Implementazione modello SIRD con logiche Equation Based e Agent Based

Collaboratori progetto:

Ludovica Rainero
Sofia Zalambani

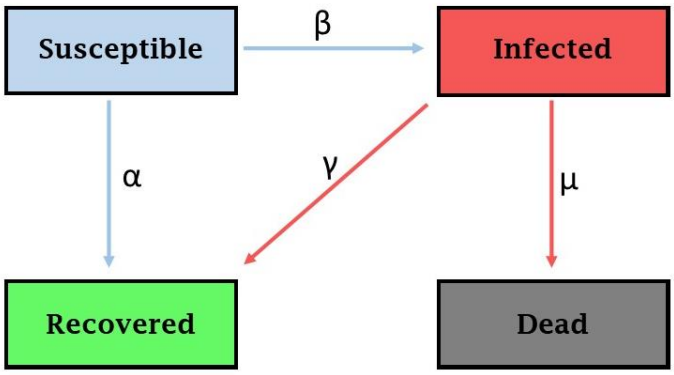
Introduzione

Il nostro progetto prevede la simulazione dell'evoluzione di una pandemia, con la popolazione che può assumere quattro stati differenti: suscettibili, infetti, guariti e morti (indicati nel codice con la traduzione inglese, Susceptible, Infected, Removed, Dead). Se una persona si trova nello stato di suscettibile, può o infettarsi o entrare nello stato di guarito tramite vaccinazione; se si trova nello stato di infetto, può guarire o morire, secondo una certa probabilità; gli stati di guariti e morti sono inerti.

Il passaggio tra stati è regolato da quattro parametri $(\alpha, \beta, \gamma, \mu)$, che rappresentano rispettivamente la rate di vaccinazione, la probabilità di contagio, la probabilità di guarigione e la probabilità di morte.

L'evoluzione è stata sviluppata in due codici distinti, basati su logiche diverse, che chiameremo Equation Based e Agent Based.

La differenza principale è che l'Equation Based ha una logica deterministica, quindi inserendo gli stessi dati iniziali fornirà le medesime condizioni finali; l'Agent Based ha invece logica probabilistica, di conseguenza lo stato finale può variare fornendo le stesse condizioni iniziali. Le variazioni saranno molto piccole, essendo comunque un sistema lineare.



Istruzioni compilazione e esecuzione

Per settare CMake:

```
cmake -S . -B build -DCMAKE_BUILD_TYPE=Debug
```

Per buildare:

```
cmake --build build
```

Vengono prodotti quattro eseguibili: ***equation, agent, equation.test, agent.test***., i primi due per eseguire i programmi e gli ultimi due per il testing.

Logica programma

Equation Based

Il codice Equation Based è basato sulla discretizzazione delle equazioni differenziali del modello SIRD

$$\frac{dS}{dt} = -\frac{\beta SI}{N} - \alpha S_0$$

$$\frac{dI}{dt} = \frac{\beta SI}{N} - \gamma I - \mu I$$

$$\frac{dR}{dt} = \gamma I$$

$$\frac{dD}{dt} = \mu I$$

dove N rappresenta il numero totale di persone e S_0 il numero di suscettibili iniziali. Questo approccio si basa sull'ipotesi di full-mixing, quindi tutta la popolazione ha la stessa probabilità di infettarsi.

Il codice è strutturato nel seguente modo.

La classe People (people.cpp – people.hpp) contiene gli attributi privati S_, I_, R_, D_, che rappresentano la quantità di persone appartenenti ad ogni stato ad un certo istante di tempo. I metodi Set assegnano il valore agli attributi privati effettuando prima un controllo di range; se il controllo non viene soddisfatto, la modifica non avviene e gli attributi mantengono il valore istanziato col costruttore di default.

La classe Parameters (parameters.cpp – parameters.hpp) contiene come attributi privati Alfa_, Beta_, Gamma_, Mu_, che rappresentano i parametri della pandemia. Analogamente a People, i metodi Set controllano il valore fornito dall'utente e, se rientra nel range, lo assegnano al rispettivo attributo privato.

La classe Pandemic (pandemic.cpp – pandemic.hpp) contiene come attributi privati un oggetto Parameters, un vettore di People, che rappresenta lo storico dell'evoluzione della pandemia, e un intero, che rappresenta il tempo della simulazione.

Il metodo simulate() contiene la discretizzazione delle equazioni differenziali.

Nella prima parte (righe 68-125) vengono effettuati i calcoli con delle variabili locali double e poi il valore, troncato, assegnato ad un oggetto di tipo People. Sono inseriti controlli per impedire che il numero di persone scenda sotto lo zero (righe -) o il totale vari (righe -). Se il numero di suscettibili diventa negativo, è stato scelto di porre il numero uguale a zero e assegnare le persone interamente ai guariti.
Il controllo sul totale viene fatto calcolando la differenza tra il numero iniziale di persone e il numero contenuto nell'oggetto locale. Dato che la differenza è dovuta alla troncatura nel passaggio da double a int, questa viene assegnata ai diversi stati in maniera proporzionale al rispettivo scarto, con una distribuzione random normalizzata alla somma degli scarti.
La simulazione si ferma quanto suscettibili e infetti arrivano entrambi a zero.

Agent Based

Nel codice Agent Based viene fatta cadere l'ipotesi di full-mixing e la popolazione è rappresentata come disposta su una mappa (o griglia); per questo motivo la probabilità di infezione di un individuo dipende dalla sua posizione spaziale.

È presente una classe Parameters, analoga a quella dell'Equation Based.

La classe Pandemic (pandemic.cpp – pandemic.hpp) contiene come attributi privati

- quattro interi (S_, I_, R_, D_) che rappresentano il numero di persone appartenenti ad ogni stato in un determinato istante di tempo;
- un intero (Side_), la lunghezza del lato della mappa;
- un vettore di Person (Grid_), dove Person è un enum contenente i possibili stati che una persona può assumere.

I suoi metodi sono Reading_cell() per leggere lo stato della persona, Writing_cell() per modificarlo, infected_neighbours() per contare il numero di infetti intorno alla persona, start() per assegnare il numero iniziale di infetti, e evolve() per far avanzare di un giorno la simulazione della pandemia.

Al metodo start() vengono forniti una referenza alla griglia da inizializzare e il numero iniziale di infetti da inserire. Viene generato un numero casuale e si verifica se la casella con indice corrispondente è infetta; se no, viene cambiato lo stato, se sì, si controlla quella vicina.

All'interno di evolve() viene controllato lo stato di ogni singola casella: se sono guariti o morti, non accade nulla; se sono suscettibili, viene calcolata la probabilità di infezione della persona in base al numero di vicini, dopodiché viene generato un numero random tra 0 e 1 e, in base al valore assunto, la persona può vaccinarsi, infettarsi o rimanere suscettibile; se la persona è infetta, viene nuovamente generato un numero random che determina il cambio di stato (guarito, morto, ancora infetto).

Descrizione Input/Output

Equation Based

Quando si avvia l'eseguibile per l'interazione con l'utente, compare subito un messaggio di testo per chiedere se si desidera inserire i dati da File, da Standard Input oppure creare una generazione random.

All'interno della directory è fornito un File, chiamato data.ini, contenente i dati necessari alla simulazione. L'utente può decidere di modificare il file, prima di avviare l'eseguibile, o da Command Line con il comando *vim data.ini* o da Editor di testo. Se l'utente fornisce dati fuori range (ad esempio, numero iniziale di infetti negativo) verranno assegnati con valori di default. I valori non interi verranno troncati.

Da Standard Input verranno richiesti i numeri iniziali della popolazione (S-I-R-D), i parametri $(\alpha, \beta, \gamma, \mu)$ e il tempo della simulazione. Se viene fornito un dato fuori range, o un numero non intero per le persone, viene usato il valore di default.

Nel caso in cui si scelga la simulazione random, verranno stampati a schermo i numeri generati e la simulazione partirà automaticamente.

In tutti e tre i casi, vengono stampati a schermo il giorno della simulazione e i valori della popolazione corrispondenti.

Se l'utente assegnasse sia agli infetti che ai suscettibili il valore 0, la simulazione non potrebbe partire. Per questo è stato inserito, all'interno del main, un try/catch statement per interrompere il programma. Interrompiamo il programma anche in altre due situazioni:

- Se si cerca di assegnare sia a Gamma_ che a Mu_ il valore 0.5; in quel caso i suscettibili cambierebbero sicuramente stato all'iterazione successiva, diversamente da quanto accade in una reale pandemia;
- Se (Gamma_+Mu_>Beta_); se così fosse, il numero di persone che escono dallo stato di infezione sarebbe maggiore di quello che vi entra e la pandemia non inizierebbe.

Agent Based

Una volta avviatosi il programma, vengono chiesti in ingresso i parametri della pandemia (tranne la vaccinazione, che è posta costante nel programma), il lato della mappa, il numero iniziale di infetti e il tempo della simulazione.
La mappa iniziale contiene solo suscettibili e infetti, come l'inizio di una pandemia.
Nel main_pandemic.cpp avvengono controlli try/catch analoghi all'Equation Based.
In output viene disegnata una finestra grafica contenente la mappa, dove ogni persona corrisponde ad una cella, e il colore ne definisce lo stato (suscettibili = blu, infetti = rosso, guariti = verde, morti = nero).

Strategia testing

Per entrambe le logiche sono stati testati Setter, Getter e costruttori di default, verificando direttamente il valore assegnato. Ugualmente per i metodi di lettura e scrittura della cella nell'Agent Based.

Nella logica Equation Based è stata generata una simulazione (metodo simulate()) e si verifica che il numero di persone sia maggiore o uguale a zero ad ogni iterazione.

In Agent Based viene verificato:

- il metodo evolve, controllando che la cella non acceda ad uno stato proibito (es. morto che diventa suscettibile);
- il metodo start, contando il numero di infetti nella mappa e verificando che sia uguale al dato iniziale fornito.

Riferimenti

Il programma è presente al link https://github.com/ludorain/pandemic_SIRD