Simulazione diffusione epidemia secondo modello SIR

Anastasi Edoardo Lanzi Samuele Merola Samuele

A.A. 2019-2020

https://github.com/samuelelanzi/SIRmodel

Sommario

Il nostro programma si sviluppa in due modalità:

Modalità 1 Permette di ottenere l'andamento di persone suscettibili (S), infetti (I) e rimossi (R) in un'epidemia in funzione del tempo. Utilizza come input i valori di β e γ (valori caratteristici di ogni epidemia) e presenta in output i valori di S, I e R per ogni giorno (vengono simulati 60 giorni, partendo da una situazione di 100 infetti su una popolazione di 1000 unità); i valori in output vengono presentati sia in tabella che in un grafico.

Modalità 2 Permette di visualizzare in una griglia grafica lo stato di ciascuna unità durante il corso di una epidemia, allo stesso tempo viene stampato il grafico delle curve di suscettibili, infetti e guariti in funzione del tempo nell'epidemia simulata. In input si assegnano le celle da ritenere infette all'origine con click del mouse sulle celle. In output, la situazione dell'epidemia in aggiornamento per ogni giorno fino al termine del contagio.

1 Scelte progettuali e implementative

1.1 Modalità 1

Il modello SIR è un modello matematico che descrive l'evoluzione di una epidemia. L'andamento di S, I, R è descritto dalle seguenti formule:

$$S_i = S_{i-1} - \beta I_{i-1} S_{i-1} \tag{1}$$

$$I_i = I_{i-1} + \beta I_{i-1} S_{i-1} - \gamma I_{i-1} \tag{2}$$

$$R_i = R_{i-1} + \gamma I_{i-1} \tag{3}$$

Come possiamo notare sono presenti due costanti caratteristiche dell'epidemia: β indica la densità di probabilità dei contagi

$$\beta = \frac{2 \cdot \beta_0}{N-1} \tag{4}$$

dove con β_0 indichiamo la probabilità che ha un individuo di essere contagiato incontrandone un altro e con N il numero di individui. γ indica la probabilità di guarigione,

$$\gamma = \frac{1}{T} \tag{5}$$

dove T rappresenta il periodo di guarigione espresso in giorni.

Il nostro programma utilizza queste formule per calcolare i valori di S, I e R per ogni giorno i-esimo (si veda ***sir.cpp***).

La durata della simulazione (60 giorni) e il numero totale della popolazione (1000 unità) sono già definiti al compile time. Anche i parametri β e γ vengono definiti dal programmatore, di default abbiamo assegnato $\beta = 0.0035$ e $\gamma = 0.12$ [1] (si veda il ***main.cpp***) Con questi valori il programma riesce a calcolare i valori di S, I e R di ciascun giorno (si veda ***sir.cpp***) e stamparli come standard output (si veda ***print.cpp***), inoltre in ***main.cpp*** viene costruita una finestra con l'ausilio della libreria SFML, vengono creati tre vettori che contengono le coordinate dei valori di S, I e R in funzione dei giorni trascorsi e i punti vengono stampati in un piano cartesiano su questa finestra.

[1] per questi valori abbiamo fatto rifermento a quelli citati dai professori Lamberti, Cifra e Marone del Dipartimento di Matematica dell'Università Sapienza di Roma (https://www1.mat.uniroma1.it/people/pensionati/lamberti/lauree/sir.pdf)

1.2 Modalità 2

Questa modalità prevede di osservare in via grafica l'andamento del contagio in una griglia bidimensionale composta da 256 celle per lato. Ogni cella rappresenta un individuo e può essere di stato S, I o R. Gli individui infetti hanno una certa probabilità di contagiare le celle vicine di stato suscettibile e rimangono infetti per un certo lasso di tempo distribuito uniformemente tra 14 e 20 giorni (si faccia riferimento a ***person.hpp*** e ***random.cpp***). Definito l'oggetto `Person` e inizializzato come `Susceptible`, in ***board.hpp*** viene creata la griglia `Board` come vettore di tipo `Person`. A questo punto il ***main.cpp*** può, con l'aiuto di SFML, riempire una finestra con una griglia piena di celle. Visivamente (si veda ***display.hpp***) le `Person` di stato `Susceptible` vengono colorate di rosso, le `Person` di stato `Infectious` vengono colorate di verde e quelle di stato `Recovered` di blu.

All'avvio del programma l'utente con il mouse dovrà premere su ogni cella che vuole considerare infetta allo stato iniziale e quindi si colorerà di verde. Raggiunto lo stato iniziale desiderato, l'utente darà inizio alla simulazione dei contagi premendo un qualsiasi tasto sulla tastiera. Il ***main.cpp*** stamperà su terminale il conteggio corrente dello stato delle celle (calcolato in ***count.cpp***) aggiornandolo ad ogni ciclo e i valori di β e γ calcolati a partire dalle formule inverse delle equazioni (1) e (3) verrà inoltre stampato il valore R_0 [2]; con questi valori un'altra finestra SFML terrà traccia dell'andamento di S(t), I(t) e R(t) e li graficherà simultaneamente (si vedano le righe 43-145 di ***main.cpp***) . Anche la `Board` (e di conseguenza la schermata grafica) si aggiornerà tenendo conto dei contagi. La situazione dei contagi e l'evoluzione delle celle è definita in ***evolve.cpp***.

2 Istruzioni di compilazione e esecuzione

Per la corretta compilazione del progetto è necessario l'utilizzo di g++ (compilatore del linguaggio C++), la libreria grafica SFML e CMake. Se non si possiedono, è necessario installarli:

```
sudo apt-get install g++ libsfml-dev cmake
```

Dopo aver installato CMake, si può passare alla compilazione:

```
cd SIRmodel
cd option1 (cd option2)
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Debug ..
cmake --build . --target all
make VERBOSE=1
```

Ora il progetto è compilato, si può procedere con i test tramite il comando

```
ctest
```

oppure con l'esecuzione del programma tramite il comando

```
./sir-sfml
```

[2] Maneggiando l'equazione (2) possiamo ricavare che:

$$\Delta I = \beta\big(S_i - \frac{\gamma}{\beta}\big)I_i$$

$$\Rightarrow \Delta I > 0 \Leftrightarrow S_0 > \frac{\gamma}{\beta}$$

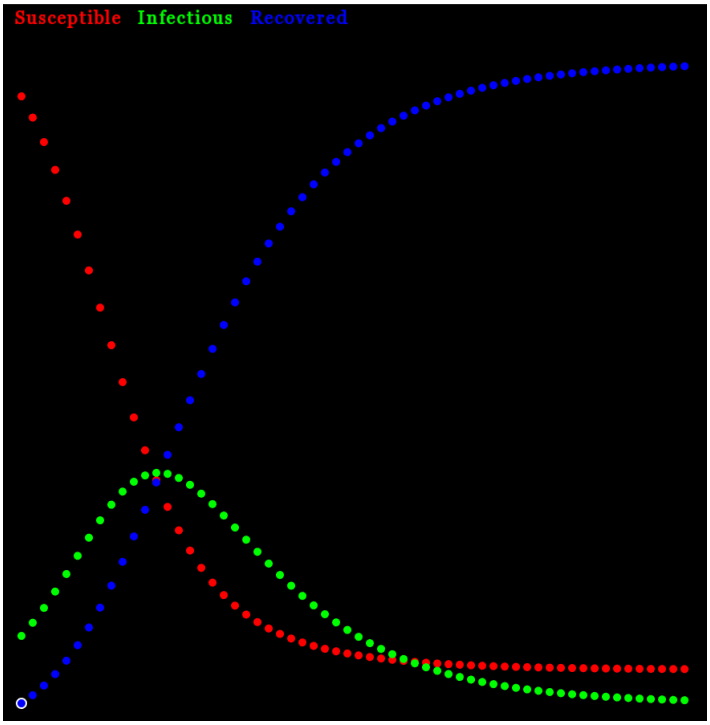
il che significa che l'epidemia si propagherà. Al contrario se $S_0 < \frac{\gamma}{\beta}$ l'epidemia si estinguerà. Il rapporto $\frac{\beta}{\gamma}$ assume dunque un valore di soglia per l'epidemia stessa. Scriviamo (tenendo presente che $S_0 = N$):

$$R_0 = \frac{\beta}{\gamma}N$$

3 Risultati output e interpretazione

3.1 Modalità 1

Eseguendo la prima modalità del programma, verranno mostrati in output del terminale tramite uno `std::cout` i valori di S, I e R per ogni giorno di simulazione [Figura 1 (a)]; inoltre, grazie alla libreria SFML, verrà aperto un grafico che rappresenta tali valori in funzione dei giorni trascorsi. Si vedranno quindi dei punti rossi che rappresentano il numero di persone suscettibili per giorno, dei punti blu che rappresentano i non suscettibili per giorno e dei punti verdi che rappresentano quanti sono infetti in quel giorno [Figura 1 (b)]. Si può notare che il risultato è coerente con il modello SIR atteso (si veda Scelte progettuali e implementative).



(a) *output tramite finestra grafica SFML*

```
S, I, R
900, 100, 0
868.5, 119.5, 12
832.175, 141.485, 26.34
790.966, 165.716, 43.3182
745.089, 191.706, 63.2041
695.096, 218.695, 86.2089
641.891, 245.657, 112.452
586.701, 271.368, 141.931
530.977, 294.528, 174.495
476.242, 313.92, 209.839
423.916, 328.575, 247.509
375.165, 337.897, 286.938
330.797, 341.718, 327.486
291.233, 340.275, 368.492
256.548, 334.127, 409.325
226.546, 324.034, 449.42
200.853, 310.843, 488.304
179.001, 295.393, 525.605
160.495, 278.453, 561.052
144.853, 260.68, 594.467
131.637, 242.615, 625.748
120.459, 224.679, 654.862
110.987, 207.19, 681.823
102.938, 190.375, 706.686
96.0793, 174.389, 729.531
90.215, 159.327, 750.458
85.1842, 145.239, 769.577
```

(b) *standard output da terminale*

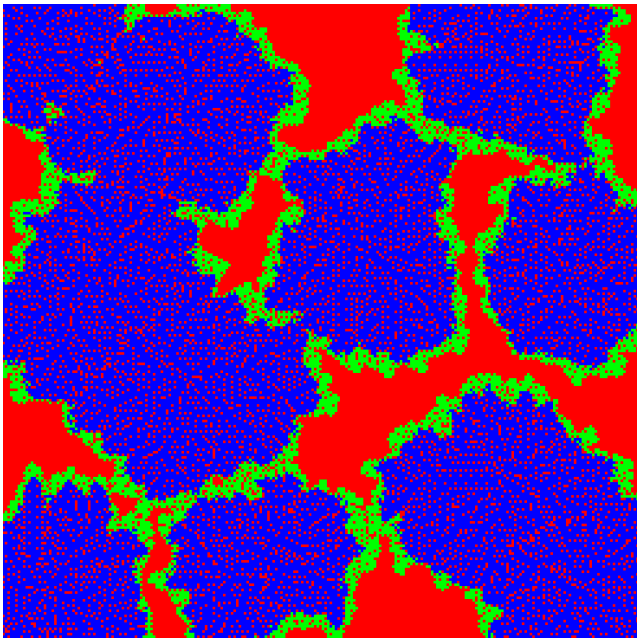
Figura 1

3.2 Modalità 2

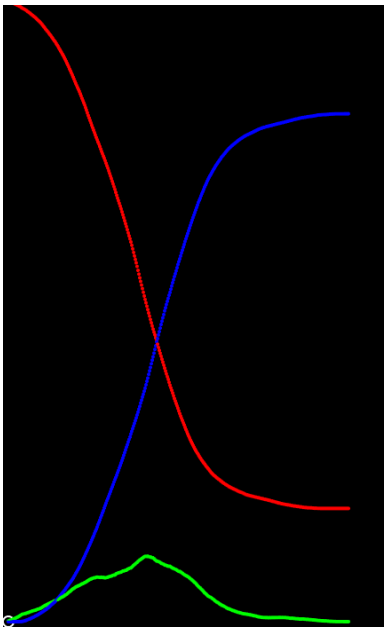
Eseguendo la seconda modalità del programma, dopo aver scelto le celle infette allo stato iniziale (come descritto in Scelte progettuali e implementative), si darà il via alla simulazione premendo un tasto della tastiera. La griglia con le celle [Figura 2(a)] si aggiorna ogni 30ms che rappresenta un giorno, le persone infette rimarranno tali per un numero di giorni distribuito uniformemente tra 14 e 20. Ad ogni ciclo la griglia mostrerà lo stato i-esimo di ciascuna cella facendone variare il colore. Contestualmente, nell'altra finestra [Figura 2 (b)] vengono graficati i valori di S(t) I(t) e R(t) conteggiando i valori reali dell'attuale simulazione. Al termine della simulazione si potranno osservare quante celle sono state colpite dall'epidemia e quante sono ancora suscettibili, mentre nel grafico si potrà apprezzare l'andamento in funzione dei giorni trascorsi.

Il programmatore, prima della compilazione, stabilisce un valore di contagiosità `constexpr int probabilityToInfect` (tra 0 e 100) in ***person.hpp, line 9***. Utilizzando valori differenti si vede come può evolvere l'epidemia: inserendo un'alta probabilità di contagio l'epidemia procederà rapidamente e in pochi giorni gran parte della popolazione risulterà infetta, al contrario inserendo una probabilità molto bassa si vede che l'epidemia rallenta fino a cessare prima di infettare molte celle.

Una verifica ulteriore della correttezza dei risultati della situazione simulata è il confronto dei valori di β, γ e R_0 stampati su terminale con quelli calcolati per via teorica utilizzando le formule del modello SIR (4) e (5). Entro le incertezze statistiche date da fluttuazioni casuali, essi sono compatibili.



(a) *griglia 256x256 rappresentante la situazione di ogni individuo*



(b) *grafico della situazione di contagio secondo il modello attuale*

Figura 2

4 Test e condizioni per la corretta esecuzione

4.1 Modalità 1

Questa modalità per funzionare correttamente necessita di valori logicamente accettabili per i valori di γ e β ovvero entrambe le variabili devono assumere valori strettamente maggiori di 0: per accertarci di ciò abbiamo imposto questi `assert` in ***sir.cpp***. Anche la durata in giorni della simulazione devono essere un numero positivo, pertanto abbiamo impostato anche questo `assert` in ***sir.cpp***. Inoltre per stampare i punti del grafico, il programma legge i valori nei tre `std::vector<Point>` dichiarati in ***main.cpp***, è necessario che questi `std::vector` contengano elementi, dunque abbiamo impostato tre `assert` che impongono la `size()` di ciascuno maggiore di zero.

Abbiamo incluso dei test che possono essere eseguiti per verificare:

- La corretta implementazione della `struct Point`, controllando che alcuni punti che ci aspettiamo essere uguali lo siano effettivamente e in modo analogo punti differenti siano diversi (si veda ***point.test.cpp***).
- Il corretto funzionamento di `create_model()`, controllando che non generi errori passando alcuni parametri iniziali casuali (si veda ***sir.test.cpp***).

4.2 Modalità 2

Questa modalità è estremamente versatile e permette di simulare un grandissimo numero di situazioni con condizioni iniziali differenti. L'unico valore modificabile che ha un dominio prestabilito è `probabilityToInfect` che il programma utilizza nella funzione `Infect()`. Questo valore rappresenta la probabilità di contagiarsi che ha un individuo dopo essere entrato a contatto con individui infetti. Trattandosi di una probabilità deve avere un valore compreso tra 0 e 100. Abbiamo quindi aggiunto questi `static_assert` in ***person.hpp***. Un'ulteriore tutela della correttezza del codice si trova in ***display.cpp***: imponiamo che la `board.size()` coincida con il valore `m_board_side` che deriva dalla dimensione `int dim` imposta alla griglia in ***main.cpp***.

Anche qui abbiamo incluso dei test che possono essere eseguiti per verificare:

- La corretta implementazione della `struct Point`, controllando che alcuni punti che ci aspettiamo essere uguali lo siano effettivamente e in modo analogo punti differenti siano diversi (si veda ***point.test.cpp***).
- Il corretto funzionamento delle coordinate delle celle della `board`. In ***board.test.cpp*** infatti ci accertiamo dell'uguaglianza dello stato di celle uguali. Poi eseguiamo `infectSure()` a una cella e ci accertiamo che il suo stato sia diverso rispetto ad altre di stato `Susceptible` per default. Infine controlliamo che le funzioni `infectSure()`, `update()` e `evolve()` non generino errori.