

Opzione 3: Simulazione di un'epidemia

Niccolò Ciavarelli
Gabriel Fenati
Alessandro Mancini

24 Giugno 2020

1 Descrizione delle principali scelte progettuali e implementative

1.1 Descrizione del programma

Con il programma realizzato ci si propone di simulare l'evoluzione di un'epidemia all'interno di una popolazione in cui sono inizialmente presenti un dato numero di individui definiti "suscettibili" (ossia infettabili) e un dato numero di individui infetti: coloro che daranno avvio all'epidemia.

Il progetto è raggiungibile mediante il seguente link:

https://github.com/CNFGMA/Project19_20

1.2 La classe Board: il cuore del programma

Alla base del programma c'è una griglia di dimensioni fissate, realizzata mediante una classe di nome Board, con un vettore riga contenente un numero costante di celle, che rappresentano gli individui di una popolazione chiusa e lo spazio che essi hanno a disposizione. La griglia possiede un bordo, che delimita la sua estensione e i movimenti degli individui che la abitano.

Ogni cella, se abitata, può inizialmente essere infetta o suscettibile, altrimenti è non occupata. Questi possibili stati sono rappresentati con una enum class.

1.3 L'evoluzione epidemica

Con lo scorrere del tempo, che avviene mediante un ciclo iterativo, la griglia evolve: ad ogni iterazione le celle suscettibili possono infettarsi e le celle infette possono essere ospedalizzate; una volta che esse sono ospedalizzate, indipendentemente dall'esito di guarigione/decesso, vengono indicate in modo unico (si veda ad esempio il modello SIR) come "recovered". L'evoluzione delle infezioni/guarigioni è gestita dalla funzione "evolve". Nel frattempo, le celle infette e suscettibili possono muoversi nella griglia con una certa probabilità verso le celle adiacenti non occupate. Il movimento è gestito dalla funzione "step".

Una volta che una cella è divenuta recovered non è più interessante dal punto di vista dell'evoluzione epidemica e di conseguenza rimane ferma. Si assume inoltre che una cella recovered non possa più infettare né essere infettata.

Al fine di simulare una sorta di quarantena locale, il movimento di una cella è

proibito nel caso vi sia una sfavorevole combinazione tra due dati: il numero di infetti adiacenti alla cella in questione e la percentuale di infetti sul totale della popolazione (si veda la funzione "quarantine").

Ogni cella suscettibile ha una probabilità d'infezione dipendente dal numero di infetti vicini e ogni cella infetta ha una probabilità di divenire recovered che aumenta con il tempo (si suppone un modello in cui con il passare del tempo, grazie alla sanità ed alla ricerca medica ad esempio, la guarigione sia sempre più probabile e il tracciamento dei contagi e la loro ospedalizzazione più efficienti). Tutto ciò è reso mediante due funzioni chiamate "gamma_t" e "beta_inf". Nel valutare l'evoluzione dello stato di ogni singola cella, al fine di simulare gli innumerevoli parametri di contingenza che possano aggravare o migliorare una certa situazione (buona risposta immunitaria, lenta guarigione, mantenimento della distanza interpersonale, ecc.), si relazionano in modo opportuno le succitate probabilità ad esiti (pseudo)casuali di distribuzioni uniformi (in particolare sono stati utilizzati generatori pseudo-casuali di tipo Mersenne Twister ed opportune distribuzioni di probabilità della libreria "random" di C++).

Ad ogni iterazione il programma stampa a schermo (su terminale) una rappresentazione della griglia con la situazione corrente dell'infezione: le celle infette sono rappresentate in colore rosso, quelle suscettibili in giallo, quelle recovered in verde.

Il programma termina quando l'epidemia si conclude, ossia quando non vi sono più infetti: a questo punto verrà restituito un file di output contenente i dati legati all'evoluzione dell'epidemia (si veda il paragrafo 4 per i dettagli).

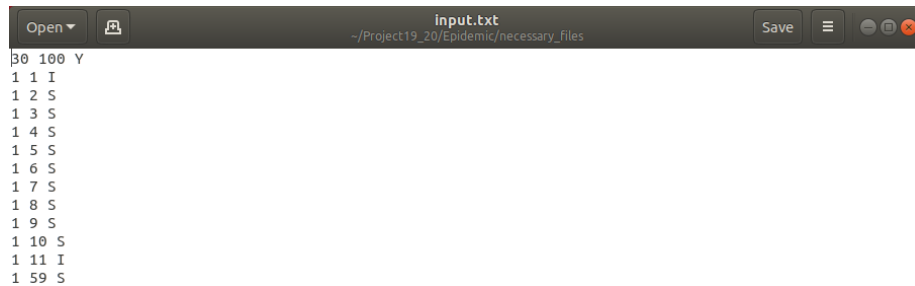
1.4 Organizzazione in Files

Il programma è diviso in codici sorgente (.cpp) e header files (.hpp). La suddivisione è basata sui rapporti tra le funzioni e le classi del programma.

Le funzioni gamma_t, beta_inf e quarantine sono nell'header file dal nome "time_function.hpp", in modo da usufruirne come una sorta di libreria da noi definita.

1.5 Personalizzabilità e input

Il programma permette inoltre di scegliere la configurazione iniziale da cui avrà origine l'epidemia, impostando il numero iniziale di individui infetti, suscettibili (e potenzialmente recovered) e le dimensioni della griglia (si consiglia 30X100). E' inoltre necessario indicare se si vuole far uso della quarantena o meno: questa informazione (insieme alle dimensioni iniziali della griglia) viene specificata nella prima riga del file di input. L'utente può predisporre, nella cartella "default_files", un file di tipo .dat o .txt in cui scrivere le coordinate delle celle e il loro stato iniziale, in tre colonne nel formato come da esempio:



```
30 100 Y
1 1 I
1 2 S
1 3 S
1 4 S
1 5 S
1 6 S
1 7 S
1 8 S
1 9 S
1 10 S
1 11 I
1 59 S
```

Il programma chiederà, all'avvio, di inserire il nome del file di tipo .txt o .dat (con il relativo path) da cui leggere le informazioni iniziali. Nei casi in cui il file non esista o i dati di input non siano nel formato corretto, il programma termina, segnalando sul terminale opportuni messaggi di errore. Se non viene predisposto alcun file, è possibile premere il tasto "Enter" e il programma verrà avviato con un file di default contenuto nella cartella "necessary_files" (essa contiene files essenziali per il programma e per i test, pertanto è da non modificare!) e dal nome "input.txt".

2 Principali problemi incontrati e relative soluzioni

In questa sezione elencheremo i principali problemi incontrati durante lo sviluppo del programma e i modi ideati per risolverli.

2.1 Sparizione delle celle

Durante lo sviluppo del movimento delle celle ci siamo accorti che il numero di celle iniziali e finali (ad epidemia conclusa) non coincidevano: alcune celle andavano perse nel ciclo iterativo. Ciò era dovuto al fatto che più celle avessero la possibilità di muoversi verso una medesima destinazione, ovvero una singola cella potesse spostarsi più volte in un singolo ciclo.

Per risolvere questo problema abbiamo implementato due nuovi stati (nominati "MoveI" e "MoveS", rispettivamente per celle in moto Infette e Suscettibili) nella enum class che rappresenta una cella. Il programma dunque muove singolarmente ogni cella e la pone in uno di questi due stati. Al termine di ogni ciclo ogni cella viene convertita allo stato aggiornato.

2.2 Overloading operator() const e non const

Durante lo sviluppo della classe per la griglia, abbiamo implementato due operatori (overloading dell' "operator()") per impostare ed accedere agli stati delle celle: uno const, per leggere gli stati delle celle senza modificare la griglia ed impedire che le celle del bordo venissero alterate, e uno non const, per modificare gli stati della griglia. Ci siamo resi conto, tuttavia, che inizialmente il programma richiama questi due operatori nel modo sbagliato: ad esempio, nel confrontare lo stato di una cella (==), il programma non utilizzava la versione const e dunque restituiva un errore. Per risolvere questo problema, abbiamo fatto in modo che la griglia sia di tipo const nel momento in cui si vuole che

venga chiamato l'operator() const.

Ciò è stato implementato, ad esempio, nelle due funzioni responsabili del movimento e dell'evoluzione infettiva della griglia, tramite l'uso di due griglie: una di tipo const e una di tipo non const.

3 Istruzioni su come compilare, testare, eseguire

Per la compilazione è necessario l'utilizzo di CMake: il programma viene compilato mediante i flag -Wall -Wextra e l'opzione di link -fsanitize=address. Nel dettaglio, si inseriscano da terminale i seguenti comandi (nella directory del programma):

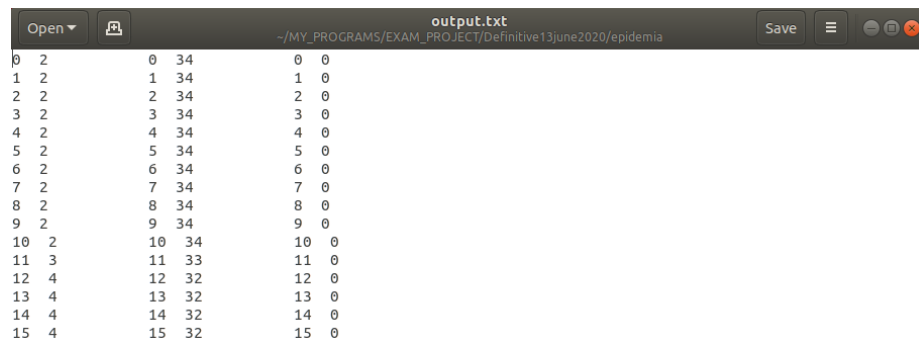
1. cmake .
2. make
3. ./project_epidemic

Per verificare correttezza e funzionalità del programma sono stati predisposti opportuni test (per approfondimenti si veda la sezione 6).

Per effettuare contemporaneamente tutti i test si digiti da terminale il comando: ./run_global_tests

4 Descrizione del formato di output

Alla conclusione dell'epidemia il programma fornisce un file di output (con nome di default "output.txt") contenente tutte le coordinate e gli stati delle celle durante l'epidemia: una sorta di accumulo di bollettini statistico-epidemiologici. Il formato di tale file è visibile nel seguente esempio:



0	2	0	34	0	0
1	2	1	34	1	0
2	2	2	34	2	0
3	2	3	34	3	0
4	2	4	34	4	0
5	2	5	34	5	0
6	2	6	34	6	0
7	2	7	34	7	0
8	2	8	34	8	0
9	2	9	34	9	0
10	2	10	34	10	0
11	3	11	33	11	0
12	4	12	32	12	0
13	4	13	32	13	0
14	4	14	32	14	0
15	4	15	32	15	0

Il primo valore di ciascuna coppia nelle tre colonne rappresenta il tempo (in unità arbitrarie, per esempio giorni), mentre il secondo valore è, rispettivamente, il numero degli infetti, dei suscettibili, dei recovered corrispondenti a tale istante (ciclo iterativo i-esimo).

Tali risultati si possono poi rappresentare in un grafico o scatter plot, per osservare in questo modo l'andamento dell'epidemia dal suo inizio alla sua conclusione (si veda il paragrafo 5 per approfondire tali risultati).

5 Interpretazione dei risultati ottenuti

5.1 Stampa a schermo durante esecuzione

Mostriamo qui inizialmente l'output grafico del programma in esecuzione, ossia la stampa a schermo (su terminale) della griglia abitata dalla popolazione durante l'epidemia:



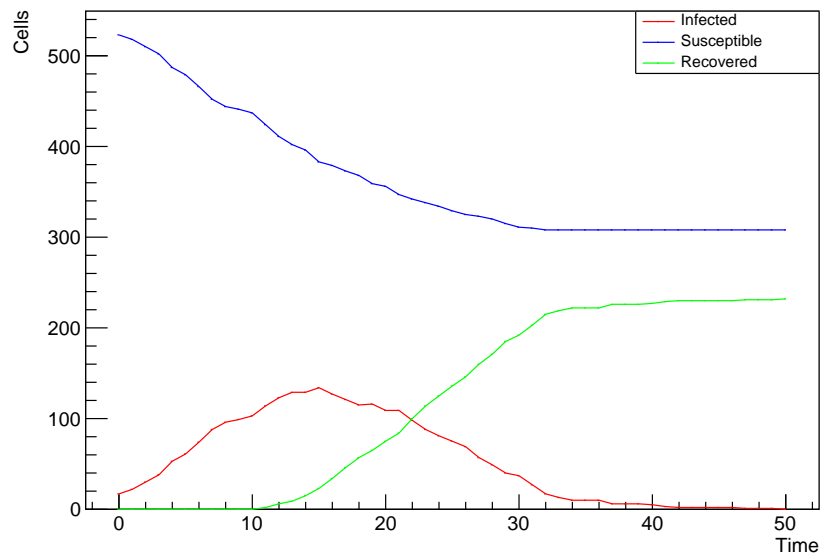
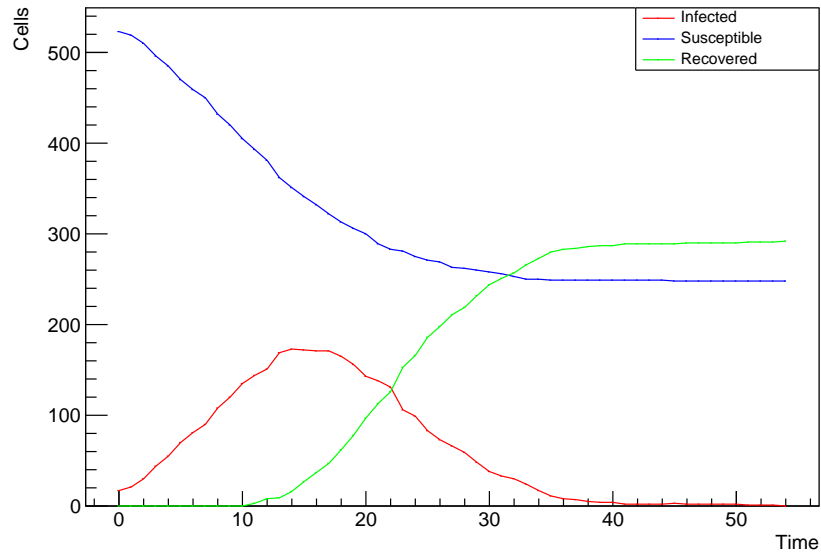
Esempio di stampa su terminale dalla piattaforma Ubuntu 18.04.

Ad ogni iterazione il programma stampa a schermo una rappresentazione grafica di questo tipo: gli individui di colore giallo sono i suscettibili, quelli rossi gli infetti e quelli verdi i recovered.

5.2 Risultati a fine epidemia

Il programma simula un'evoluzione epidemiologica sfruttando delle basi probabilistiche ed al termine dell'epidemia produce un file di output che mostra l'andamento complessivo di tale epidemia. I valori presenti in questo file possono essere graficati con opportuni software per osservare l'andamento globale del contagio dal suo inizio alla sua conclusione.

Mostriamo qui alcune epidemie rappresentate mediante il programma di analisi dati Root:



Questi due grafici sono stati ottenuti a partire da una popolazione di 540 individui: sull'asse delle ascisse è rappresentato il tempo: variabile discreta che indica il numero dell'iterazione del programma. Sull'asse delle ordinate invece viene rappresentato il numero di celle in ogni stato (si veda la legenda in alto a destra in ogni immagine). Il grafico sarebbe quindi in realtà formato da punti, che sono stati uniti con delle linee per mostrare chiaramente gli andamenti. In particolare il primo grafico è stato ottenuto senza l'opzione quarantena, al contrario del secondo. Si nota che, con la quarantena, il grafico del numero di infetti totali risulta avere un massimo inferiore e risulta essere meno piccato

rispetto a quello del primo grafico (senza quarantena). Si nota anche che il grafico dei suscettibili, con la quarantena, risulta avere pendenza inferiore di quello senza quarantena. Inoltre, il numero finale di suscettibili è nettamente superiore nel secondo grafico (o, equivalentemente, il numero finale di recovered è sensibilmente maggiore nel primo grafico): questo significa che, grazie alle misure di quarantena, sono avvenuti meno contagi globalmente. Ciò mostra gli effettivi benefici della quarantena rispetto alla dannosità dell'epidemia sulla popolazione.

6 Strategia di test

Per testare il corretto funzionamento del programma si è deciso di effettuare degli Unit Test mediante Doctest. In particolare sono state testate tutte le funzioni del programma che siano controllabili e il cui esito sia prevedibile: le succitate "evolve" e "step", la funzione per il conteggio degli infetti vicini ad una cella ("near_infect"), i metodi della classe "Board" (in particolare il costruttore e i diversi tipi di conteggio delle celle), le funzioni che regolano le probabilità di contagio, di ospedalizzazione e la quarantena ("time_function"). I test sono stati eseguiti seguendo una "scala gerarchica": ossia testando innanzitutto le parti del programma più basilari (principalmente la classe "Board"), per poi testare le altre parti del programma che lavorano con queste (come "evolve" e "step"). Si è inoltre testato l'input in modo opportuno.

Nelle sezioni successive vengono riportati i principali test eseguiti.

6.1 Test della Board

Per testare la classe base del programma è stato verificato che questa venga creata nel modo corretto. Date, dunque, una base B ed un'altezza H , è stato testato che il numero di celle totali (compresi i bordi della Board) fosse pari al prodotto $(B+2)*(H+2)$ e diverso da altri valori.

Si è poi testato che le celle della Board vengano correttamente allocate: costruendo diverse Board con un diverso numero di celle Suscettibili, Infette e Recovered è stato contato il loro numero mediante dei template appositamente realizzati.

Infine si è testato che le celle negli stati "MoveI" e "MoveS" vengano resettate correttamente ai corrispettivi stati "I" ed "S" dalla funzione "refresh_status" (si veda la sezione 2.1).

6.2 Test dell'evoluzione epidemica

Si è testata la classe Board al lavoro con le funzioni "evolve" e "step" (si veda sezione 1.3) per verificare in particolare che il problema della sparizione delle celle (si veda sezione 2.1) fosse stato risolto.

Sono state testate innanzitutto le due funzioni singolarmente. Ciò è stato fatto creando delle Board, popolandole e applicandovi le due funzioni ciclicamente una alla volta, verificando che il numero delle celle si conservasse. Poi le due funzioni sono state testate in coppia, verificando ancora, ad epidemia conclusa (con 0 Infetti), che il numero di celle si fosse conservato.