

<https://github.com/lauriluca99/LAR-TGW-3D>

#### LINEAR ALGEBRAIC RAPRESENTATION:

LAR è uno schema rappresentativo per modelli geometrici e topologici, il dominio di questo schema consiste in complessi di cellule formati a loro volta da matrici sparse (matrici con grande affluenza di zeri). L'analisi di questi complessi cellulari è fatta attraverso semplici operazioni algebriche lineari, la più comune è la moltiplicazione sparsa matrice/vettore.

Dato che LAR permette una computazione efficiente di qualsiasi modello topologico è stato ritenuto necessario affiancarlo ad un linguaggio, anch'esso efficiente e veloce, che sia in grado di sfruttare tutte le sue potenzialità.

*Perché LAR?*

Scegliamo LAR in quanto l'aumento della complessità dei dati geometrici e dei modelli topologici richiedono una migliore rappresentazione e un modello matematico appropriato per tutte le strutture topologiche è proprio un complesso co-chain formato da collezioni di matrici sparse.

Un complesso chain consiste in una sequenza di moduli dove la singola immagine di ognuno è contenuta nel nucleo della successiva. (successivo conosce precedente)

$$\cdots \xleftarrow{d_0} A_0 \xleftarrow{d_1} A_1 \xleftarrow{d_2} A_2 \xleftarrow{d_3} A_3 \xleftarrow{d_4} A_4 \xleftarrow{d_5} \cdots$$

Figure 1: This is the caption

Un complesso co-chain è la stessa cosa ma con direzioni opposte.

$$\cdots \xrightarrow{d^{-1}} A^0 \xrightarrow{d^0} A^1 \xrightarrow{d^1} A^2 \xrightarrow{d^2} A^3 \xrightarrow{d^3} A^4 \xrightarrow{d^4} \cdots$$

Figure 2: This is the caption

#### TGW-3D:

L'algoritmo Topological Gift Wrapping ha come obiettivo il calcolare le d-celle di una partizione di spazio generate da loro partendop da un oggetto geometrico d-1 dimensionale.

TGW prende una matrice sparsa di dimensione d-1 in input e produce in output la matrice sparsa di dimensione d sconosciuta aumentata dalle celle esterne.

#### SPATIAL ARRANGEMENT:

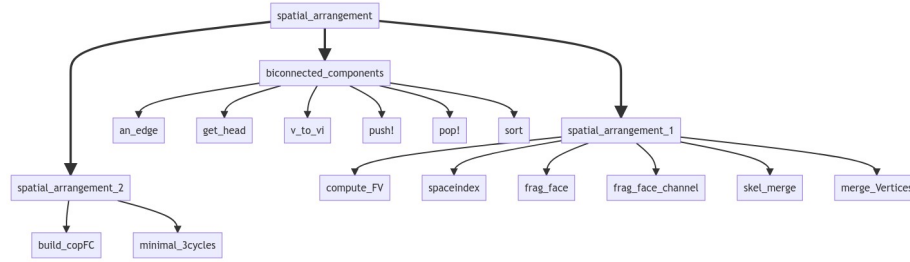


Figure 3: This is the caption

Calcola la partizione dei complessi cellulari dati, con scheletro di dimensione 2, in 3D.

Un complesso cellulare è partizionato quando l'intersezione di ogni possibile paio di celle del complesso è vuota e l'unione di tutte le celle è l'insieme dello spazio Euclideo. La funzione ritorna la partizione complessa come una lista di vertici  $V$  e una catena di bordi EV, FE, CF.

*SPATIAL\_ARRANGEMENT\_1:*

Si occupa del processo di frammentazione delle facce per l'utilizzo del planar arrangement.

1. `compute_FV`:

Ritorna l'array FV di tipo `Lar.Cells` dal prodotto di due array sparsi in input di tipo `Lar.ChainOp`.

2. `Spaceindex`:

Dato un modello geometrico, calcola le intersezioni tra i bounding box. Nello specifico, la funzione calcola le 1-celle e il loro bounding box attraverso la funzione `boundingBox`. Si suddividono le coordinate  $x$  e  $y$  in due dizionari chiamando la funzione `coordintervals`. Per entrambe le coordinate  $x$  e  $y$ , si calcola un `intervalTree` cioè una struttura dati che contiene intervalli. La funzione `boxCovering` viene chiamata per calcolare le sovrapposizioni sulle singole dimensioni dei bounding Box. Intersecando quest'ultime, si ottengono le intersezioni effettive tra bounding box. La funzione esegue lo stesso procedimento sulla coordinata  $z$  se presente. Infine, si eliminano le intersezioni di ogni bounding box con loro stessi.

3. `frag_face`:

Effettua la trasformazione in 2D delle facce fornite come parametro `sigma`, dopo di che ogni faccia `sigma` si interseca con le facce presenti in `sp_index` sempre fornito come parametro della funzione.

4. `skel_merge`:

effettua l'unione di due scheletri che possono avere 1 o 2 dimensioni.

5. `merge_vertices`:

effettua l'unione dei vertici, dei lati e delle facce vicine.

*BICONNECTED\_COMPONENTS*:

calcola le componenti biconnesse del grafo EV rappresenato da bordi, ovvero coppie di vertici.

1. `an_edge`:

funzione che, dato in input un punto, prende un lato connesso ad esso.

2. `get_head`:

funzione che, dato in input un lato e la coda, fornisce la testa

3. `v_to_vi`:

funzione che, dato un vertice in input, ritorna falso se la prima occorrenza della matrice è pari a 0 oppure ritorna il valore trovato.

4. `push!`:

inserisce uno o più oggetti nella matrice.

5. `pop!`:

rimuove l'ultimo oggetto nella matrice e lo ritorna.

6. `sort`:

ordina la matrice e ne ritorna una copia.

*SPATIAL\_ARRANGEMENT\_2*:

Effettua la ricostruzione delle facce permettendo il wrapping spaziale 3D.

1. `minimal_3cycles`:

funzione che riporta i parametri dati in input in 3 dimensioni e calcola le nuove celle adiacenti per estendere i bordi della figura geometrica. Infine ritorna la matrice sparsa tridimensionale.

2. `build_copFC`:

funzione alternativa alla precedente.