Studio Preliminare Progetto LAR Splitting 2D – CPD22

Gruppo: 5.a – Castagnacci Giulia 581749, Giordano Elisabetta 536265

28 Apr 2022

Repository GitHub: https://github.com/GiuliaCastagnacci/LARSplitting2D.git

Contents

Premessa **1**

Analisi delle funzioni classe refactoring **1**

Grafo delle dipendenze **3**

Premessa

Generazione di una disposizione 2D del piano euclideo (quindi di un complesso di catene) a partire da ciascuna faccia (2 celle) di un insieme di forme 3D. O da un insieme di primitive 1D (linee, poligoni, cerchi, ecc.)

Input: una struttura Lar, da trasformare in un insieme di poligoni (2D o 1D) o una matrice degli stessi oggetti.

Output: una raccolta di complessi di catene 2D (disposizioni 2D), incorporato in due matrici sparse diagonali a blocchi.

Analisi delle funzioni classe refactoring

https://github.com/GiuliaCastagnacci/LARSplitting2D/blob/main/src/refactoring.jl

- **crossingTest:** Funzione di supporto per pointInPolygonClassification. Utilizzata per aggiornare il valore di "count" a seconda dello stato, new o old (entrambi di tipo Int), incrementando o decrementando il count di 0.5.
- **setTile:** Funzione utilizzata per impostare il tileCode di una bounding box 2D includendo il point 2D di coordinate x e y. A seconda della posizione di "point", il tileCode varia da 0 a 15. Serve per impostare la tile del piano in

base alla posizione del punto di query, per testare successivamente i codici del tile dei bordi di un poligono 2D e determinare se tale punto è interno o esterno o sul confine del poligono. Questa funzione si deve parallelizzare.

- **pointInPolygonClassification:** Funzione utilizzata per la classificazione dei punti di un poligono. In particolare, identifica se un punto si trova all'interno, all'esterno o sul bordo di un poligono, richiama la setTile per dirlo.
- **input_collection:** Prende in input un array di tipo Lar.LAR e viene utilizzata per la selezione di una faccia e per la costruzione di una collezione di dimensione d-1, da una di dimensione d o d-1. Ritorna in output il risultato dell'applicazione della funzione Lar.struct2lar(), la quale valuta un oggetto geometrico di tipo Lar.Struct, generando la tripla (V, FV, EV). Il risultato dato può essere dato in input ad algoritmi di pipeline 2D-3D.
- **boundingbox:** Prende in input un vertice di tipo Lar.Points e restituisce in output due array che indicano gli estremi del bounding box. Serve, quindi, a creare il bounding box di un segmento, cioè la scatola entro cui sono contenuti tutti i punti.
- **coordIntervals:** Prende in input una matrice (i bounding box) e un intero che specifica la coordinata su cui si lavora. La funzione restituisce un dizionario ordinato, dove la chiave è l'intervallo sulla singola coordinata, mentre il valore associato è un array di indici che indicano a quali bounding box si riferiscono.
- **boxcovering:** Prende in input una matrice (i bounding box), un intero che specifica la coordinata su cui si lavora, e un intervalTrees. La funzione restituisce una matrice che contiene tutte le intersezioni tra bounding box.
- **spaceIndex:** Prende in input una tupla costituita da una matrice che contiene i punti del modello, di tipo Lar.LAR, e da un array di array che contiene le celle cioè scomposizioni dello spazio geometrico. La funzione restituisce in output un array di array dove l'elemento i-esimo rappresenta quali intersezioni ha il bounding box i-esimo con gli altri 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.
- **Intersection:** Funzione utilizzata per l'intersezione di due segmenti di linea in 2D, Restituisce in output i due parametri di linea del punto di intersezione. I segmenti si intersecano se entrambi i parametri di ritorno,

alpha e beta, sono contenuti nell'intervallo [0, 1].

- **linefragments:** Funzione che ordina i segmenti intersecati di ogni segmento. Per ogni segmento, se interseca con qualcosa, prende i punti (x1, y1), (x2, y2) del segmento i-eisimo, lo confronta con tutti gli altri segmenti presenti nel suo indice spaziale "Sigma" che fornisce il sottoinsieme di linee i cui box di contenimento intersecano il box di ciascuna linea di input (dato da EV), e restituisce i parametri (alfa e beta) necessari a determinare il punto di intersezione tra coppie di segmenti. Se le rette si intersecano, si verifica che i parametri alfa e beta siano ammissibili, se lo sono vengono immagazzinati nella struttura dati "params".
- **fragmentlines:** Funzione utilizzata per l'intersezione di coppie di segmenti di linea. In particolare, crea un indice spaziale "Sigma", calcola i parametri d'intersezione degli spigoli e restituisce i nuovi punti generati dall'intersezione tra spigoli, tramite i parametri d'intersezione, all'interno di un array. Per N punti d'intersezione trovati, vengono generati N-1 spigoli.
- **fraglines:** Prende in input tre variabili di tipo float, rappresentanti il piano cartesiano. Restituisce un modello che passa in input alla funzione fragmentlines.
- **congruence:** Prende in input un model di tipo Lar.LAR, inizializza un BallTree, che divide ricorsivamente i punti in gruppi delimitati da ipersfere, un raggio di ricerca e un array vuoto di W elementi (W = matrice 2xW). Per ogni vertice cerca i vertici più vicini nel raggio R e li sostituisce. Crea un dizionario chiave, valore (id nuovi vertici) che verrà poi utilizzato per etichettare i vertici degli spigoli in EW.

Grafo delle dipendenze

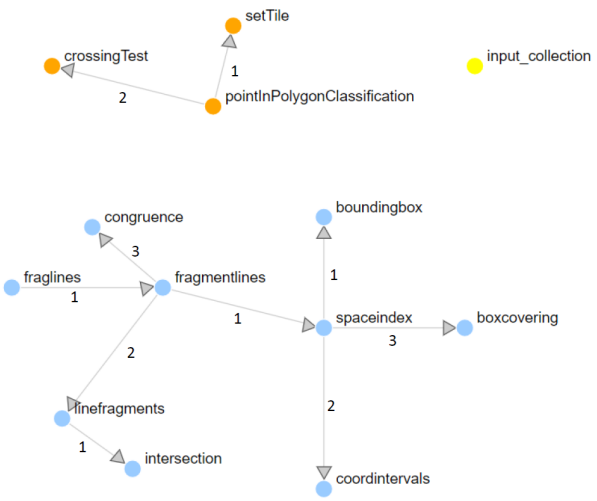

Figure 1: Grafo delle dipendenze refactoring