Persistence Algorithms

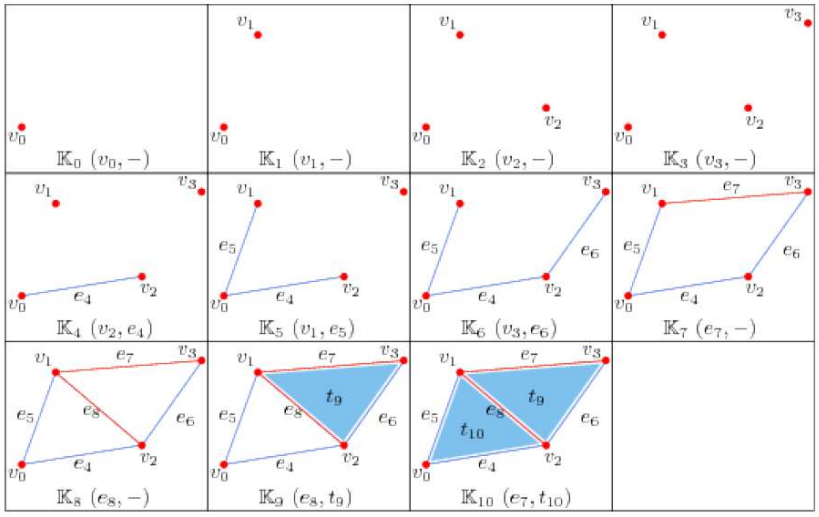
$K_0 \subseteq K_1 \subseteq K_2 \cdots \subseteq K_n = K$ Simplex-wise

p-simplex $\sigma_j = K_j \setminus K_{j-1}$ Simplex added

Fact Two things happen

- A non-boundary cycle c with classes $[c]+h$ for any $h \in H_p(K_{j-1})$ is created (born). σ_j is a positive simplex

- A $(p-1)$-cycle c along with its class $[c]$ becomes boundary (dies). σ_j is a negative simplex.



- When a positive p-simplex added, β_p increases by 1
  When e_8 is added, two cycles $e_5+e_8+e_4$ and $(e_7+e_6+e_8)$ are created, but only one is independent.

$$e_7+e_6+e_8 = (e_5+e_4+e_6+e_7)+(e_5+e_8+e_4)$$

- There is no canonical way of choosing which cycle is created

- For a negative p-simplex, we get opposite effect. β_{p-1} decreases by 1.
- For negative p-simplex, we can identify the cycle $c=\partial\sigma_j$ to be destroyed

Combinatorial Algorithm (Pairing)

1. Let $c=\partial\sigma_j$; c was created when its 'youngest' $(p-1)$-simplex σ_i was added

σ_i ('youngest' latest in the filtration)

2. If σ_i was paired already with the negative simplex σ_j', then σ_j' destroyed c' created by σ_i

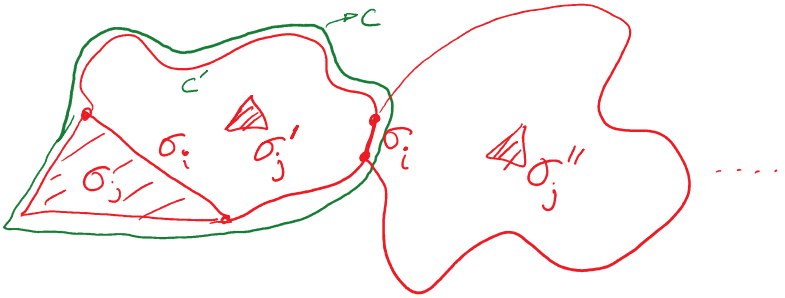
3. Consider cycle $c''=c+c'$. Set σ_i to the youngest

$(p-1)$-simplex in C'', set $C := C''$ and continue.

4. If C is empty, then σ_j is a positive simplex else if σ_i is unpaired, pair (σ_i, σ_j).

Why does it terminate?

- Every time we get a new σ_i, it appears earlier in the filtration



Algorithm 2 PAIRPERSISTENCE($\sigma_1, \sigma_2, \cdots, \sigma_n$)

Input:
An ordered sequence of simplices forming a filtration of a complex
Output:
Determine if a simplex is 'positive' or 'negative' and generate persistent pairs

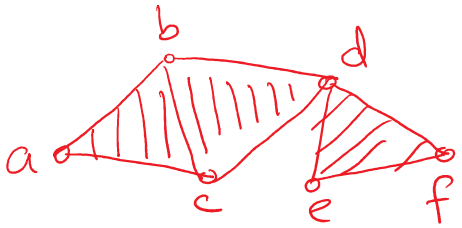
1: **for** $j = 1$ to n **do**
2: $c = \partial_p \sigma_j$
3: σ_i is the youngest positive $(p-1)$-simplex in c
4: **while** σ_i is paired and c is not empty **do**
5: Let c' be the cycle destroyed by the simplex paired with σ_i * set in step 10 *\
6: $c = c' + c$ * this addition may cancel simplices *\
7: Update σ_i to be the youngest positive $(p-1)$-simplex in c
8: **end while**
9: **if** c is not empty **then**
10: σ_j is a negative p-simplex; generate pair (σ_i, σ_j); associate c with σ_j as destroyed
11: **else**
12: σ_j is a positive p-simplex * σ_j may get paired later *\
13: **end if**
14: **end for**

Fact A p-simplex σ_i pairs with a $(p+1)$-simplex σ_j iff $\mu_p^{ij} > 0$.

Persistence algorithms (Matrix reduction)

- $\partial_p : C_p \to C_{p-1}$ can be represented by a boundary matrix D_p:

$$D_p[i,j] = \begin{cases} 1 & \text{if } \sigma_i = \partial_p \sigma_j \\ 0 & \text{otherwise} \end{cases}$$

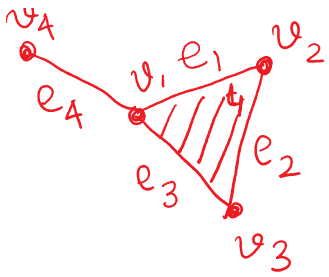


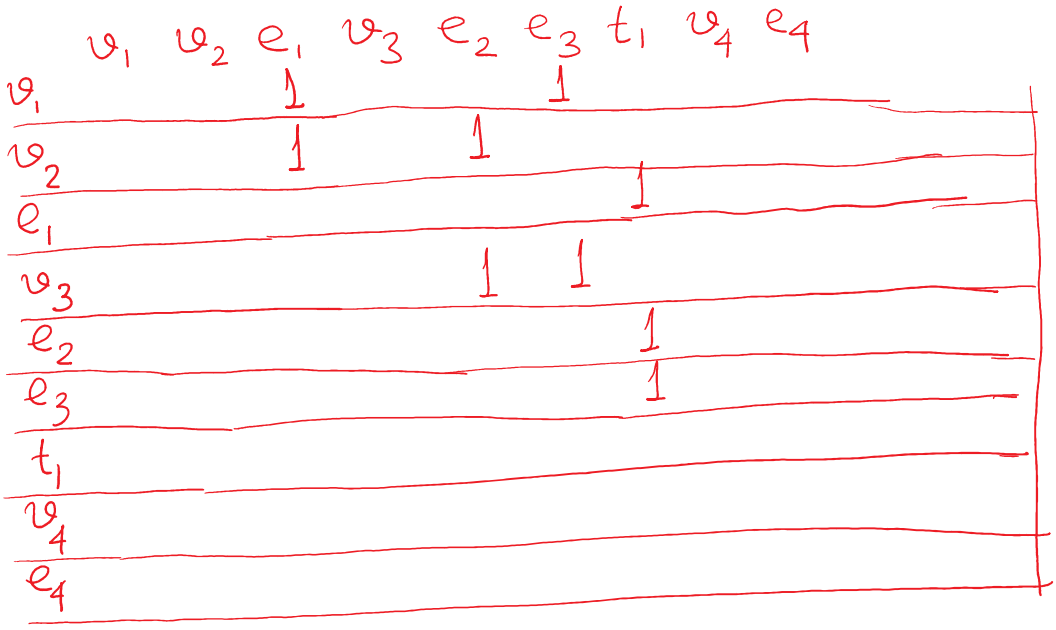
D_2:

	abc	bcd	def
ab	1	0	0
cd	0	1	0
ac	1	0	0
bc	1	1	0
bd	0	1	0
de	0	0	1
df	0	0	1
ef	0	0	1

- Combine all D_p's to get D

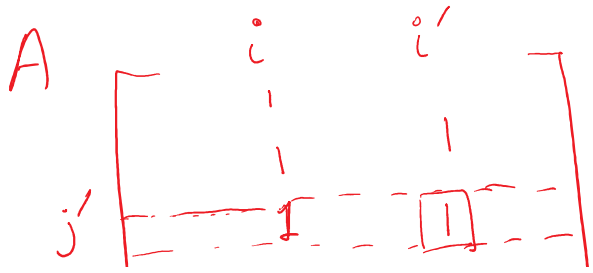
$$D[i,j] = \begin{cases} 1 & \sigma_i = \partial_* \sigma_j \\ 0 & \text{otherwise} \end{cases}$$





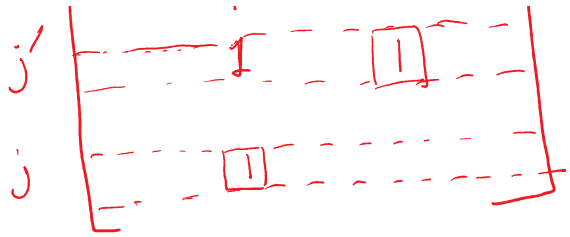
Def (Filtered matrix): $(\sigma_1, \sigma_2, \ldots, \sigma_m)$ filtered sequence of simplices. σ_i occupies row and column i of D.

- $low_A[i]$: denotes the index of the row containing lowest 1 in column i. For empty column, it is undefined.



$low_A[i] = j$

$low_A[i'] = j'$



Def (Reduced matrix): Matrix A is reduced if $low_A[j] \neq Low_A[j']$ for any $j \neq j'$.

Theorem 19. *Let D be the $m \times m$ filtered boundary matrix for a filtration $\mathcal{F}$ (Definition 69). Let $R = DV$, where R is in reduced form and V is upper triangular. Then, the simplices σ_i and σ_j in $\mathcal{F}$ form a persistent pair if and only if $\text{low}_R[j] = i$.*

* Multiplying D with an upper triangular matrix V on right is equivalent to obtain R from D by only left-to-right column additions.

Matrix Reduction

Algorithm 3 MATPERSISTENCE(D)

Input:
Boundary matrix D of a complex with columns and rows ordered by a given filtration
Output:
Reduced matrix with each column j either being empty or having a unique $\text{low}_D[j]$ entity

```
1: for j = 1 → |col_D| do
2:     while ∃j' < j s.t. low_D[j'] == low_D[j] and low_D[j] ≠ −1 do
3:         col_D[j] := col_D[j] + col_D[j']
4:     end while
5:     if low_D[j] ≠ −1 then
6:         i := low_D[j] \* generate pair (σ_i, σ_j) *\
7:     end if
8: end for
```

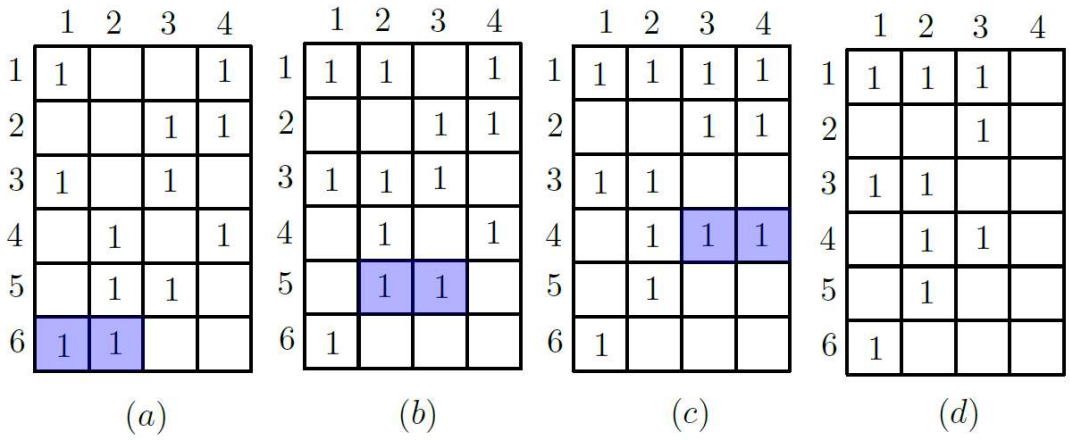


Figure 3.11: Matrix reduction for a 6×4 matrix D: low of columns are shaded to point out the conflicts. (a) $\text{low}_D[1]$ conflicts with $\text{low}_D[2]$ and $\text{col}_D[1]$ is added to $\text{col}_D[2]$, (b) $\text{low}_D[2]$ conflicts with $\text{low}_D[3]$, (c) $\text{low}_D[3]$ conflicts with $\text{low}_D[4]$, (d) the addition of $\text{col}_D[3]$ to $\text{col}_D[4]$ zero out the entire column $\text{col}_D[4]$.

- We can reorder the simplices without changing persistence pairs.
  - σ_j^i is the j-th i-simplex in the order

$$(\sigma_1^0, \sigma_2^0, \ldots \sigma_{n_0}^0) \ldots (\sigma_1^p, \sigma_2^p, \ldots, \sigma_{n_p}^p) \ldots (\sigma_1^d, \sigma_2^d, \ldots \sigma_{n_d}^d)$$

- Now we can operate on each matrix $D_p = [\partial_p]$ separately

- Complexity of MatPersistence is $O(n^3)$ where n simplices are inserted in the filtration.
  - Explain why $O(n^3)$

- The complexity can be brought down to $O(n^\omega)$ where $\omega < 2.373$ is the exponent for matrix multiplication.

Clearing

- If p-simplex σ^p is positive its column is zeroed out
- Zeroing out is more costly because generally it involves more column additions.
- A zeroed out column do not participate in any further column addition.
- If we knew (σ^{p-1}, σ^p) is a pair while processing D_p, then we can eliminate processing the column of σ^{p-1} in D_{p-1} because it is already known to be positive.

- Clearing: process D_p, $p=1,2,\cdots,d$ in reverse order of dimension. If (σ^{p-1}, σ^p) is a pair in D_p processing, ignore the column

of σ^{p-1} in D_{p-1}

Algorithm 4 ClearPersistence($D_1, D_2, \ldots, D_d$)

Input:
Boundary matrices ordered by dimension of the boundary operators with columns ordered by filtration

Output:
Reduced matrices with each column for negative simplex having a unique low entry

```
1: MatPersistence(D_d)
2: for i = (d − 1) → 1 do
3:    for j = 1 → |col_{D_i}| do
4:       if σ_j is not paired while processing D_{i+1} then
5:          \* column j is not processed if σ_j is already paired*\
6:          while ∃j' < j s.t. low_D[j] ≠ −1 and low_{D_i}[j'] == low_{D_i}[j] do
7:             col_{D_i}[j] := col_{D_i}[j] + col_{D_i}[j']
8:          end while
9:          if low_D[j] ≠ −1 then
10:            k := low_{D_i}[j] \* generate pair (σ_k, σ_j) *\
11:         end if
12:      end if
13:   end for
14: end for
```

- Still the last matrix D_d has to be processed usually. Sometimes this highest dimensional boundary matrix processing can be costly.
  (Ex: Rips filtrations : edge-vertex much smaller than highest dimensional matrix)
- For this we do a trick : D_p^* is D_p^T with rows and columns ordered reversed.

Fact: (σ^{p-1}, σ^p) is a pair while processing D_p iff (σ^p, σ^{p-1}) is a pair while processing D_p^*.

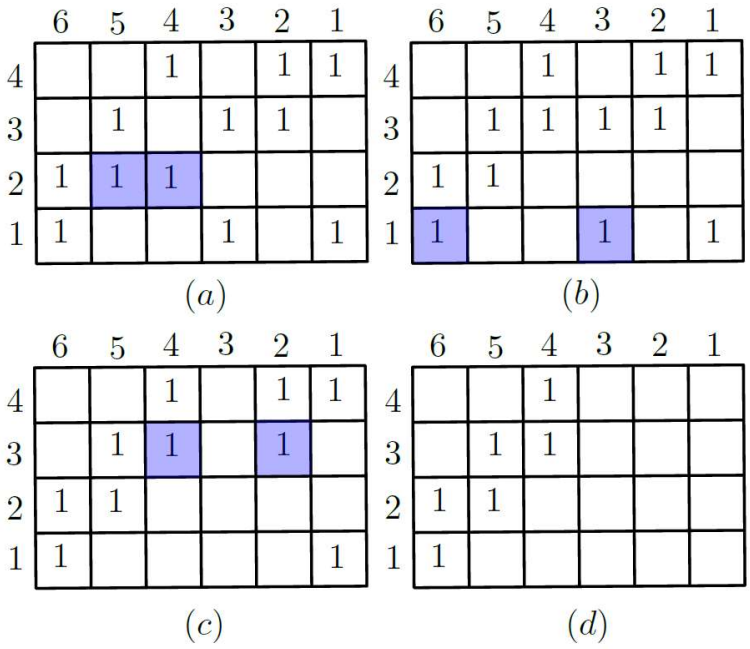


Figure 3.12: Matrix reduction with the twisted matrix D^* of the matrix D in Figure 3.11 which is first transposed and then got its rows and columns reversed in order; the conflicts in $\text{low}_D[\cdot]$ are resolved to obtain the intermediate matrices shown (a) through (d); The last transformation from (c) to (d) assumes to complete all conflict resolutions from columns 3 through 1. Observe that every column-row pair correspond to row-column pair in the original matrix. Also, all columns that are zeroed out here correspond to all rows in the original that did not get paired with any column meaning that they are either negative simplex, or positive simplex not paired with any.

- Now we process $D_1^*, D_2^*, \ldots D_d^*$ in increasing order of dimension.

- Because if (σ^{p+1}, σ^p) is a pair in D_p^*, then σ^{p+1} is a negative simplex and its column in D_{p+1}^* cannot contain any defined low entry.
- Call ClearPersistence $(D_1^*, D_2^*, \ldots D_d^*)$

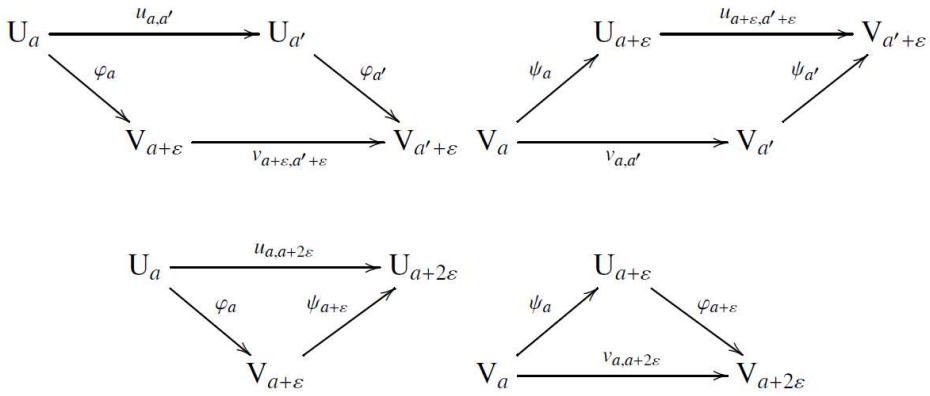
Persistence Modules and Interleaving

Definition 71 (persistence module)**.** A *persistence module* over an index set $A \subseteq \mathbb{R}$ is any collection $\mathbb{V} = \{V_a\}_{a\geq 0}$ of vector spaces V_a together with linear maps $v_{a,a'} : V_a \to V_{a'}$ so that $v_{a,a} = id$ and $v_{a',a''} \circ v_{a,a'} = v_{a,a''}$ for all $a, a', a'' \in A$ where $0 \leq a \leq a' \leq a''$. Sometimes we write $\mathbb{V} = \{V_a \xrightarrow{v_{a,a'}} V_{a'}\}_{0\leq a\leq a'}$ to denote this collection with the maps.

Definition 72 (ε-interleaving)**.** Let $\mathbb{U}$ and $\mathbb{V}$ be two persistence modules over an index set $A \subseteq \mathbb{R}$. We say $\mathbb{U}$ and $\mathbb{V}$ are ε-interleaved if there exist two families of maps $\varphi_a : U_a \to V_{a+\varepsilon}$ and $\psi_a : V_a \to U_{a+\varepsilon}$ satisfying the following two conditions:

1. $v_{a+\varepsilon,a'+\varepsilon} \circ \varphi_a = \psi_{a'} \circ u_{a,a'}$ and $u_{a+\varepsilon,a'+\varepsilon} \circ \psi_a = \varphi_{a'} \circ v_{a,a'}$ [rectangular commutativity]
2. $\psi_{a+\varepsilon} \circ \varphi_a = u_{a,a+2\varepsilon}$ and $\phi_{a+\varepsilon} \circ \psi_a = v_{a,a+2\varepsilon}$ [triangular commutativity]

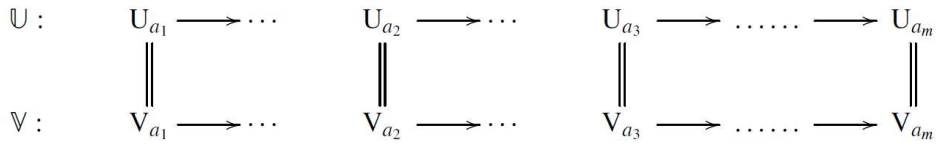
The two parallelograms and the two triangles below depict the rectangular and the triangular commutativities respectively.



Definition 73 (interleaving distance)**.** Given two persistence modules $\mathbb{U}$ and $\mathbb{V}$, their interleaving distance is defined as

$$d_I(\mathbb{U}, \mathbb{V}) = \inf\{\varepsilon \mid \mathbb{U} \text{ and } \mathbb{V} \text{ are } \varepsilon\text{-interleaved}\}$$

Observe that, when $\varepsilon = 0$, Definition 72 implies that the maps $\varphi_a : \mathrm{U}_a \to \mathrm{V}_a$ and $\psi_a : \mathrm{V}_a \to \mathrm{U}_a$ are isomorphisms. In that case, we get the following diagrams where each vertical map is an isomorphism and each square commutes. We get two isomorphic persistence modules.



Definition 74 (isomorphic persistence modules)**.** We say two persistence modules $\mathbb{U}$ and $\mathbb{V}$ indexed over an index set A are isomorphic if the following two conditions hold (illustrated by the diagram above).

1. $\mathrm{U}_a \cong \mathrm{V}_a$ for every $a \in A$, and
2. for every $x \in \mathrm{U}_a$, if x is mapped to $y \in \mathrm{V}_a$ by the isomorphism, then $u_{a,a'}(x) \in \mathrm{U}_{a'}$ is mapped to $v_{a,a'}(y) \in \mathrm{V}_{a'}$ also by the isomorphism.

Interval decomposition

Definition 76 (Interval module)**.** Given an index set $A \subseteq \mathbb{R}$ and a pair of indices $b, d \in A$, four types of *interval modules* denoted $\mathbb{I}[b,d), \mathbb{I}(b,d], \mathbb{I}[b,d], \mathbb{I}(b,d)$ respectively are special persistence modules defined as:

- (closed-open): $\mathbb{I}[b,d) : \{\mathrm{V_a} \xrightarrow{v_{a,a'}} \mathrm{V_{a'}}\}_{a,a' \in A}$ where $\mathrm{V}_a = \mathbb{Z}_2$ for all $a \in [b,d)$ and $V_a = 0$ and $v_{a,a'}$ is identity map for $b \le a \le a' < d$ and zero map otherwise.
- (open-closed): $\mathbb{I}(b,d] : \{\mathrm{V_a} \xrightarrow{v_{a,a'}} \mathrm{V_{a'}}\}_{a,a' \in A}$ where $\mathrm{V}_a = \mathbb{Z}_2$ for all $a \in (b,d]$ and $v_{a,a'}$ is identity map for $b < a \le a' \le d$ and zero map otherwise.
- (closed-closed): $\mathbb{I}[b,d] : \{\mathrm{V_a} \xrightarrow{v_{a,a'}} \mathrm{V_{a'}}\}_{a,a' \in A}$ where $\mathrm{V}_a = \mathbb{Z}_2$ for all $a \in [b,d]$ and $v_{a,a'}$ is identity map for $b \le a \le a' \le d$ and zero map otherwise.
- (open-open): $\mathbb{I}(b,d) : \{\mathrm{V_a} \xrightarrow{v_{a,a'}} \mathrm{V_{a'}}\}_{a,a' \in A}$ where $\mathrm{V}_a = \mathbb{Z}_2$ for all $a \in (b,d)$ and $v_{a,a'}$ is identity map for $b < a \le a' < d$ and zero map otherwise.

Proposition 23.

- *Any persistence module over a finite index set decomposes uniquely into interval modules, that is,* $\mathbb{U} = \bigoplus_{j \in J} \mathbb{I}\langle b_j, d_j \rangle$ *[154].*
- *Any p.f.d. persistence module decomposes uniquely into interval modules [99, 269].*
- *Any q-tame persistence module decomposes uniqule into interval modules [73].*

Definition 77 (PD for persistence module)**.** Let $\mathbb{U} = \bigoplus_j \langle b_j, d_j \rangle$ be the interval decomposition of a given persistence module $\mathbb{U}$ (Proposition 23). The collection of points $\{(b_j, d_j)\}$ with proper multiplicity and the points on the diagonal $\Delta : \{(x, x)\}$ with infinite multiplicity constitute the persistence diagram $\mathrm{Dgm}\mathbb{U}$ of the persistence module $\mathbb{U}$.

Theorem 24. *Given two q-tame persistence modules defined over the totally ordered index set* $\mathbb{R}$, $\mathsf{d}_I(\mathbb{U}, \mathbb{V}) = \mathsf{d}_b(\mathrm{Dgm}\,\mathbb{U}, \mathrm{Dgm}\,\mathbb{V})$.

- Comment on [b,d] intervals vs. [b,d+1) PD point as we did so far for persistence filtrations.