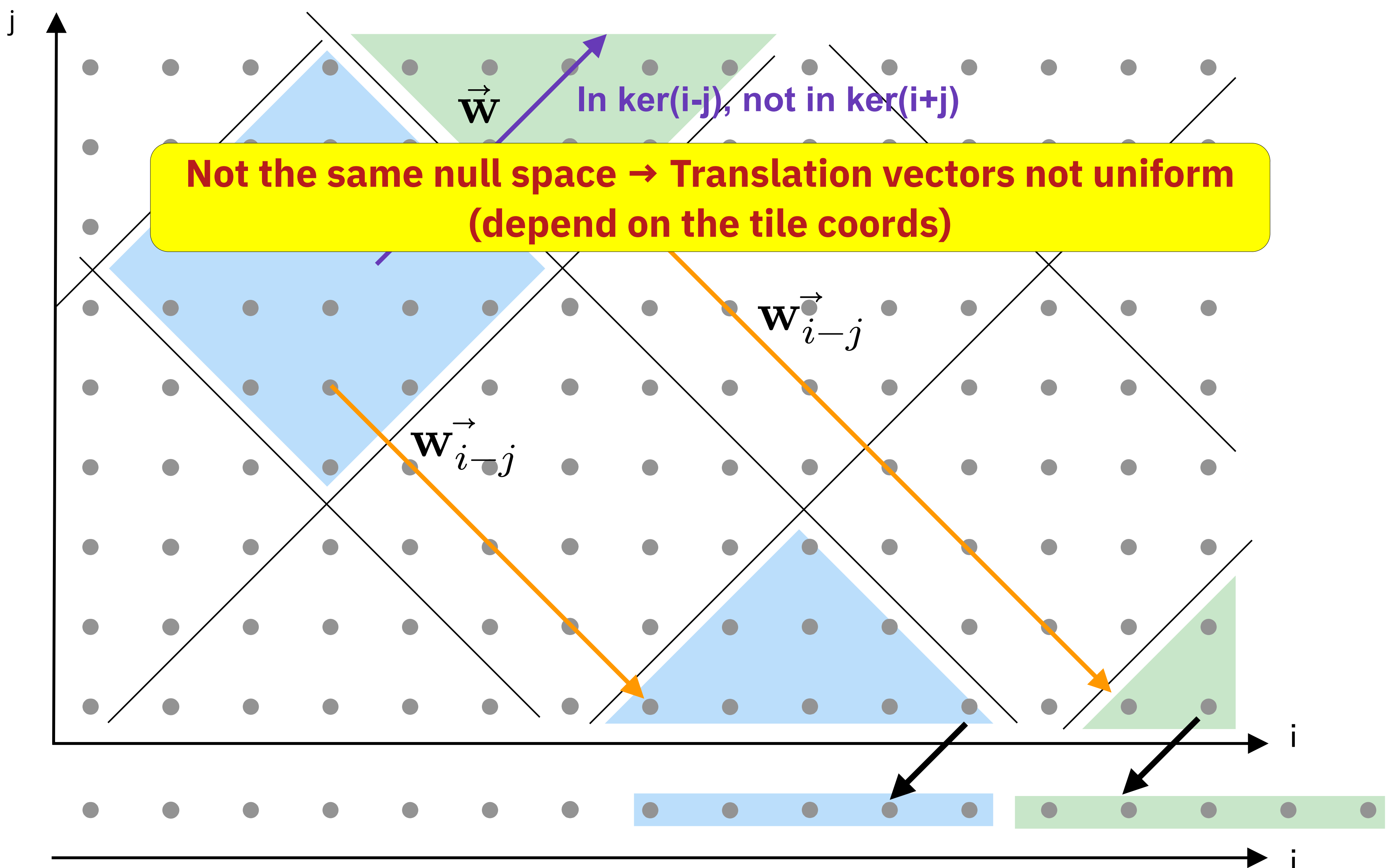
Non-uniformly intersecting dependences

Example : (i, j) → (i+j) ; (i, j) → (i-j)



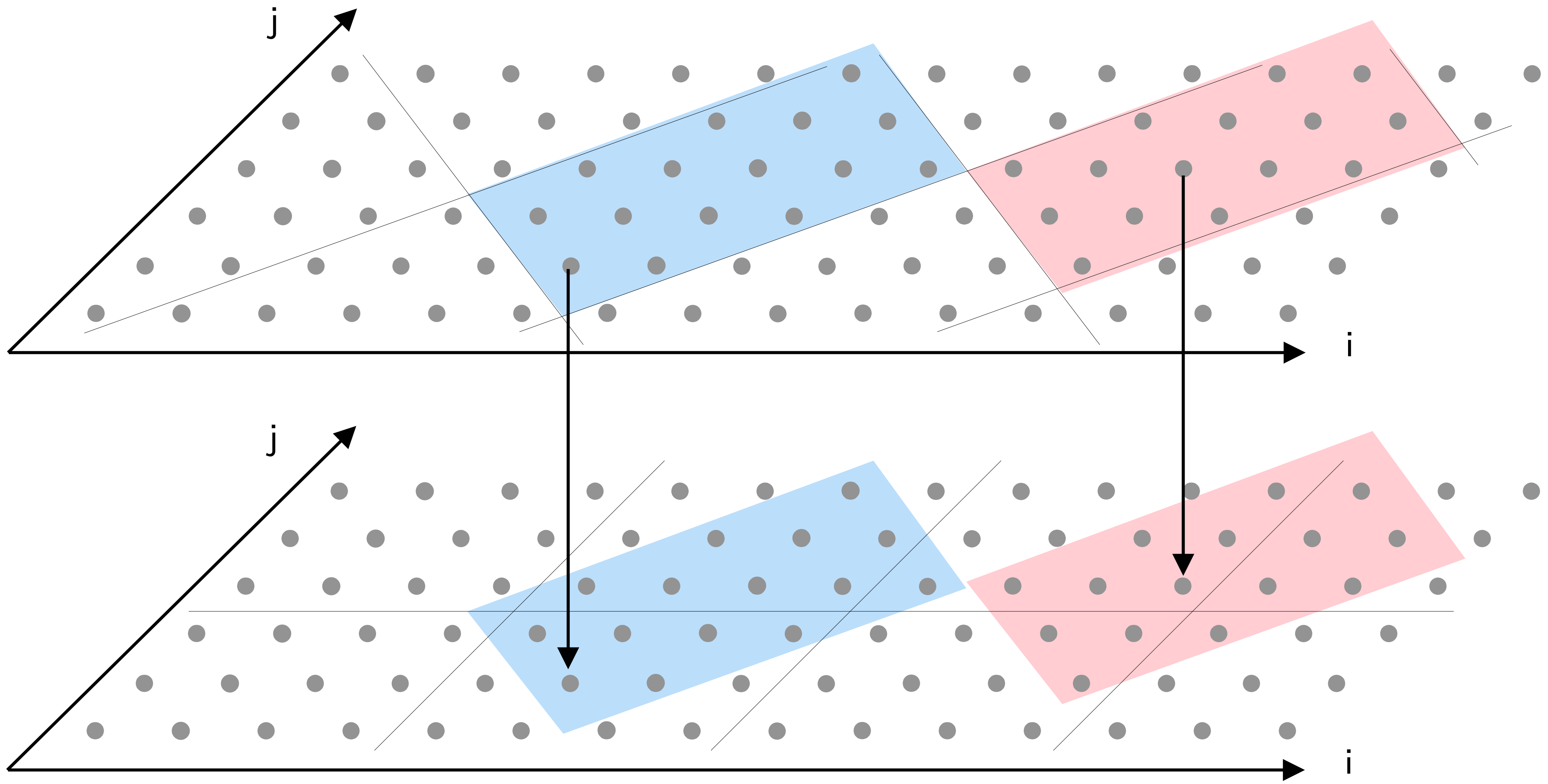
Non-uniformly intersecting dependences

Example : (i, j) → (i+j) ; (i, j) → (i-j)



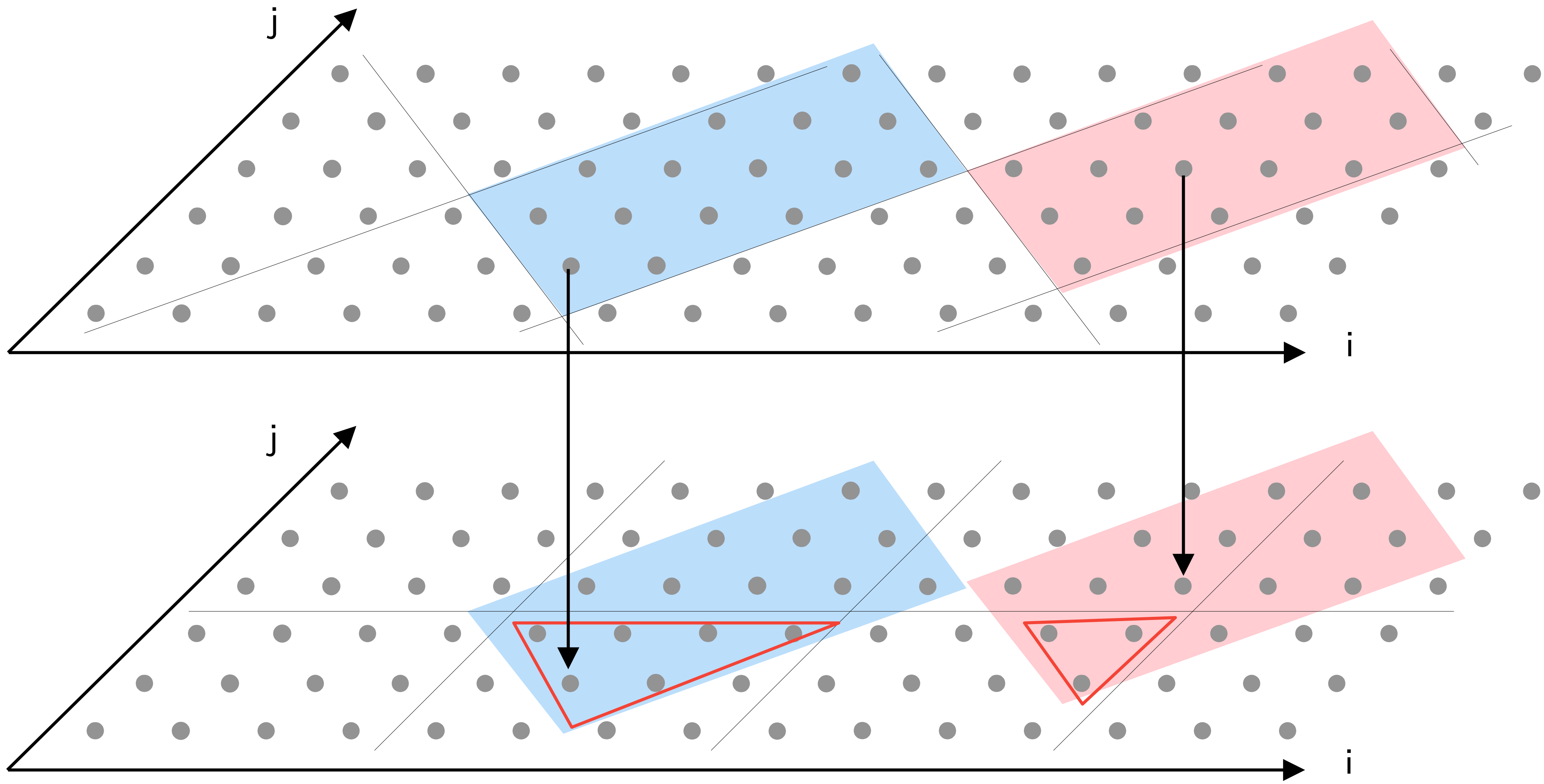
Dependences between tiled spaces?

Example : (i, j) → (i, j) (identity!)



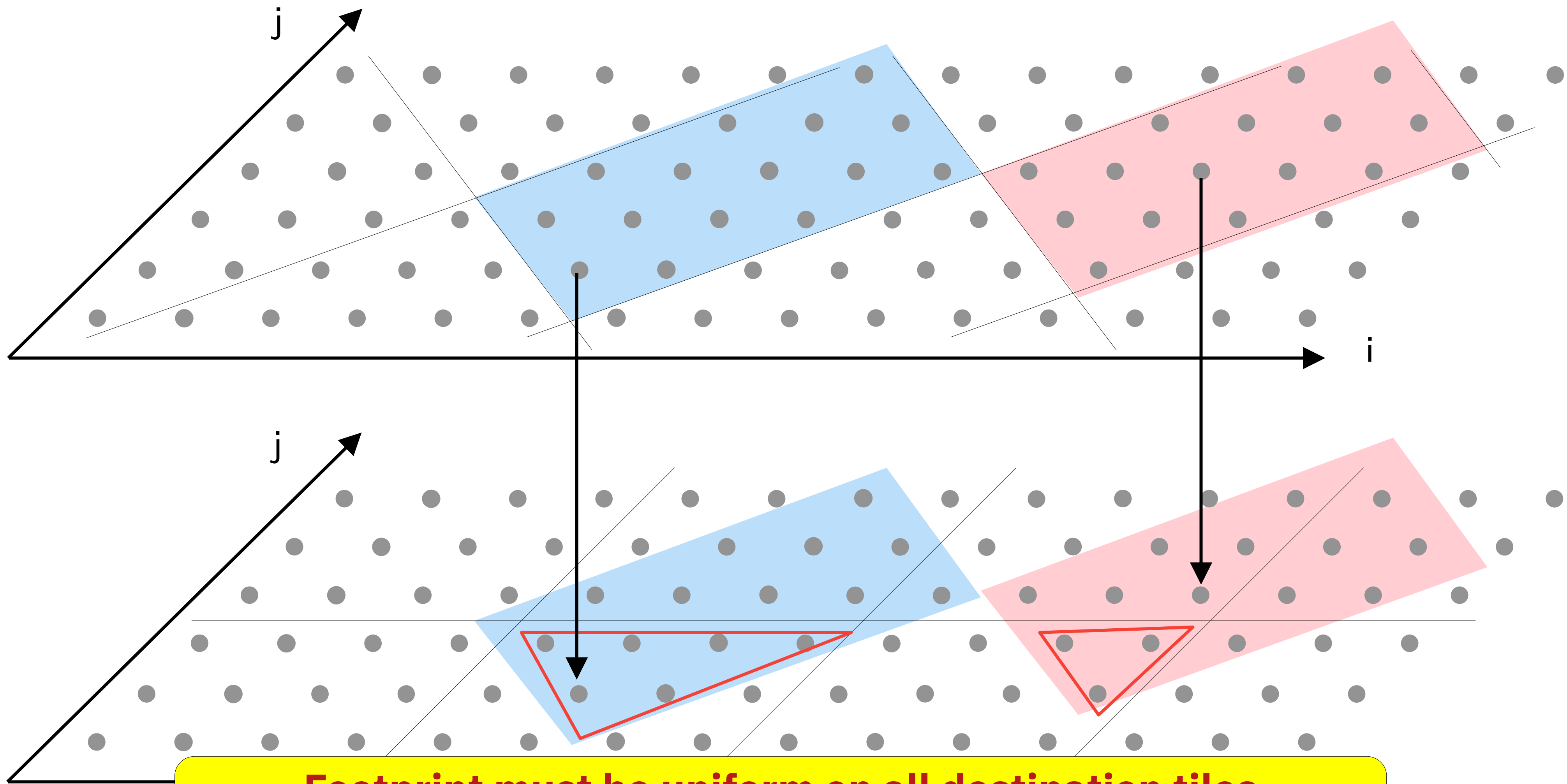
Dependences between tiled spaces?

Example : (i, j) → (i, j) (identity!)



Dependences between tiled spaces?

Example : (i, j) → (i, j) (identity!)



Footprint must be uniform on all destination tiles
→ Trying to express this using translations

Conclusions

- Partitioning data arrays with dependences for spatial locality
- **Generalizing** the idea of MARS
  - Bringing in affine dependences
- **Theoretical study**
  - Potential applications : anything that's tiled...
  - **Need to close two gaps** : find out when footprints are invariant for...
    Different linear part, same null space
    Tiled iteration spaces

Thank you