



UNIVERSITÀ DEGLI STUDI DI CATANIA

DIPARTIMENTO DI MATEMATICA E INFORMATICA

CORSO DI LAUREA MAGISTRALE IN INFORMATICA

GRAPHLOW:

SIMULATORE GRAPH-BASED

DI FLUSSI LAVICI SULL'ETNA

RELAZIONE PROGETTO

Andrea Busacca - Matricola: W82000138

Andrea Sequenzia - Matricola: W82000160

Indice

1	Introduzione	3
2	Analisi di dati	4
3	Downsampling	8
4	Algoritmo di aggregazione	9
5	Costruzione del grafo	10
5.1	Visualizzazione con Gephi	11
6	Algoritmo Eruption	12
6.1	GUI	14
6.2	Visualizzazione con QGIS	15
7	Conclusioni	17

1. Introduzione

L'obiettivo di GRAPHLOW è quello di ottenere un simulatore efficiente e semplificato dei flussi lavici sull'Etna. L'idea di base consiste nel modellare un grafo sulla topologia dell'Etna, creando dei dati e dei pesi basati su simulazioni in **MAGFLOW** [2]. Nella prossima sezione di questa relazione, viene mostrata l'analisi sui dati forniti dall'Istituto Nazionale di Geofisica e Vulcanologia di Catania. Successivamente si passa a descrivere la fase di processing per snellire i dati ed aggregarli il più possibile. Dopo questa fase, viene definita la fase di creazione del grafo e della sua visualizzazione. Infine, si spiega in che modo si è scelto di emulare un'eruzione a partire da una bocca specifica.

2. Analisi di dati

Per questo progetto sono stati forniti i seguenti dati:

1. DEM_CT
2. simulazioni eseguite con **MAGFLOW**

Il file DEM contiene una matrice 2275x1875 dove ogni cella indica l'altezza sul livello del mare. Nell'header sono contenute le coordinate in Easting e Northing della zona corrispondente all'Etna. Da queste coordinate si è potuto estrarre da Google Maps la zona corrispondente, mostrata in Figura 1. Usando i dati all'interno della matrice, è stata creata una mappa delle altezze



Figura 1: Zona indicata dalle coordinate del DEM

in scala di grigio (Figura 2).

Le simulazioni eseguite in **MAGFLOW** sono 28.402. Il nome di ognuna di esse indica la coordinata della bocca nella griglia all'interno della simulazione e la classe dell'eruzione (1-6). Le classi sono descritte nell'articolo [3] e vengono mostrate in Figura 3. Ogni riga di un file indica le coordinate di una cella del DEM che viene inondata. Queste simulazioni sono state usate per



Figura 2: Mappa delle altezze

creare una mappa di pericolosità (Figura 4) che fa corrispondere ad ogni pixel un valore nella scala di grigio, proporzionale al numero di simulazioni che inondano quel pixel. La creazione di queste mappe è stata implementata nello script *init_map.py*.

Dopo aver creato queste mappe, si è passati ad implementare una struttura dati chiamata lin-

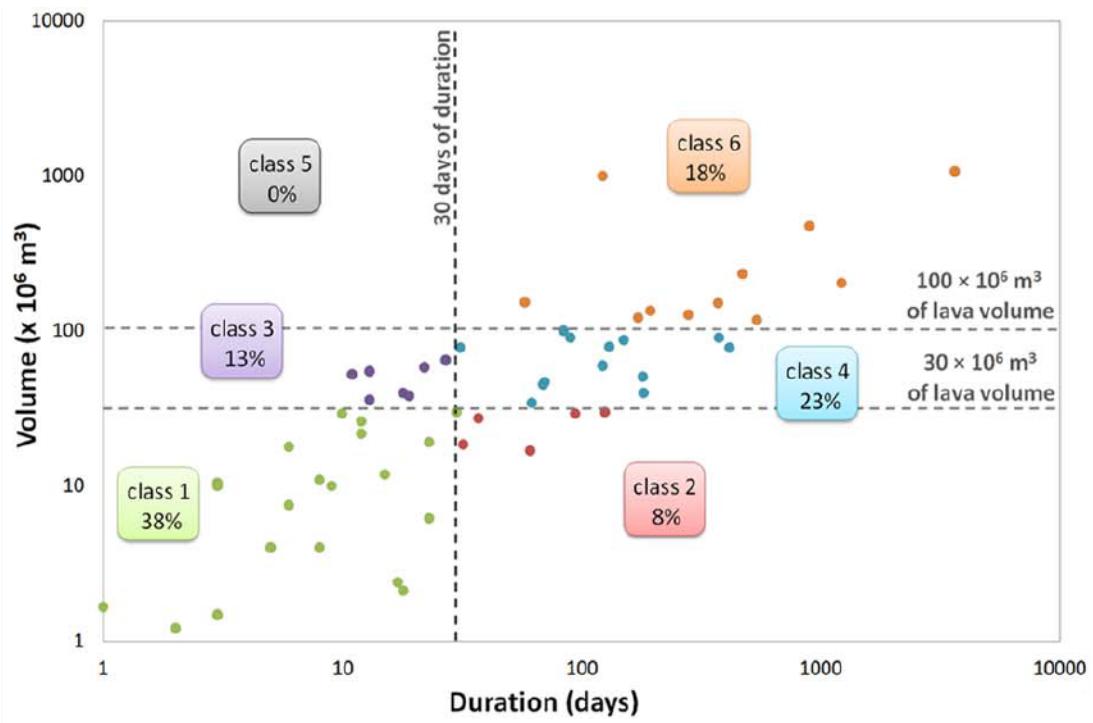


Figura 3: Classi eruttive

ked_map e successivamente salvarla in formato CSV. Ad ogni riga di questo file CSV corrisponde una coordinata del DEM. Ad esempio, la prima riga corrisponde alla coordinata (0,0), la seconda alla coordinata (0,1) e così via. Per ogni riga vengono indicati gli id delle simulazioni che hanno invaso quella coordinata e la classe della colata (l'ultima cifra della cella, da 1 a 6). Un esempio che illustra delle righe all'interno della linked_map è mostrato in Tabella 2.

13152	13153	13154	13155	13156	13151	
14611	14612	14613	14614	14615	14616	13881
14611	14612	14613	14614	14615	14616	

Tabella 1: Esempio Linked Map

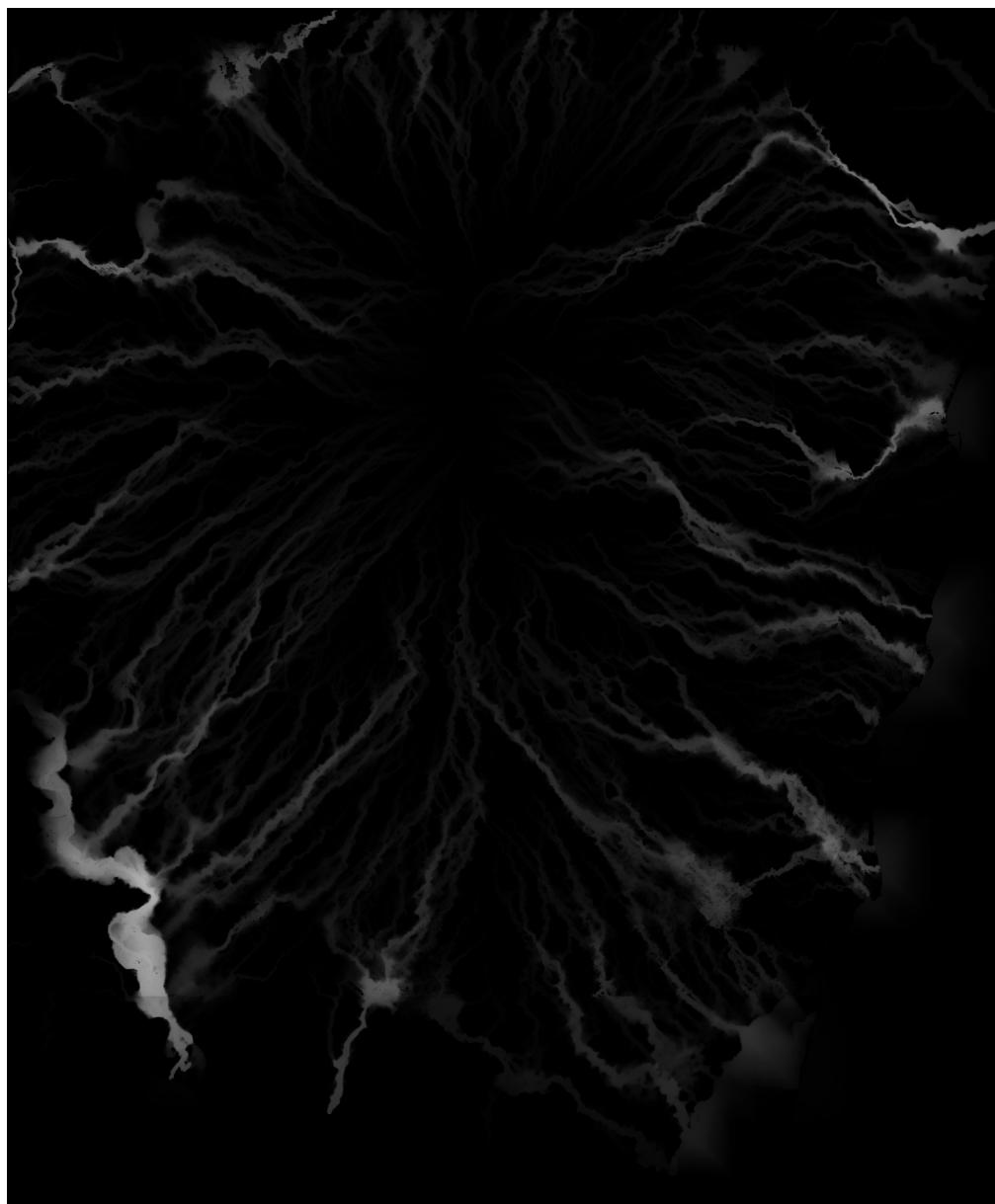


Figura 4: Mappa di pericolosità

3. Downsampling

In questa fase del lavoro, l'obiettivo prefissato è stato quello di minimizzare il numero di nodi del grafo. Per farlo, è stato scritto un metodo che scala la mappa del file DEM di un fattore 25. In questo modo la dimensione della matrice è passata da 2275x1875 a 91x75.

È stata creata la classe Region (regione da ora in poi) con vari attributi:

- **coord**: una coppia di coordinate (x,y) che permette di tenere traccia della posizione dell'oggetto nella mappa originale;
- **marked**: un booleano che permette di marcare un oggetto. Viene utilizzato nelle fasi successive;
- **sim**: attributo di tipo *set* (si è scelta questa struttura dati per questioni puramente di efficienza computazionale) che contiene gli ID di tutte le simulazioni eseguite con *MAGFLOW* che inondano la cella associata;

Sono state quindi istanziate 6825 regioni (una per ogni cella della matrice) e ne sono stati inizializzati gli attributi.

4. Algoritmo di aggregazione

Per poter sfoltire il numero di regioni è stato scritto un algoritmo di aggregazione basato sul seguente criterio di equivalenza: due regioni vengono aggregate se e solo se queste hanno la stessa lista di simulazioni. Questo ha permesso di abbassare ulteriormente il numero di oggetti Region a 5820, aggregando soprattutto le regioni invase da nessuna simulazione.

L'idea dell'algoritmo è quella di visitare ogni regione ed esplorarne in ampiezza il vicinato di Moore. Sono state istanziate due code di supporto, chiamate rispettivamente: *green* e *red*. In *green* vengono inseriti i vicini che rispettano la relazione di equivalenza e in *red* tutti gli altri. Fino a quando la coda verde non viene svuotata si continua ad esplorare le regioni contenute in essa. Ogni qualvolta viene svuotata la coda verde si passa alla prossima regione che è stata inserita nella coda rossa e così via. Quando non ci sono più elementi nella coda rossa l'algoritmo termina e ritorna una lista chiamata *node_list*, in cui ogni elemento è costituito da due liste: *region_list* e *near_node_list*, che sono rispettivamente un insieme di regioni aggregate e un insieme di regioni vicine.

5. Costruzione del grafo

Viene inizializzato un grafo orientato vuoto, i cui nodi hanno un insieme di attributi:

- **id_node**: identificativo univoco del nodo;
- **(x,y)**: posizione del baricentro del nodo nella mappa DEM, determinata dall'algoritmo *get_median_position* che calcola la mediana delle posizioni x e y delle regioni;
- **height**: l'altitudine del nodo. Denotata con q_u , viene determinata con l'algoritmo *get_height* che calcola la mediana delle altezze delle regioni;
- **region_list**: lista di regioni che compongono i nodi;
- **coord_regions**: lista di coordinate associate a ciascuna delle regioni aggregate;
- **n_region**: numero di regioni del nodo;
- **near_node_list**: lista di nodi vicini, coi quali creare gli archi del grafo;
- **n_sim**: numero di simulazioni che inondano il nodo;
- **is_vent**: booleano che indica se il nodo contiene una bocca;
- **rank**: un numero intero che serve in una fase successiva a determinare l'avanzamento temporale del flusso lavico delle simulazioni MAGFLOW;
- **current_flow**: attributo perno per poter implementare le simulazioni GRAPHLOW;
- **awash**: booleano che, in fase di simulazione real-time, indica se il nodo viene inondato.

Ogni nodo viene creato scorrendo la *node_list* descritta in precedenza, inizializzandone i vari attributi e successivamente vengono creati gli archi grazie alla lista *near_node_list*.

Questi hanno i seguenti attributi:

- **u**: nodo da cui parte l'arco;

- **v**: nodo terminale dell’arco;
- **weight**: il peso dell’arco, determinato come $\omega(u, v) = |A \cap B| / |A|$, dove A è la lista delle simulazioni di u e B è la lista delle simulazioni di v;
- **transmit_rank**: denotato con $\phi^{(u)}$, è un attributo che serve in una fase successiva a determinare un’informazione temporale di una determinata simulazione;
- **slope**: determina la pendenza dell’arco;
- **forwarding_flow**: indica quanta lava passa in un certo istante dall’arco;
- **trasmittance**: è il parametro transmit_rank normalizzato rispetto agli altri archi uscenti dal nodo u. Viene eseguita come segue:

$$\gamma(u, v) = \frac{\phi_i^{(u)}}{\max(\phi_j^{(u)}) + \sum_{j=0}^7 \phi_j^{(u)}} \quad \text{dove} \quad \phi^{(u)} = \left\{ \phi_j^{(u)} : j = 0, \dots, 7, j \neq i \right\} \quad (1)$$

5.1. Visualizzazione con Gephi

Per visualizzare il grafo finora costruito, si è scelto di usare Gephi, un software open source per esplorare e manipolare grafi di grandi dimensioni [1]. Per fare ciò, si è dovuto esportare il grafo in formato *GEXF*¹. Grazie al plugin Geo Layout², è stato possibile visualizzare un layout coerente dal punto di vista geografico, usando le coordinate salvate come attributo nel grafo e utilizzando la proiezione cilindrica equidistante. Il risultato è mostrato in Figura 5.

Sono state fatte varie prove di visualizzazione, in modo semplice e immediato, per una migliore comprensione dei dati. Ad esempio, impostando la dimensione e il colore dei nodi in relazione all’attributo *height*, è stato ottenuto il risultato mostrato in Figura 6.

¹GEXF (Graph Exchange XML Format) è un formato usato per descrivere strutture di reti complesse con dei dati associati.

²<https://gephi.org/plugins//plugin/geolayout-plugin>

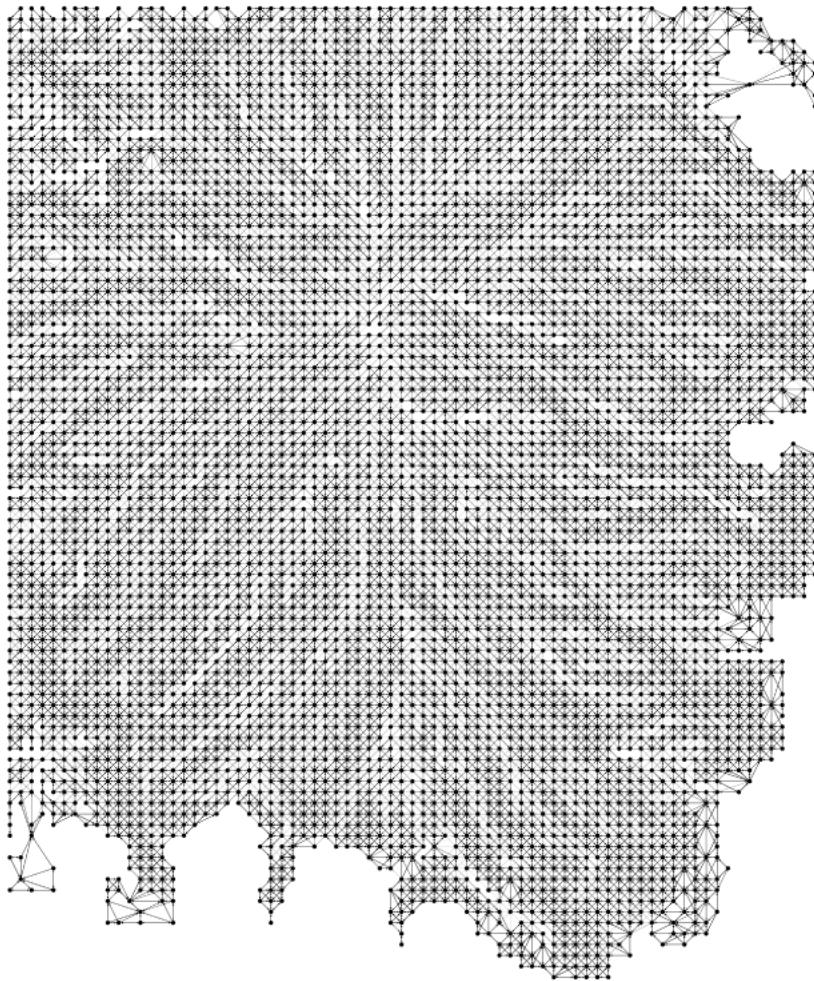


Figura 5: Visualizzazione del grafo in Gephi

6. Algoritmo Eruption

L'ultima fase del progetto è stata quella di cercare di riprodurre il possibile comportamento di un'eruzione, utilizzando il grafo precedentemente costruito. Per fare ciò, sono stati provati vari metodi. Di seguito, viene descritto solamente quello definitivo che ha dato i migliori risultati. Questo algoritmo è parametrizzato, permettendo all'utente di scegliere l'id della bocca, la quan-

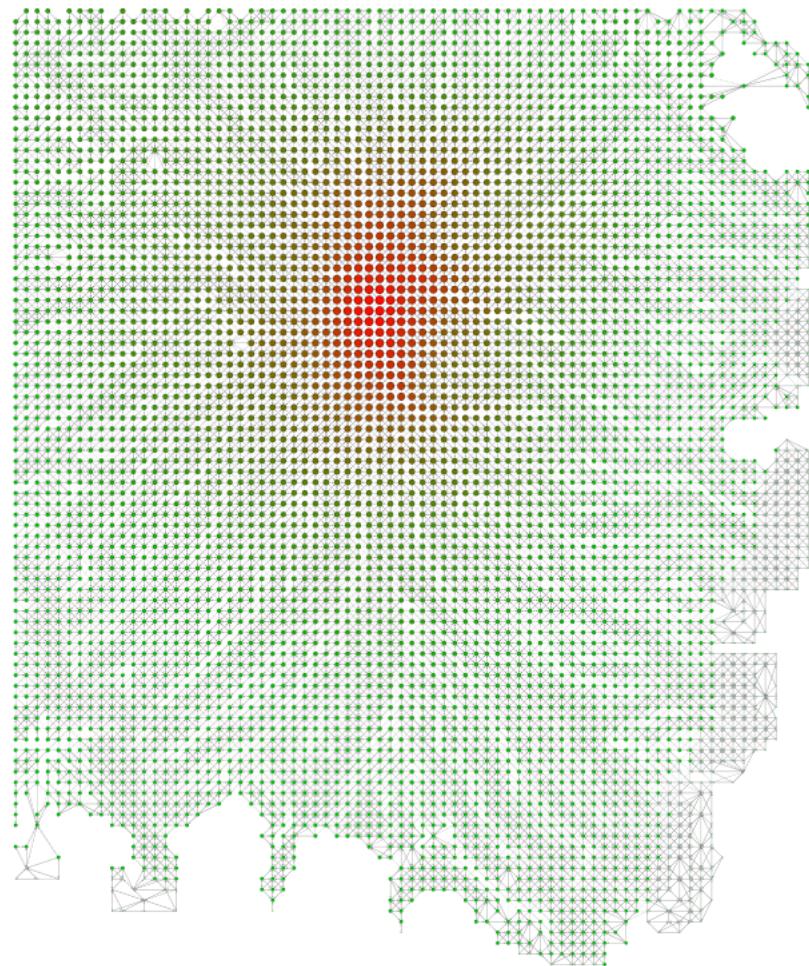


Figura 6: Visualizzazione heightmap sul grafo

tità di lava emessa dalla stessa e la durata in giorni. L’idea è quella di far partire l’eruzione dalla bocca scelta dall’utente (*id_vent*) e farle emettere della lava gradualmente. Una volta individuato il nodo della bocca, viene propagata la lava a tutti i nodi vicini.

Il principio di propagazione è il seguente:

siano

- h_u e h_v le quantità di lava contenute rispettivamente in u e v

- q_u e q_v le altitudini di u e v

allora u propaga a v se e solo se

$$\Delta_h > \beta \quad \wedge \quad h_u + q_u > h_v + q_v \quad (2)$$

dove

$$\Delta_h = h_u - h_v. \quad (3)$$

La quantità di lava che viene propagata da u a v è determinata come

$$\rho_{(u,v)} = \gamma(u,v) \cdot \alpha \cdot \Delta_h \cdot \frac{1}{1 + e^{-\tau}} \quad (4)$$

dove

- $\gamma(u, v)$ è la trasmittanza dell'arco (u, v)
- α è il parametro di smorzamento
- τ è il rapporto $\frac{h_u + q_u}{h_v + q_v}$

In (4) è stata utilizzata la funzione sigmoide per normalizzare tra 0 e 1 il valore τ , tale da rendere ρ proporzionale alla differenza delle altezze.

L'algoritmo itera su tutti i nodi in cui è stata propagata lava e termina quando nessuno di questi può propagare lava.

Infine, il grafo risultante viene esportato come mappa ASCII che è possibile importare su QGIS³ per la visualizzazione.

6.1. GUI

Per eseguire GRAPHLOW, da terminale bisogna spostarsi nel path ”*.../Volcano/*” e digitare il comando `python main.py`. Vengono messi a disposizione dell'utente quattro comandi:

- **man:** descrive i comandi e i relativi parametri;

³<https://www.qgis.org/>

- **showsim**: esporta la ASCII map di una simulazione MAGFLOW;
- **eruption**: esporta la ASCII map di una simulazione GRAPHLOW
- **exit**: esce dal programma.

```
===== Welcome in GRAPHLOW =====
\\
        ooo
        oo000o
        o000000oooo
        oo0000ooo  oooo
        /vvv\ \
        V V V \\
        V VV   AAAAH! / RUN FOR YOUR LIVES!
        VVV  \
        VVVV  \
        VVVVVV  \
        VVVVVVVV  \
Available commands:
- man
- eruption [id_vent] [volume] [n_days] [alpha] [threshold]
- showsim [id_vent] [class]
- exit
Insert a command >
```

Figura 7: Screenshot di GRAPHLOW

6.2. Visualizzazione con QGIS

QGIS è un Sistema di Informazione Geografica Open Source, rilasciato sotto la GNU General Public License ed è un progetto ufficiale della Open Source Geospatial Foundation (OSGeo). Per visualizzare una simulazione MAGFLOW e/o un'eruzione simulata con GRAPHLOW, come prima cosa viene caricato il DEM_CT e calcolata la shader map, per una visualizzazione più intuitiva della zona di riferimento. Successivamente viene caricata la mappa ASCII e visualizzata secondo una specifica scala di colori. Nell'esempio mostrato in Figura 8, si è scelto di visualizzare una simulazione con bocca 2233. Come simulazione MAGFLOW è stata selezionata quella di classe 6. Per calcolare la simulazione con GRAPHLOW sono stati utilizzati i seguenti parametri:

- **id_vent** 2233
- **volume** 10000
- **n_days** 7
- **alpha** default (0.125)
- **threshold** default (1)

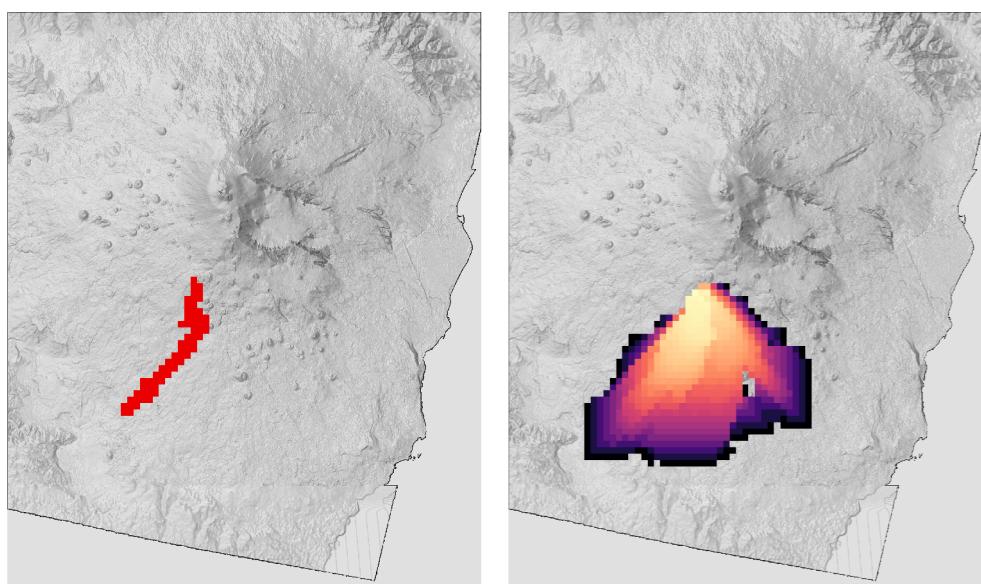


Figura 8: Comparazione simulazione in MAGFLOW (a sinistra) e in GRAPHLOW (a destra)

7. Conclusioni

Le simulazioni ottenute con GRAPHLOW non sono coincidenti a quelle ottenute con MAGFLOW. I motivi sono molteplici e noti. Il primo di questi è senz'altro dato dal fatto che non si tiene conto di nessun principio fisico del flusso lavico; la scarsa informazione locale ne è un'altra causa, perché i pesi degli archi sono calcolati globalmente.

Tuttavia, data la sua velocità di computazione, le simulazioni prodotte da GRAPHLOW possono essere usate come informazione di tipo probabilistico.

Per migliorare ulteriormente le prestazioni è possibile fare il porting del codice in C++ e cercare di parallelizzare alcune istruzioni.

Riferimenti bibliografici

- [1] Mathieu Bastian, Sébastien Heymann, and Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks, 2009.
- [2] Annalisa Cappello, Alexis Héault, Giuseppe Bilotta, Gaetana Ganci, and Ciro Del Negro. Magflow: a physics-based model for the dynamics of lava-flow emplacement. *Geological Society, London, Special Publications*, 426(1):357–373, 2016.
- [3] Ciro Del Negro, Annalisa Cappello, Marco Neri, Giuseppe Bilotta, Alexis Héault, and G. Ganci. Lava flow hazards at mount etna: Constraints imposed by eruptive history and numerical simulations. *Scientific reports*, 3:3493, 12 2013.