ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Simulazione di una Pandemia tramite Modello SIR e Automa Cellulare

Progetto a cura di Matteo Donigaglia

Dipartimento di Fisica e Astronomia

9 Settembre 2024

1 Modello SIR

Il modello SIR è uno strumento utile allo studio della diffusione di epidemie, di notizie e di altri fenomeni sociali. Questo modello prevede che la popolazione venga categorizzata in tre gruppi distinti: i Suscettibili (S), gli Infetti (I) e i Rimossi (R).

Il modello è governato da tre equazioni che permettono di ricavare i valori di S, I ed R all'istante t, a partire dagli stessi valori all'istante $t-1$.

$$S(t+1) = S(t) - \beta \cdot \frac{S(t)}{N} I(t) \qquad I(t+1) = I(t) + \beta \cdot \frac{S(t)}{N} I(t) - \gamma \cdot I(t) \qquad R(t+1) = R(t) + \beta \cdot \gamma \cdot I(t)$$

dove S(t), I(t) ed R(t) sono le variabili in funzione del tempo, β è il coefficiente di infettività e γ quello di rimozione.

La classe SirModel implementa queste equazioni grazie al metodo update().
Per rimanere il più possibile in linea con la consegna, il modello è stato implementato così come presentato (la guarigione è stata intesa come una forma di rimozione). In fase di sviluppo era stata però considerata l'introduzione di un terzo parametro η che permettesse ai malati di tornare sani. In seguito questa opzione è stata scartata per il modello SIR e introdotta solo nell'automa cellulare.

Vista la presenza di una componente grafica nel progetto, le percentuali calcolate da SirModel sono stampate sia accanto all'automa cellulare **(Fig.1)** grazie a generateText(), sia come standard output **(Fig.2)** da generateStandardOutput().

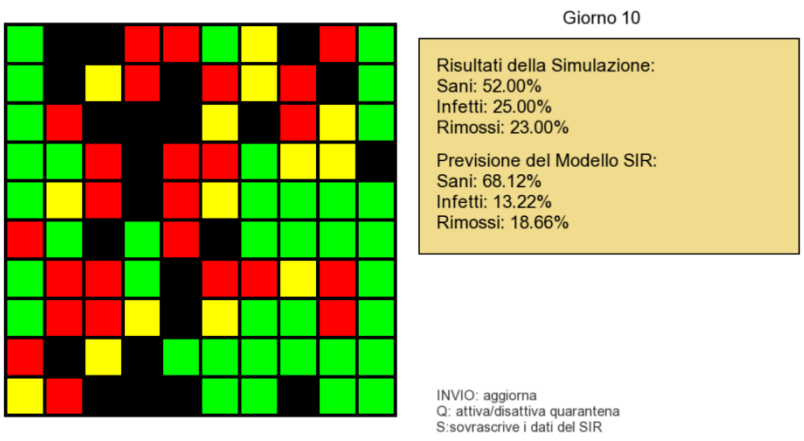
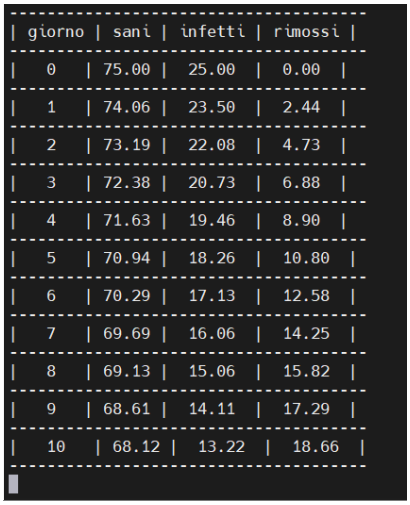


Fig. 1: finestra grafica in cui sono riportati i dati del modello



giorno	sani	infetti	rimossi
0	75.00	25.00	0.00
1	74.06	23.50	2.44
2	73.19	22.08	4.73
3	72.38	20.73	6.88
4	71.63	19.46	8.90
5	70.94	18.26	10.80
6	70.29	17.13	12.58
7	69.69	16.06	14.25
8	69.13	15.06	15.82
9	68.61	14.11	17.29
10	68.12	13.22	18.66

Fig. 2: cronologia delle percentuali previste dal modello stampate a schermo come standard output

2 Automa Cellulare

Come è visibile in **Fig.1**, la componente grafica del programma permette anche di visualizzare una scacchiera che simuli, partendo dagli stessi dati forniti al modello SIR, l'evoluzione di un sistema 2D.
Tale simulazione è stata implementata tramite tre classi: Person, Desease e Population.
Grazie alla libreria <random> è stato possibile implementare metodi di generazione casuale di dati e quindi di gestire in modo stocastico le percentuali rappresentate da β e γ.

2.1 Person

Person è la prima classe implementata e contiene tutti i dati relativi al singolo individuo:

- Lo stato (sano, malato, guarito o morto)
- Lo stato di confinamento (in quarantena o libero)
- I giorni trascorsi dall'ultima infezione

Essa contiene inoltre tutti i metodi responsabili dell'evoluzione dello stato del singolo individuo tramite metodi random come startRandomly(), infect() e checkDestiny(). Il primo determina se al giorno 0 una cella è malata o meno, mentre gli altri due permettono di determinare se una cella si ammala, guarisce o muore al passare dei giorni.
Visto che parte dei suoi metodi sono probabilistici, è stato possibile testare solo i casi limite e verificare che i soggetti evolvessero correttamente.

2.2 Desease

Desease è l'altra classe, insieme a Person, alla base di Population. Si tratta di una classe estremamente semplice, in grado di immagazzinare i dati inseriti relativi alla malattia e di restituirli quando necessari. Essa contiene:

- La probabilità di infezione
- La probabilità di morte
- La probabilità di guarigione
- Il tempo di convalescenza

2.3 Population

Population è la classe responsabile di contenere e gestire la matrice di elementi dell'automa cellulare. Essa contiene:

- La dimensione della matrice N×N
- Il numero di giorni trascorsi dall'inizio della simulazione
- Una matrice di Person
- La matrice in corso di aggiornamento
- La malattia che si intende simulare

I metodi di Population hanno lo scopo principale di ciclare su tutti gli elementi della matrice i corretti metodi di Person e di gestire le interazioni tra una cella e le otto celle vicine.
Population ha inoltre il compito di preparare la stringa da stampare a schermo a fianco della simulazione e di comunicare con il main e con SirModel.
(Nota: si è preferito tenere tutta la parte grafica nel main, ma si poteva integrare in Population o in una classe a parte.)

3 Compilazione ed Esecuzione

Come anche riportato in pandemic.test.cpp per testare il programma è necessario compilare il file di testing con:

g++ -o test.out pandemic.test.cpp Sir_model.cpp population.cpp person.cpp desease.cpp

ed eseguirlo con:

./test.out

Per lanciare il programma invece è necessario compilarlo con:

g++ pandemic.cpp Sir_model.cpp population.cpp person.cpp desease.cpp -lsfml-graphics -lsfml-window -lsfml-system -Wall -Wextra -fsanitize=address

e quindi eseguirlo con:

sudo ./a.out

Come mostrato in **Fig.3**, il programma richiederà all'utente di inserire le specifiche della simulazione grazie alle quali inizializzerà la configurazione iniziale, restituirà quindi le percentuali di partenza del modello SIR e aprirà una finestra dove proietterà l'automa cellulare.

L'utente sarà quindi libero di far evolvere la simulazione a suo piacimento, mentre i dadi restituiti dal modello SIR verranno memorizzati tramite standard output.

```
mattedoni@NB1:~/Desktop_Progetto_Programmazione/progettoSIR_Matteo_Donigaglia$ g++ -o test.out pandemic.test.cpp Sir_model.cpp population.cpp person.cpp des
ease.cpp
mattedoni@NB1:~/Desktop_Progetto_Programmazione/progettoSIR_Matteo_Donigaglia$ ./test.out
[doctest] doctest version is "2.4.6"
[doctest] run with "--help" for options
===============================================================================
[doctest] test cases:  8 |  8 passed | 0 failed | 0 skipped
[doctest] assertions: 45 | 45 passed | 0 failed |
[doctest] Status: SUCCESS!
mattedoni@NB1:~/Desktop_Progetto_Programmazione/progettoSIR_Matteo_Donigaglia$ g++ pandemic.cpp Sir_model.cpp population.cpp person.cpp desease.cpp -lsfml-g
raphics -lsfml-window -lsfml-system -Wall -Wextra -fsanitize=address
mattedoni@NB1:~/Desktop_Progetto_Programmazione/progettoSIR_Matteo_Donigaglia$ sudo ./a.out
inserisci la dimensione della popolazione (il numero delle persone sarà n^2)
10
inserisci la percentuale di popolazione già infetta (es. 10)
10
inserisci il tasso di infezione, cioe' la probabilità che una persona infetti un vicino (es. 30)
5
inserisci il tasso di mortalità, cioe' la probabilità che una persona infetta muoia (es. 20)
3
inserisci il tasso di guarigione, cioe'la probabilità che una persona infetta guarisca (es. 25)
5
inserisci il periodo di immunita' post-guarigione, cioè il numero di giorni di immunita' (es. 4)
7
Setting vertical sync not supported
-------------------------------------
| giorno | sani | infetti | rimossi |
-------------------------------------
|   0    | 93.00 |  7.00  |  0.00  |
-------------------------------------
```

Fig. 3: compilazione ed esecuzione del programma

4 Conclusioni

A causa della presenza di metodi random è difficile valutare se l'evoluzione dell'automa cellulare sia consistente o meno con i dati inseriti. Si può però notare che all'aumentare della dimensione della scacchiera la percentuale di malati inizialmente generati diventa sempre più consistente con la percentuale inserita, inoltre eseguendo simulazioni dei casi limite (0% e 100%) si può notare una corretta evoluzione della scacchiera.

I dati simulati dall'automa cellulare non trovano grande riscontro con quelli previsti dal modello SIR, e le cause possono essere molteplici:

- Nell'automa cellulare i guariti tornano sani
- Il modello SIR non prevede giorni di immunità
- La simulazione implementa la pandemia in 2D

Introducendo il parametro η di cui si parlava in precedenza ad esempio, si può modificare il modello SIR in modo che assomigli maggiormente alla simulazione.

Altrimenti si potrebbe fare in modo che i guariti fossero considerati dalla simulazione come rimossi e si potrebbe inserire un numero di giorni di immunità talmente grande da renderli rimossi per tutta la simulazione.

L'ultimo punto probabilmente è il più ostico da affrontare, infatti se si pensa ad una malattia con il 100% di infettività e la si applica ad una popolazione di 99 sani e un malato, il modello SIR restituirà 2 malati il giorno successivo, mentre la simulazione ne avrà mediamente 9. Per risolvere questa discrepanza (che sembra aumentare con l'infettività) si potrebbero manipolare i dati in entrata ad uno dei due metodi in modo che ci sia una migliore corrispondenza.

Visto che non si può pensare di eccedere il 100% di infettività nel modello SIR potrebbe essere una soluzione usare:

$$\beta_{SIR} = \beta_{inserito} \qquad \beta_{SIMULAZIONE} = \frac{\beta_{inserito}}{8}$$