Programmazione per la fisica
Relazione progetto finale

Bartoli Arnaldo
Raumer Giovanni
Severino Samo

05/07/2021

Link Repository GitHub: https://github.com/GiovanniRaumer/progetto.git

Indice

1 Introduzione

Il progetto qui presentato consiste nell'implementazione del modello epidemiologico SIR (Suscettibili, Infetti e Rimossi) in linguaggio di programmazione c++. Una prima parte del progetto riguarda il modello vero e proprio, con le sue equazioni caratteristiche: una volta inseriti i dati di una possibile epidemia quali ad esempio suscettibili e infetti iniziali il programma è in grado di fornire una simulazione dell'epidemia e della sua evoluzione in termini numerici; è possibile simulare un lockdown generalizzato per studiarne gli effetti sull'evoluzione di un epidemia. La seconda parte fornisce invece una simulazione grafica dell'evento epidemiologico tramite automa cellulare: una griglia di dimensioni definite dall'utente simula i contagi dovuti al contatto tra suscettibili e infetti ed evidenzia la velocità di trasmissione dell'epidemia. Per questa seconda parte è inoltre possibile simulare una campagna vaccinale e vederne gli effetti in termini di protezione della popolazione e rallentamento dei contagi.

2 Implementazione del modello SIR

L'implementazione del modello consiste in tre file:

- un header file chiamato "sir_l.hpp"
- un file chiamato "sir_l.cpp"
- un file chiamato "main_sir_l.cpp" che include il main e le funzioni utili per la stampa dei risultati su terminale

Nell'header file viene definita una struct chiamata "State": l'epidemia nella sua evoluzione viene gestita per giorni, ogni oggetto di classe State quindi memorizza il numero di suscettibili, infetti, rimossi, i parametri gamma e beta e rappresenta la situazione giornaliera dell'epidemia. A partire da un oggetto State infatti si costruisce un oggetto della medesima classe che rappresenta il giorno successivo. Questo compito viene svolto dalla classe "Epidemic": a partire da uno State iniziale definito dall'utente, tramite il metodo pubblico "evolve" calcola gli stati successivi secondo le equazioni definite dal modello. Il metodo evolve dichiarato in sir_l.hpp è definito in sir_l.cpp.
Un'istanza Epidemic sulla quale è stato chiamato il metodo evolve è quindi un vettore di oggetti di tipo State i cui dati verranno stampati su terminale dalle funzioni definite nel file di tipo .cpp.
È infatti nel file "main_sir_l.cpp" che vengono definite due funzioni che permettono di visualizzare su terminale i risultati della simulazione. il primo output grafico è una semplice tabella di 4 colonne, una per ogni classe di popolazione (S,I,R; i dati stampati sono ottenuti "scorrendo" gli elementi del vettore restituito da evolve e i loro membri) e con i corrispondenti valori giornalieri; il secondo output è un grafico analogo alla tabella, ruotato di 90 gradi per non limitare l'estensione dell'asse del tempo verticale. La limitazione si presenta però nel numero di individui per classe. L'output è un grafico scalato che riesce a rappresentare la cardinalità delle classi anche per numeri arbitrariamente grandi; tuttavia a causa di alcune approssimazioni, quando gli individui di una classe sono molto inferiori a quelli di un'altra classe l'andamento può subire delle piccole oscillazioni.

2.1 Compilazione ed esecuzione del programma

La compilazione viene effettuata con il compilatore g++ con le seguenti opzioni per rilevare problemi nel codice:

```
g++ -Wall -Wextra -g -fsanitize=address
```

Ad essere compilati saranno i file "sir_l.cpp" e "main_sir_l.cpp":

```
g++ -Wall -Wextra -g -fsanitize=address sir_l.cpp main_sir_l.cpp
```

Se la compilazione va a buon fine, quindi senza alcun tipo di errore, warning o memory leak si può procedere con l'esecuzione del programma. All'utente si richiede di inserire 5 parametri come segue:

```
./a.out 100 4 37 0.7 0.4
```

Il primo e il secondo parametro sono rispettivamente il numero di suscettibili e infetti iniziali, il terzo corrisponde alla durata della simulazione in unità di giorni mentre il quarto e il quinto sono il tasso di contagio beta e il tasso di rimozione gamma. È chiaro che questi parametri debbano rispettare alcuni vincoli che il programma segnala in caso di errore. Tutti i controlli sugli input sono gestiti nel file main_sir_l: un try-block racchiude le linee di codice addette al controllo vero e proprio, da cui possono essere sollevate eccezioni con opportuni messaggi raccolte dall'handler catch; i vincoli da rispettare sono adeguatamente segnalati ad input errato, assieme ad un riepilogo delle istruzioni per l'immissione dati.
Il programma, con i valori di esecuzione, fornisce i seguenti output.

day	S	I	R
1	100	4	0
2	97	5	2
3	94	6	4
4	90	7	7
5	86	8	10
6	81	9	14
7	76	10	18
8	71	11	22
9	66	12	26
10	61	13	30
11	56	13	35
12	51	13	40
13	47	12	45
14	43	11	50
15	40	10	54
16	37	9	58
17	35	8	61
18	33	7	64
19	31	6	67
20	30	5	69
21	29	4	71
22	28	3	73
23	27	2	75
24	27	1	76
25	27	0	77

end of epidemic

Figura 1: tabella ottenuta dal modello SIR

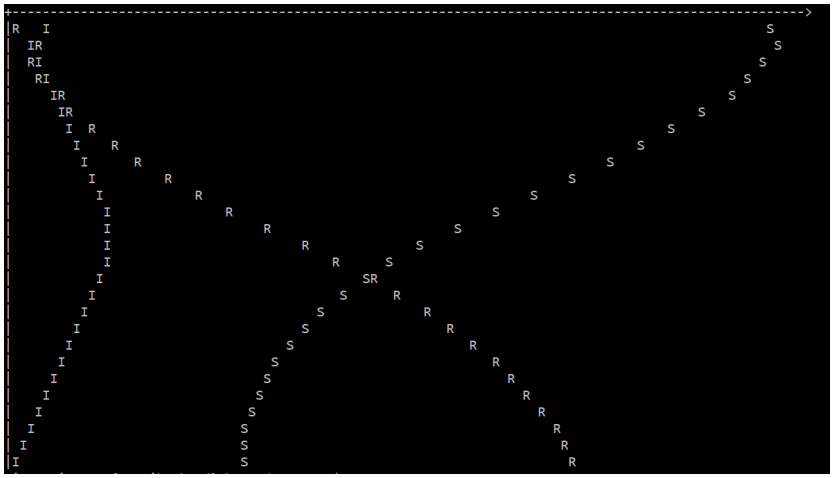


Figura 2: grafico ottenuto dal modello SIR

Si può notare che il numero di iterazioni eseguite dal programma o in altri termini la durata della simulazione richiesta dall'utente è maggiore dell'effettiva durata elaborata dal programma, questo perché i dati inseriti portano alla conclusione dell'epidemia, ossia alla scomparsa di individui infetti prima del raggiungimento dell'i-esimo giorno. In caso contrario, quando cioè la durata richiesta è inferiore all'effettiva durata dell'epidemia, il programma fornisce le iterazione richieste. L'utente ha l'opportunità di utilizzare il modello simulando un lockdown generalizzato della popolazione che ha come effetto quello di decrementare il parametro beta. Per questo tipo di simulazione l'utente deve inserire 2 ulteriori input: un primo numero compreso tra 0 e 1 rappresenta la frazione di infetti sulla popolazione totale che induce il lockdown come misura contenitiva. Una volta raggiunta questa soglia il tasso di contagio beta viene diminuito di una percentuale definita dall'utente tramite il settimo e ultimo parametro, anch'esso un numero compreso tra 0 e 1. Se inseriamo i seguenti parametri

```
./a.out 100 4 37 0.7 0.4 0.1 0.6
```

si ottengono degli output (Figura 3 - Figura 4) che dimostrano come l'epidemia si estingua in un tempo minore. Si ottiene infatti il picco del numero di infetti giorni prima rispetto alla situazione senza misure di contenimento, il picco inoltre è di minor intensità.

day	S	I	R
1	100	4	0
2	97	5	2
3	94	6	4
4	90	7	7
5	86	8	10
6	81	9	14
7	76	10	18
8	71	11	22
9	68	10	26
10	65	9	30
11	63	8	33
12	61	7	36
13	59	6	39
14	58	5	41
15	57	4	43
16	56	3	45
17	55	2	47
18	55	1	48
19	55	0	49

end of epidemic

Figura 3: tabella ottenuta dal modello SIR con lockdown generalizzato

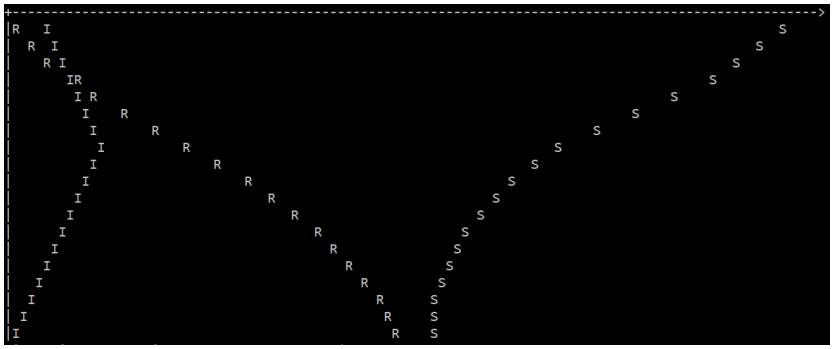


Figura 4: grafico ottenuto dal modello SIR con lockdown generalizzato

2.2 Test

Abbiamo effettuato alcuni test per verificare la correttezza del codice (file "test_sir_l.cpp"). Per prima cosa è stata verificata la costanza della popolazione per tutte le evoluzioni dello stato iniziale. Abbiamo poi controllato che il parametro beta rimanesse costante per le evoluzioni senza lockdown e che variasse in caso di lockdown richiesto dall'utente. Infine è stato verificato che il numero di infetti fosse nullo al termine della pandemia (per un opportuno numero di giorni dato in input).

3 Implementazione del modello tramite automa cellulare

Per questa seconda parte del progetto abbiamo deciso di dividere il codice in 3 file:

- un header file chiamato "sir_v.hpp"
- un file chiamato "sir_v.cpp"
- un file chiamato "main_sir_v.cpp" che include il main e le funzioni utili per la stampa dei risultati su terminale

Oltre alle equazioni del modello abbiamo deciso di implementare anche un'automa cellulare che simulasse l'epidemia e fornisse un output con un più alto impatto grafico. Consiste infatti in una tabella quadrata di lato definito dall'utente suddivisa in celle. Ognuna di queste rappresenta un individuo che può appartenere sempre ad una delle tre categorie del modello SIR e la simulazione del contagio avviene non più tramite equazioni ma secondo delle regole di vicinanza tra infetti e suscettibili che verranno descritte in seguito.
Nell'header file "sir_v.hpp" vengono definite la struct "Human" e la classe "Population", utili all'implementazione della griglia, racchiuse nel namespace "pandemic". Ogni elemento della cella appartiene alla struct "Human" che tramite un enum differenzia i 3 tipi di individui S, I, R. Sono state definite delle variabili di classe quali days_I e v, rispettivamente un oggetto di tipo int e un booleano. Il primo tiene il conteggio dei giorni durante i quali il soggetto rimane infetto e quindi contagioso: questo torna utile nel momento in cui l'individuo guarisce, avvenimento regolato dall'inverso del parametro gamma, ossia la durata media della malattia in esame in un soggetto. La variabile booleana v invece, utilizzata in caso di simulazione con vaccino, sta ad indicare se l'individuo è stato immunizzato o meno, questo renderà più difficile il contagio in funzione dell'efficacia del siero.
La classe "Population", ossia la classe che costituisce il tipo della griglia consiste in un vettore contenente vettori di Human e viene inizializzata in base ai parametri forniti dall'utente. La classe ha diversi metodi definiti nel file "sir_v.cpp", tra cui due metodi di accesso alle celle: "human", uno di tipo const di sola lettura e un secondo che permette l'accesso e la modifica della cella in caso di infezione, rimozione o vaccinazione. Sono poi stati definiti alcuni metodi utili all'evoluzione giornaliera della griglia: infected, recovered, vaccinated e S_not_vax contano rispettivamente gli infetti, i rimossi, i vaccinati e i suscettibili non vaccinati presenti nella "Population" sulla quale vengono chiamati.

3.1 Le funzioni "contacts" e "evolve"

In sir_v.hpp sono state definite due funzioni inline: contacts, che prende come argomenti anche due int, conta i suscettibili attorno ad una particolare cella la cui posizione è indicata dai due argomenti e la funzione evolve per l'evoluzione della griglia.
Per quanto riguarda evolve: ogni volta che viene chiamata su un oggetto di tipo population, per prima cosa ne crea una copia sulla quale si vanno ad effettuare delle modifiche e che andrà a costituire la griglia del giorno successivo. Viene poi effettuato un controllo per verificare se l'utente abbia chiesto o meno di considerare la vaccinazione della popolazione. In caso positivo, viene vaccinata l'1% della popolazione ogni giorno secondo un criterio di distribuzione casuale. Gli individui protetti non sono completamente immunizzati, la probabilità di infezione dipenderà dall'efficacia del vaccino inserita dall'utente.
Il corpo principale della funzione consiste nella dinamica del contagio: i parametri utili sono l'indice di riproduzione di base R0 (beta su gamma) e il periodo medio di infettività, ossia il reciproco del parametro gamma. La funzione scorre tutte le caselle della griglia e ogniqualvolta trova un individuo

infetto procede infettando un numero pari a R0 di individui suscettibili scelti casualmente attorno alla casella considerata. Essendo il numero di riproduzione di base un numero di tipo double, nel rapporto beta su gamma viene trascurata la parte decimale; viene quindi considerato questo scarto come la probabilità che avvenga un contagio in più, e ciò avviene dopo aver estratto con successo un numero casuale tra 0 e 1, in caso contrario il contagio si ferma alla parte intera di R0. Se il numero di suscettibili nell'intorno dell'infetto è minore di R0 questi individui vengono tutti contagiati. In caso di individuo suscettibile vaccinato, il contagio avviene solo dopo aver estratto un numero casuale: se questo è maggiore dell'efficacia del vaccino allora il contagio avviene, altrimenti l'individuo rimane suscettibile.
La variabile di classe d, che rappresenta il periodo di infettività di un infetto, viene incrementata ad ogni iterazione; la guarigione avviene se e solo se la variabile days_I ha raggiunto il valore del periodo medio di infettività. Anche qui consideriamo la parte decimale del periodo medio come probabilità che la contagiosità duri un giorno in più e viene trattata allo stesso modo: dopo aver estratto un numero casuale si decrementa o meno la variabile days_I di un determinato individuo.

3.2 Output grafico

Nel file main_sir_v.cpp, la tabella popolata di individui viene stampata a partire dalla funzione "print": questa prende come argomento un oggetto di tipo Population e un int come contatore per i giorni. La funzione sfrutta il contatore e i metodi della classe Population come sopra descritti per stampare a schermo: giorno (numero), numero di soggetti di tipo S, I, R ed eventuali vaccinati (V), laddove i soggetti sono i singoli elementi Human dell'oggetto Population. Questi ultimi vengono poi stampati, ricorrendo ai metodi human sopracitati, con un simbolo esemplificativo del rispettivo stato (S, I, R, V), così da costituire una tabella come in Figura 5.
Un ciclo for nel main itera la chiamata di print (e di evolve, fondamentale per l'evoluzione della popolazione) fino a fine evoluzione della pandemia o fino al giorno deciso dall'utente, così che a schermo si apprezzino i cambiamenti della popolazione (nonché dell'oggetto Population preso come argomento da evolve e print).

3.3 Compilazione ed esecuzione del programma

La compilazione avviene con gli stessi comandi visti per la prima parte del progetto.
Ad essere compilati saranno i file "sir_v.cpp" e "main_sir_v.cpp":

```
g++ -Wall -Wextra -g -fsanitize=address sir_v.cpp main_sir_v.cpp
```

Per quanto riguarda l'esecuzione del programma, anche qui vengono richiesti come minimo 5 parametri:

```
./a.out 45 20 365 0.6 0.5
```

In questo caso il primo parametro da inserire non corrisponde al numero iniziale di suscettibili ma al lato della griglia quadrata mentre i restanti parametri corrispondono come prima al numero di iniziale di infetti, al numero di iterazioni, al parametro beta e gamma. In figura 5.a è riportato un esempio di output fornito dal programma.

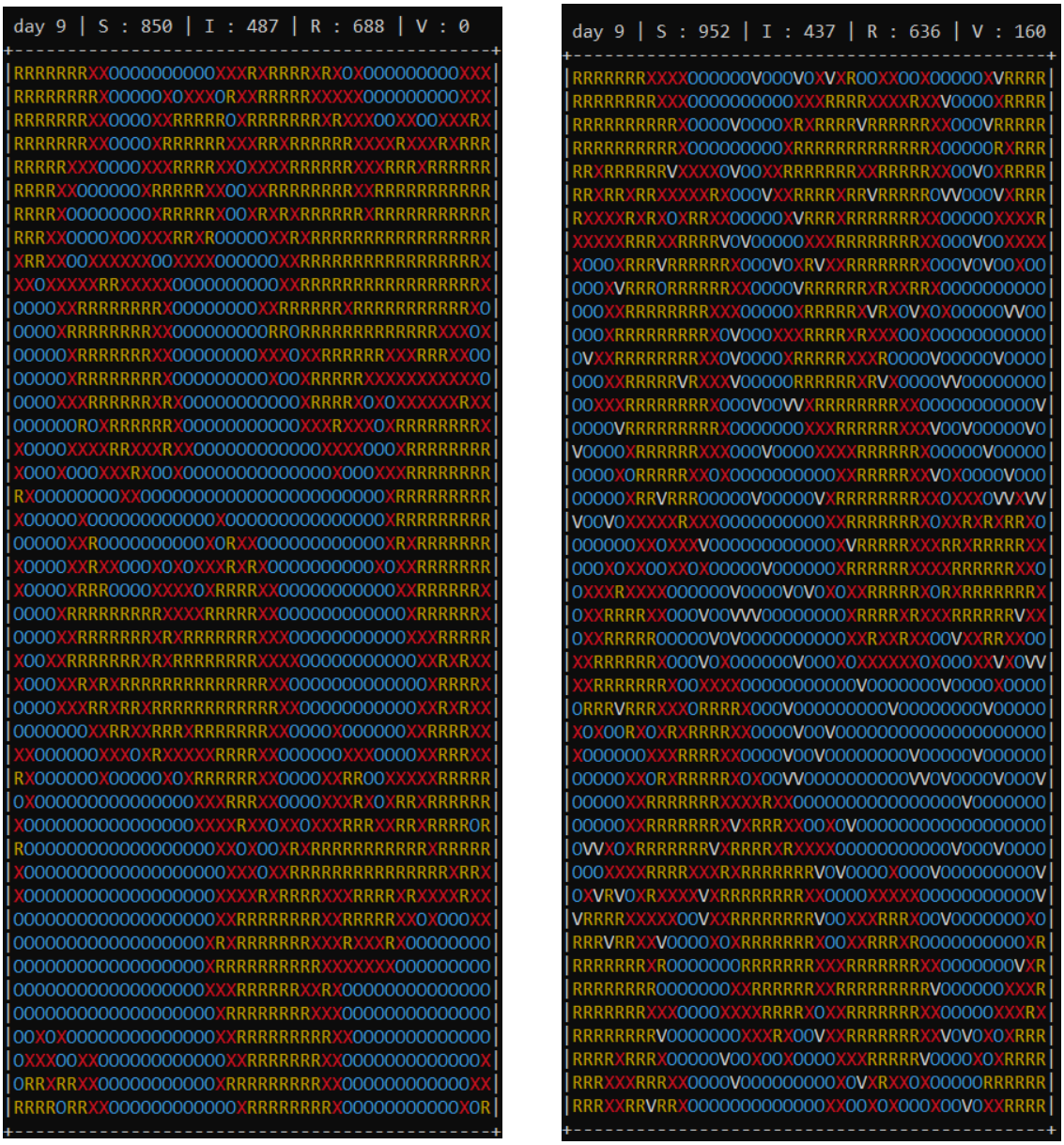
Se si vuole includere la vaccinazione della popolazione il programma necessita di altri tre parametri come segue:

```
./a.out 45 20 365 0.6 0.5 v 2 0.94
```

oltre ai primi 5 parametri necessari, i 3 parametri aggiuntivi indicano le caratteristiche del piano vaccinale: la "v" segnala l'aggiunta degli individui vaccinati e la diversa modalità di contagio, il settimo parametro rappresenta l'iterazione durante la quale ha inizio la vaccinazione degli individui mentre l'ultimo indica l'efficacia del siero. L'output in questo caso presenta delle differenze (Figura 5.b).
Chiaramente anche questi parametri hanno dei vincoli da rispettare. I controlli sugli input sono gestiti nel file main_sir_v, in modo del tutto analogo a quello per l'implementazione del modello SIR con tabella e grafico (una diversa serie di istruzioni e diversi avvisi appariranno a schermo in caso di errata immissione dati).

3.4 Test

Anche per questa parte del programma abbiamo effettuato alcuni test, contenuti nel file "test_sir_v.cpp". Abbiamo verificato che una volta passato da I a R lo stato di un soggetto non potesse mutare, in particolare che non potesse tornare I. Abbiamo poi verificato che a vaccinazione avviata, in presenza di suscettibili ancora non vaccinati, il numero di immunizzati aumentasse da uno stato al successivo e infine, che in presenza di infetti e con determinati beta e gamma, il numero di soggetti di tipo R aumentasse da uno stato al successivo, mentre i suscettibili continuassero a diminuire.



(a) popolazione senza vaccinati

(b) popolazione con vaccinati

Figura 5: due esempi di popolazione in output

4 Conclusioni

Lo scopo di questo progetto è stato quello di creare un programma in grado di dare una descrizione quantitativa dell'evoluzione di un'epidemia. Il modello SIR è il modello più semplice e più approssimato in grado di fornire tale descrizione e per questo motivo non risulta sufficiente a descrivere realisticamente un simile evento. Tuttavia permette di evidenziare l'efficacia di strategie contenitive come il lockdown o il vaccino. Si può notare infatti l'efficacia di tali strategie in termini di individui infettati e velocità di diffusione e arresto dell'epidemia.
L'implementazione del modello qui esposta presenta sicuramente delle possibilità di miglioramento dal punto di vista dell'ottimizzazione del codice come pure dal punto di vita delle capacità del programma. Alcuni miglioramenti potrebbero essere effettuati sulla qualità dell'output grafico e altri sulla dinamica del contagio e del recupero degli individui che qui sono stati affrontati in maniera più semplice e chiara possibile. Esistono infatti numerosi modelli che includono altre classi di individui come il SEIR o o il SEIQR, implementabili a partire dal modello qui presentato.