Un mare di altruismo

Filippo Mei
Università di Bologna
filippo.mei@studio.unibo.it

Federico Pennino
Università di Bologna
federico.pennino@studio.unibo.it

Italo Perrucci
Università di Bologna
italo.perrucci@studio.unibo.it

25 gennaio 2021

Sommario

Lo scopo di questo progetto è creare un modello per simulare le interazioni tra gli individui di una specie animale marina al fine di studiare l'evoluzione di comportamenti altruisti. L'altruismo di un individuo è determinato dal valore di un gene che influenza il tipo di interazioni che esso ha con i propri simili. Tali interazioni si riducono alla scelta tra condivisione o lotta per il cibo. Nel modello la spinta evolutiva è fornita da una limitatezza del cibo e quindi dell'energia necessaria al sostentamento e alla riproduzione. Si osserva che, sebbene il meccanismo della condivisione non sia particolarmente influente sull'aumento del grado di altruismo, si è rivelato essenziale per permettere la sopravvivenza della specie nel caso in cui sia presente anche il meccanismo della lotta per il cibo. Si nota altresì che il dispendio energetico della lotta ha un peso significativo nella selezione di individui più altruisti. In generale quindi i due meccanismi utilizzati contribuiscono allo sviluppo di comportamenti altruisti nella popolazione.

1. Introduzione

Definiamo come altruista un comportamento che va a vantaggio di un altro organismo, non strettamente imparentato, che è apparentemente svantaggioso per l'organismo che esibisce tale comportamento, dove vantaggio e svantaggio sono definiti in termini del contributo alla fitness dell'individuo, cioè al suo successo riproduttivo [1].
Nel nostro modello il comportamento altruista si traduce nella condivisione del cibo fra coppie di individui: tale comportamento sfavorisce nell'immediato l'individuo che lo esibisce, portandolo a privarsi di parte dell'energia che acquisirebbe nutrendosi, necessaria al sostentamento e alla riproduzione, andando così a ridurre la sua "fitness personale", ovvero il numero di discendenti che esso genera. D'altra parte però la sua "fitness inclusiva", ovvero la somma della fitness personale e della fitness indiretta dell'individuo, dove quest'ultima è il numero di discendenti generati da quegli individui che condividono geni simili ai suoi con i quali egli ha condiviso il cibo e che quindi ha aiutato a far sopravvivere e di conseguenza a riprodursi, potrebbe aumentare nel lungo periodo. Viceversa, quegli individui non altruisti che decidono di lottare con gli altri per impossessarsi del cibo, vincendo lo scontro aumentano, nell'immediato, la loro fitness personale, dato che non condividendo il cibo ne traggono il massimo dell'energia, ma la loro fitness inclusiva diminuisce nel lungo periodo, perché gli individui con i geni simili ai loro, che sono quelli con cui generalmente lottano, uscendo sconfitti dallo scontro, si ritrovano con meno energia e quindi con una minore probabilità di riprodursi.
Con questo modello vogliamo quindi indagare se comportamenti altruisti riescano ad emergere da regole che premiano l'individualità.

È bene sottolineare che, a differenza di altri modelli dove la selezione è artificiale, cioè la fitness viene interpretata come l'ottimizzazione di una certa funzione, in questo modello la selezione è naturale, ovvero il sistema evolve sotto alcuni vincoli imposti dall'ambiente, in questo caso la limitata disponibilità di cibo e quindi di energia, e la fitness quindi emerge spontaneamente dalle regole del sistema.

2. Modello

2.1. Individui

Ogni individuo, di seguito anche chiamato pesce per motivi che saranno chiariti in seguito, è caratterizzato da tre tratti genetici:

- velocità: un numero intero che determina il numero massimo di passi che può compiere ad ogni turno, ad ogni unità corrisponde un passo;
- energia di trigger: determina l'energia minima oltre la quale è disposto a riprodursi, col vincolo che sia maggiore o uguale all'energia consumata per la riproduzione, altrimenti l'individuo potrebbe decidere di riprodursi anche non avendo un'energia sufficiente per farlo andando incontro così alla morte, cosa di per sé poco realistica;
- altruismo: varia nel range $[0, 100]$, determina il tipo di interazioni che ha con i propri simili.

I valori assunti dai geni sono distribuiti casualmente all'inizio della simulazione. Gli individui hanno un'energia il cui valore massimo è $E_{max} = 1$, che può essere consumata e ricaricata, quando si esaurisce l'individuo muore e viene eliminato dalla mappa. Dopo un determinato numero di epoche, impostabile in input, l'individuo muore di vecchiaia a prescindere da quanta energia abbia. L'unico modo a disposizione per aumentare la propria energia è mangiare il cibo sparso sulla mappa il cui valore nutrizionale, ovvero la quantità di energia che fornisce, è impostabile in input.

Allo stesso modo la quantità di cibo inizialmente presente sulla mappa e lo spawn rate, ovvero il numero di nuovo cibo che compare casualmente sulla mappa ad ogni epoca, sono impostabili in input. Una frazione del cibo totale, il cui valore è altresì impostabile in input, può fornire un bonus energetico a quegli individui i cui gradi di altruismo, sommati tra di loro, siano sufficientemente alti, come sarà meglio specificato in seguito. I modi per consumare energia sono molteplici:

- movimento: ad ogni passo compiuto viene sottratta un'energia $E_p = \alpha$, con α un parametro impostabile in input;
- riproduzione: l'energia sottratta ad ognuno dei due individui nella riproduzione è la stessa, impostabile in input;
- lotta: quando due individui, etichettati dagli indici i,j lottano per ottenere il cibo l'energia persa dall'individuo i-esimo è:

  $$E = \beta \cdot \frac{E_j}{E_i + E_j}$$

  con E_i, E_j energie correnti dei due individui, β coefficiente impostabile in input. L'energia consumata da un individuo nella lotta è, quindi, proporzionale all'energia del proprio contendente normalizzata dalla somma delle due energie, in questo modo la lotta sarà molto dispendiosa quando l'altro individuo ha molte energie da spendere mentre sarà poco dispendiosa nel caso in cui l'avversario ha poche energie a disposizione.

Non esiste una distinzione fra i sessi perchè non rilevante per quello che vogliamo studiare, ogni individuo può quindi accoppiarsi con qualsiasi altro individuo. Gli individui possono rilevare la presenza del cibo e dei propri simili solo entro un'area limitata, di seguito indicata come campo sensoriale.

2.1.1 Movimento

Poichè gli animali in generale seguono l'istinto di trovare cibo e riprodursi abbiamo modellato il loro comportamento basandoci sul seguente processo decisionale. Prima di fare un

passo, ogni individuo sonda l'area ricoperta dal suo campo sensoriale:

- se trova del cibo si muove verso di esso finché non lo raggiunge, dandogli priorità assoluta; quando un individuo arriva in una cella immediatamente adiacente a quella che contiene il cibo, prima di nutrirsene sonda il proprio campo sensoriale alla ricerca di altri individui:
  - se non ne trova allora procede normalmente a nutrirsi;
  - se ne trova qualcuno entrano in gioco i meccanismi di condivisione del cibo o di lotta per esso: l'individuo che arriva sul cibo valuta la somma del grado di altruismo proprio e del primo altro individuo che vede:
    - se la somma è maggiore o uguale a 100 allora scatta il meccanismo della condivisione;
    - se la somma è minore di 100 allora lotta scatta il meccanismo della lotta;
- se non trova del cibo ma un altro individuo allora:
  - se entrambi hanno un'energia superiore alle proprie energie di trigger inizia a muoversi verso di esso: ad ogni passo si sottrae l'energia consumata per compierlo e si sonda il campo sensoriale:
    - se ha ancora un'energia sufficiente, del cibo non è entrato nel suo campo sensoriale e l'altro individuo vi è ancora presente e se quest'ultimo ha a sua volta energia sufficiente allora continua l'inseguimento:
      - una volta arrivato nelle immediate vicinanze dell'altro, se in quel momento entrambi hanno ancora energia sufficiente si riproducono.
- se non trova né cibo né individui si muove casualmente in una delle otto celle circostanti.

2.1.2 Condivisione

Quando la somma dell'altruismo dei due individui supera il valore di 100, i.e. quando $A_i + A_j \geq 100$ allora l'individuo che è arrivato per primo sul cibo, etichettato dall'indice i, condivide con l'altro la seguente quantità di cibo:

$$Q_i = \frac{c}{2} \cdot \frac{E_j}{E_i + E_j} + \frac{c}{2} \cdot \frac{A_j}{A_i + A_j}$$

con E_i ed E_j energie correnti dei due individui, c costante che determina la quantità di energia fornita dal cibo. In questo modo il cibo ricevuto aumenta con l'aumentare del grado di altruismo dell'individuo che lo condivide e con l'aumentare della sua energia ovvero, più praticamente, col diminuire della sua fame. Si noti che la somma del cibo ottenuto dai due individui sarà sempre pari a c.
Abbiamo detto che una frazione del cibo totale può fornire energia bonus: possiamo pensare a questo bonus come a un banco di pesci più piccoli, di cui i nostri individui si vogliono nutrire. Se i due collaborano allora riusciranno entrambi a catturarne un numero maggiore rispetto a quanto non avrebbero fatto singolarmente.
Questo bonus può essere ottenuto solo dagli individui che siano disposti a collaborare quindi questo si verifica solo quando la somma dei due gradi di altruismo è alta, cioè quando $A_i + A_j \geq 150$. In questo caso il valore nutrizionale del cibo c' da sostituire a c nella formula per Q_i sarà:

$$c' = c \cdot (1 + 3x)$$

con c valore nutrizionale del cibo impostato in input e x una variabile casuale distribuita nel range $[0, 1]$. Notiamo che tale bonus fornisce generalmente un'energia significativa.

2.1.3 Lotta

Quando la somma dell'altruismo dei due individui è inferiore al valore di 100, i.e. quando $A_i + A_j < 100$ allora i due lottano. La probabilità che ha l'individuo $i - esimo$ ovvero quello

che arriva per primo sul cibo di vincere è

$$P_i = \frac{E_i}{E_i + E_j}$$

con E_i ed E_j energie correnti dei due individui. Tale probabilità è, quindi, proporzionale alla propria energia normalizzata dalla somma delle due energie, in questo modo all'aumentare della propria energia e al diminuire dell'energia del contendente aumentano le possibilità di vittoria. La vittoria è determinata tramite l'estrazione di un numero casuale x distribuito uniformemente nel range $[0,1]$: se $x < P_i$ allora vince l'individuo $i-esimo$, altrimenti vince l'avversario. Dopo la lotta si calcola l'energia persa dai due contendenti: se il vincitore è ancora vivo mangia tutto il cibo mentre lo sconfitto se ne va a mani vuote, se invece il vincitore muore per lo sforzo allora è lo sconfitto a cibarsi.

2.1.4 Riproduzione

Quando due individui si riproducono danno vita a tre figli che vengono subito posizionati sulla mappa nelle celle nelle immediate vicinanze dei due genitori. Ogni gene di ogni figlio può mutare indipendentemente dagli altri, la probabilità di mutazione è impostabile in input. Un gene, se non mutato, viene ereditato senza modifiche da uno dei due genitori, scelto casualmente: il valore assunto da quel gene sarà lo stesso di quello del genitore da cui ha ereditato. Ognuno dei tre geni muta in modo diverso:

- altruismo: distribuito casualmente nel range $[0,100]$;
- velocità:
$$v = 1 + x$$
con x distribuito casualmente nel range $[0,10]$, in questo modo ci assicuriamo che la velocità sia sempre maggiore di 0;
- energia di trigger:
$$E_{tr} = \gamma + x(1-\gamma)$$
con γ energia spesa per la riproduzione, impostata in input e x distribuito casualmente nel range $[0,1]$, in questo modo l'energia di trigger risulta sempre maggiore di γ e non supera il valore di 1.

2.2. Ambiente

Gli individui si muovono e interagiscono in un habitat marino, la cui scelta, a differenza di un habitat terrestre, ci permette di ignorare la necessità del dover bere in quanto la fauna marina soddisfa a questo bisogno semplicemente filtrando l'acqua circostante. Nella simulazione l'ambiente è costituito da una griglia bidimensionale quadrata divisa in celle di lato pari a 30 celle, ognuna delle quali può essere occupata al più da un individuo o dal cibo oppure rimanere vuota (Figura 1). Il cibo sulla griglia è indicato da una cella di colore verde. Gli individui sono rappresentati da una cella di colore variabile tra rosso e nero a seconda del loro grado di altruismo: il nero corrisponde a un basso grado di altrui-

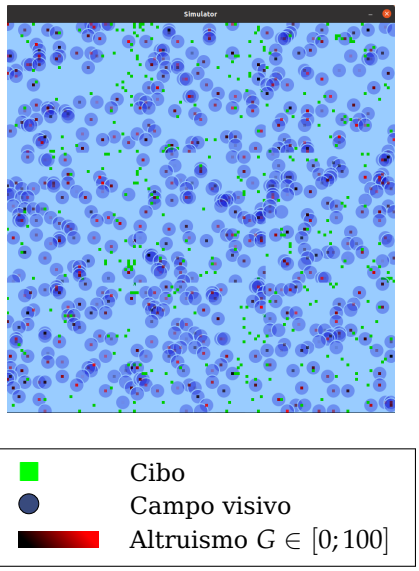

Figura 1: *Rappresentazione grafica della griglia bidimensionale quadrata divisa in celle di lato 30 celle: il cibo è di colore verde, gli individui hanno colore che va dal nero (basso grado di altruismo) al rosso (alto grado di altruismo) e sono circondati da un cerchio azzurro che rappresenta il campo sensoriale.*

smo, il rosso a un alto grado; sono circondati da un cerchio di colore blu che rappresenta il loro campo sensoriale: in realtà il campo sensoriale è un quadrato di 5 celle per lato centrato sull'individuo per un totale di 24 celle sondabili, come mostrato in Figura 2; abbiamo deciso di utilizzare un cerchio nella rappresentazione grafica perchè permetteva una resa estetica più chiara ed elegante. Man mano che l'energia diminuisce la cella rappresentante l'individuo e il suo campo sensoriale diventano sempre più trasparenti fino a scomparire del tutto con l'occorrere della morte.

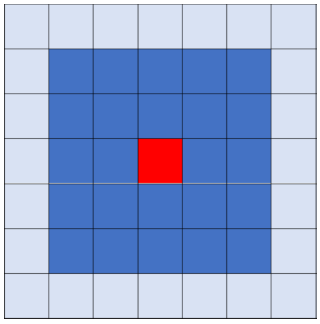

Figura 2: *Rappresentazione grafica del campo sensoriale (celle di colore blu) dell'individuo (cella di colore rosso).*

2.3. Interfaccia grafica

Dopo l'esecuzione del programma compare una finestra dalla quale è possibile fornire in input varie costanti che saranno utilizzate dal simulatore (Figura 3). Il simulatore crea nove campi diversi all'interno dei quali i pesci iniziano con energia e geni distribuiti casualmente. I campi non sono collegati tra di loro ma sono a sé stanti, nonostante appaiano disegnati senza confini evidenti. Le costanti fornite in input si applicano ad ogni campo e sono:

- Numero iniziale di pesci: numero di pesci inizialmente presente sulla mappa;
- Numero iniziale cibo: numero di celle piene di cibo inizialmente presente sulla mappa;
- Spawn rate cibo: numero di cibo che compare ad ogni epoca sulla mappa;
- Energia per passo: energia consumata nel compiere un passo, precedentemente indicata con α;
- Coeff. en. combattimento: coefficiente β che compare nella formula che determina l'energia spesa nella lotta;
- Energia cibo: quantità di energia fornita dal singolo cibo, precedentemente indicata con c;
- Costo en. riproduzione: costante che determina l'energia spesa per riprodursi, precedentemente indicata con γ;
- Frazione banchi: frazione dei banchi di pesci più piccoli sul totale del cibo che compare ad ogni epoca sulla mappa;
- Probabilità mutazione: probabilità che un gene muti;
- Tempo max di vita: numero di epoche trascorse le quali l'individuo muore di vecchiaia.

Si può inoltre decidere di eliminare la condivisione o la lotta dal modello deselezionando le apposite flag.

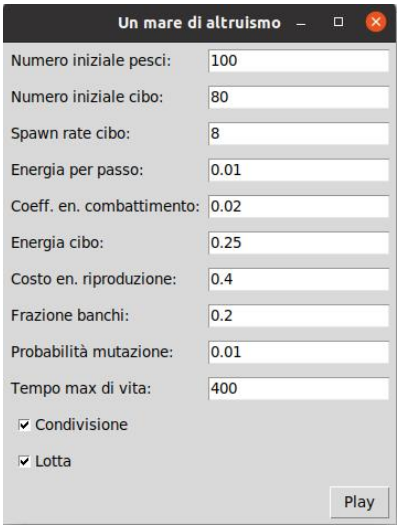

Figura 3: *Finestra iniziale nella quale inserire i parametri della simulazione. Notiamo che è possibile eliminare la condivisione o la lotta dal modello.*

Premendo sul pulsante “Play” si avvia la simulazione. Si aprono così due finestre: una della simulazione vera e propria con i nove campi popolati dagli individui chiamata, per l’appunto, “simulator” (di dimensione fissa) e una con vari grafici chiamata “stats” (di dimensione regolabile, vedi Figura 4). Nelle prime tre colonne troviamo i grafici di come variano la quantità di cibo (in verde) e il numero di individui (in rosso) al passare delle epoche, aggiornati ad ogni epoca. Nell’ultima colonna abbiamo, dall’alto verso il basso:

- un grafico dell’età media degli individui divisa per la quantità di cibo presente sulla mappa in funzione dell’altruismo medio diviso allo stesso modo;
- un grafico della quantità di cibo in funzione del numero dei pesci;
- un istogramma della distribuzione dei valori dell’altruismo.

Questi vengono aggiornati ogni 100 epoche: ogni punto dei due grafici è associato ad un campo in un’epoca, l’istogramma è la sovrapposizione degli istogrammi del singolo campo nella singola epoca per tutti i nove campi nelle 100 epoche appena trascorse.

3. Sviluppo

In questa sezione verrà discussa la struttura generale del software realizzato in questo progetto. È stato scelto di sviluppare questo programma usando due linguaggi: C++ e Python. La motivazione è dovuta principalmente al fatto che C++ (grazie al suo compilatore) risulta essere molto prestante, cosa in cui Python (essendo interpretato) non eccelle. Python, tuttavia, è un linguaggio molto semplice da usare, andando quindi a essere facilmente fruibile per *task* in cui le prestazioni sono relativamente importanti.
Python è stato usato per la costruzione della GUI iniziale (in cui è possibile selezionare i valori della simulazione) e per la visualizzazione dei dati in *live*. C++, invece, è stato usato per lo sviluppo dell’effettivo simulatore.

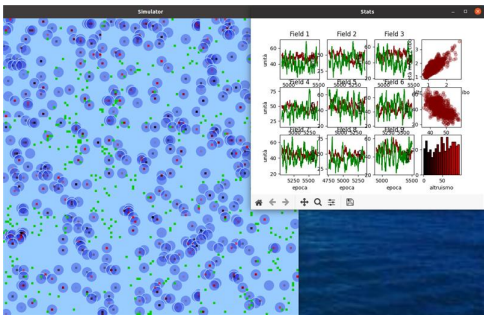

Figura 4: *A sinistra, interfaccia grafica della simulazione con i nove campi popolati dagli individui. A destra grafici della quantità di cibo (in verde) e del numero di individui (in rosso) rispetto alle epoche per ogni campo e nella colonna più a destra, dall’alto verso il basso: grafico dell’età media divisa per il numero di cibo in funzione dell’altruismo medio diviso per il numero di cibo, grafico del numero di cibo in funzione del numero di pesci, istogramma della distribuzione dei valori dell’altruismo di tutti gli individui.*

Per quanto riguarda la comunicazione tra i software sviluppati in Python e il simulatore sviluppato in C++ sono state usate due tecniche diverse.
La GUI iniziale (`main.py`) trasmette i valori al simulatore in C++ tramite `settings.json`.
Invece, le informazioni sullo stato attuale del simulatore vengono comunicate al software che ha il compito di disegnare i grafici (scritto in Python) tramite un sistema di *Inter Process Communication*.
È possibile visionare il codice sorgente del progetto tramite il seguente repository GitHub. Per le informazioni su come installare tutte le dipendenze del sistema e su come eseguire il software si rimanda al file `README.md` nel repository.

3.1. Ambiente di sviluppo

Come build system è stato scelto di utilizzare CMake. CMake è un tool per velocizzare la scrittura di file `.MakeFile`. Permette, inoltre, di automatizzare la ricerca dei percorsi e l’installazione delle librerie esterne.

Per lo sviluppo della parte grafica si è fatto uso di `GTKMM`, un wrapper in C++ della liberia standard di `GTK` scritta in C. Oltre alle funzioni classiche di `GTK`, `GTKMM` include le typesafe callback e un completo set di widget facilmente estendibili grazie all'ereditarietà.
Durante lo sviluppo si è sempre cercato di fare il minor ricorso possibile a librerie C++ non standard. Tuttavia, in alcuni casi si è fatto ricorso anche all'utilizzo di alcune funzioni della raccolta di librerie open source Boost.
Per quanto riguardo la costruzione della GUI iniziale è stato usato TkInter, una libreria in Python che permette di creare interfacce grafiche.
Per la visualizzazione dei dati in live si è scelto, invece, di usare MatPlotLib, una libreria per la creazione di grafici per il linguaggio di programmazione Python e la libreria matematica NumPy.
Infine, per l'analisi dati si è fatto uso di ROOT, un pacchetto software orientato ad oggetti di analisi dei dati sviluppato dal CERN.

3.2. `main.py`

Lo script `main.py` è il punto di ingresso del progetto. Una volta eseguito verrà concessa all'utente la possibilità di selezionare i valori con cui poi verrà avviato il simulatore.
La costruzione dell'interfaccia avviene sulla base delle informazioni salvate nel file `settings.json`. In Figura 3 viene mostrato un esempio di `main.py` in esecuzione.
Gli input richiesti possono essere di tipo numerico (interi o in virgola mobile) oppure booleano, dei valori di default sono presenti fin dal primo avvio. Una volta premuto *"Play"* il file `settings.json` verrà sovrascritto con i valori correnti e verrà avviato il simulatore in C++.

3.3. `settings.json`

Il file `settings.json` è un file di impostazioni che serve alla costruzione dell'interfaccia grafica di `main.py`.
I valori presenti nel file devono seguire la struttura indicata in Figura 5.

```
{
    ...
    "nome_campo": {
        "display": true,
        "type": "int",
        "name": "Etichetta",
        "value": 45
    }
    ...
}
```

Figura 5: *Struttura di un campo del file `settings.json`*

Il campo *display* può assumere solamente valori booleani, serve a indicare a `main.py` se quel determinato campo debba essere mostrato nell'interfaccia oppure no. Il campo *type* serve a dichiarare che tipo di valori può assumere quel determinato campo. Può assumere come valori "int", "float" oppure "bool". Nel caso in cui assuma come valore "int" oppure "float" *main.py* renderizzerà una label, invece, nel caso in cui il valore sia settato su "bool" verrà renderizzato un check-button. Il campo *value* serve, invece, per assegnare il testo dell'etichetta da renderizzare di fianco al campo di input. Infine, *value* è il valore che verrà assegnato al widget.

3.4. Struttura del simulatore

Il punto di ingresso del codice sorgente del simulatore è `src/main.cpp`. Una volta avviato il simulatore per prima cosa verranno recuperati i valori in `settings.json`. I valori presenti nel file verranno assegnati alle variabili di ambiente presenti all'interno della classe `src/Utils/Utils`.
Verrà a questo punto avviato un nuovo processo dove sarà eseguito lo script in Python che ha il compito di disegnare i grafici, descritto nella sezione 3.7.
Solo a questo punto verrà istanziato un nuovo oggetto che fa riferimento alla classe `src/area/Area`. Questa classe ha il compi-

to di istanziare e di disegnare i nove campi. Le istanze delle classi lavoreranno tutte in parallelo tra loro. Inoltre, a ogni nuova iterazione da parte di uno dei campi la classe `src/area/Area` aggiornerà la finestra.

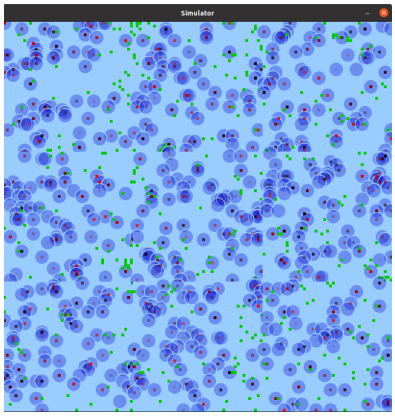

Figura 6: *Simulatore in esecuzione*

Ogni campo viene descritto dalla classe `src/field/Field`. Un campo è composto da una matrice (dimensione $30X30$) di puntatori ad oggetti della classe `src/tile/Tile`. Sono previste, inoltre, due specializzazioni della classe `src/tile/Tile` ovvero `src/tile/fish/Fish` che descrive un pesce nel campo e la classe `src/tile/food/Food` che descrive un'unità di cibo nel campo.

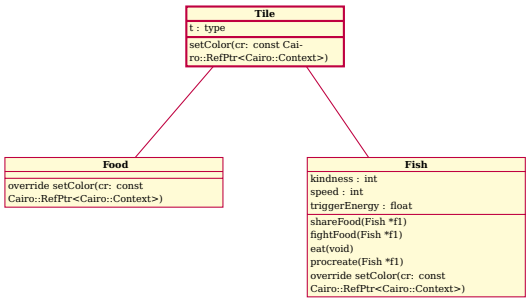

Figura 7: *Gerarchia classi `Tile`*

Nel caso in cui una casella sia vuota avrà valore `nullptr`, invece, nel caso in cui in quella casella sia presente un pesce oppure del cibo vi sarà un puntatore all'oggetto di interesse. Il comportamento e le interazioni tra i componenti del campo seguono le specifiche descritte nella sezione 2. Da notare come ad ogni iterazione del singolo campo viene aggiunta una riga al file di log `values.csv`.
Rimane da specificare che a ogni nuova iterazione del singolo campo avviene uno scambio di informazioni, seguendo la specifica descritta nella sezione 3.5, con lo script in Python descritto nella sezione 3.7.

3.5. *Inter Process Communication*

Come specificato in precedenza ad ogni avvio del simulatore corrisponde l'avvio di un nuovo processo che esegue lo script in Python descritto nella sezione 3.7. Il simulatore e lo script in Python comunicano tramite una *pipe*.
A ogni nuova iterazione del simulatore viene inviata tramite la *pipe* una stringa separata dal carattere di ';' contenente il numero di colonna, il numero di riga, il numero di pesci sul campo, la quantità di unità di cibo presenti sul campo e l'epoca attuale.
Una volta inviato questo messaggio vengono inviati n messaggi contenenti le informazioni relative a ogni singolo pesce, con n numero di pesci. La struttura di questo messaggio, sempre con i valori separati da ';', è la seguente: grado di altruismo, velocità, energia necessaria a riprodursi, tempo di vita corrente e quantità di energia corrente.

3.6. `values.csv`

`values.csv` è il file di `log` della simulazione generale. Questo è, inoltre, il file su cui viene fatta l'analisi dati descritta in sezione 3.8.
Il numero di colonne del file è variabile. Le prime quattro colonne sono sempre necessarie, tutte le successive, invece, variano in base al numero di pesci.
Le prime quattro colonne rappresentano in ordine: il numero identificativo del campo, l'epoca, il numero di pesci sul campo e la quantità di unità di cibo presenti sul campo. Le successive colonne sono $n \cdot 5$ con n numero di pesci. Le informazioni presenti nelle cin-

que caselle dedicate a ogni pesce sono: grado di altruismo, velocità, energia di trigger, età e quantità di energia corrente.

3.7. Live Data

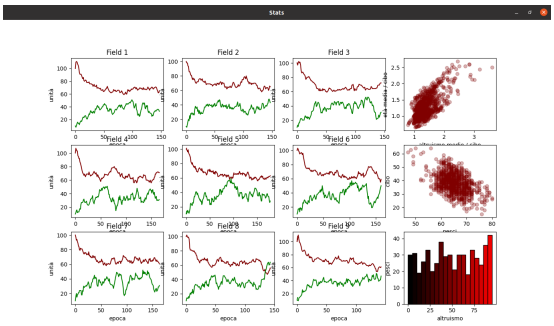

Figura 8: *Grafici live in esecuzione*

Questo script ha il compito di mostrare a tempo di esecuzione dei grafici sullo stato attuale del simulatore. È composto da due thread, uno che consuma i dati ricevuti dal simulatore e uno che aggiorna il grafico di `MatPlotLib`. Viene mostrato un esempio di esecuzione in Figura 8.

3.8. Root

Per analizzare i dati della simulazione trascritti nel file CSV è stato utilizzato il software CERN ROOT, un framework open source scritto principalmente in C++ e disponibile su Linux, macOS e Windows. Viene mostrato in Figura 9 il software di ROOT in esecuzione.
L'estrapolazione dei dati e la realizzazione dei grafici sono state effettuate mediante una macro scritta in C++ con l'aiuto di funzioni di ricerca scritte appositamente per il file CSV. Tutti i risultati, grafici compresi, vengono salvati in un file .ROOT per una successiva analisi.

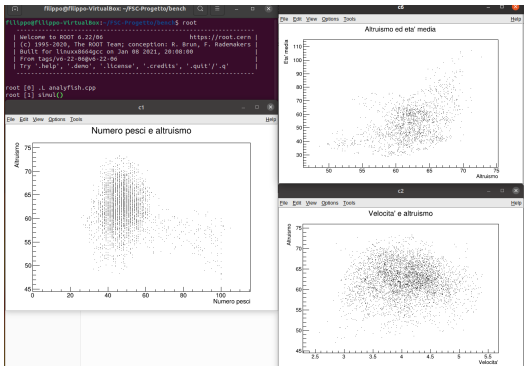

Figura 9: *Software ROOT in esecuzione su Ubuntu dal terminale. Si mostrano i comandi necessari a compilare la macro ed eseguire il codice affiancati da alcuni grafici prodotti.*

4. Analisi dei risultati

4.1. Metodo di analisi

Il programma di simulazione offre la possibilità di impostare i valori di dieci parametri e attivare/disattivare due opzioni, è perciò necessario tenere fisse parte delle impostazioni nello studio di diverse simulazioni per permettere un'analisi il più possibile consistente. Sono stati sempre tenuti fissi i valori che costituiscono lo scheletro del modello:

- l'energia consumata per uno step (0.01);
- l'energia fornita dal cibo (0.25);
- l'energia necessaria alla riproduzione (0.40);
- l'età massima di un individuo (400).

Naturalmente è possibile modellare il sistema attraverso altri valori, sono questi però quelli che sono stati ritenuti più consoni in rifermento alla dimensione della griglia e alla quantità media di cibo presente. I valori assegnati agli altri parametri saranno poi specificati caso per caso.
È importante evidenziare che la scelta di avere nove simulazioni in contemporanea aiuta a stabilire la bontà dei risultati. Infatti, date delle condizioni iniziali, se tutti i sistemi si evolvono verso una certa direzione si tenderà a individuare un possibile rapporto causa-

effetto. Tuttavia, proprio per la natura dei sistemi complessi non è raro il verificarsi di casi nei quali i sistemi assumano comportamenti diversi, per cui sarebbe errato studiarne solo uno per poter trarre delle conclusioni.
Quindi, sebbene siamo lontani dai numeri necessari per un ensemble statistico vero e proprio, le nove simulazioni in parallelo (non interagenti tra di loro) forniranno determinati valori ed in caso di evoluzione simile si utilizzerà una media di essi o si utilizzeranno in gruppo.
Rimane da definire cosa sia stato considerato per "evoluzione simile"; per confrontare i nove sistemi tra loro individuiamo quattro comportamenti della curva della popolazione e di quella del cibo in un opportuno intervallo di tempo (dell'ordine di centinaia di epoche):

- crescita stabile;
- decrescita stabile;
- andamento sinusoidale;
- andamento caotico.

Detto ciò, si imposteranno un numero di pesci e cibo iniziali e lo spawn rate di quest'ultimo in modo che il sistema abbia la possibilità di raggiungere uno stato di equilibrio, necessario per lo studio dell'evoluzione dei geni nella popolazione. Per spawn rate troppo bassi la popolazione si estingue in breve tempo, rendendo impossibile un'analisi statistica significativa.

4.2. Individuazione equilibrio del sistema

Consideriamo il sistema in equilibrio quando il numero di pesci e di cibo presentano un andamento periodico di oscillazione (non necessariamente attorno allo stesso valore) in controfase (il picco di uno corrisponde con un minimo dell'altro) come in Figura 10.

Questo andamento si traduce nel modello in una corrispondenza tra massimi di cibo e minimi di pesci e viceversa, per cui andando a graficare questi due valori (cibo nelle ascisse e pesci nelle ordinate) ci aspettiamo un andamento rettilineo con pendenza negativa.

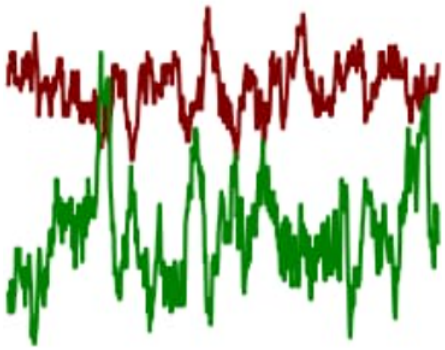

Figura 10: *Esempio di grafici in controfase*

D'altra parte nel caso il sistema non sia in equilibrio ci aspettiamo un andamento irregolare.
Per ogni campo, per ognuna delle ultime 200 epoche prima della fine della simulazione sono stati inseriti i punti in un grafico: notiamo che l'andamento atteso all'equilibrio è stato confermato (Figura 11), si nota la correlazione lineare tra le due grandezze, con pendenza negativa; in Figura 12 osserviamo invece l'andamento irregolare tipico di un sistema instabile.

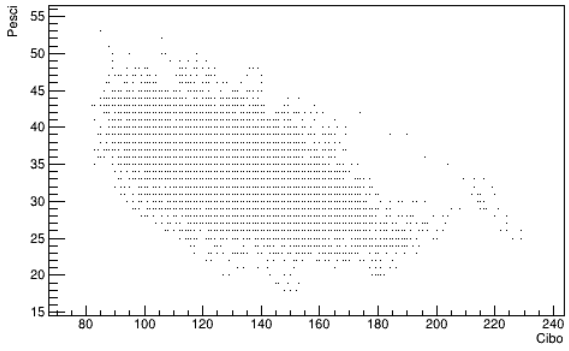

Figura 11: *Relazione lineare tra pesci e cibo*

4.3. Influenza delle interazioni

Impostati i parametri iniziali come in Tabella 1 sono state analizzate simulazioni con le meccaniche di condivisione e lotta attive o meno, in diverse combinazioni. L'output analizzato per ognuna di queste è un istogramma

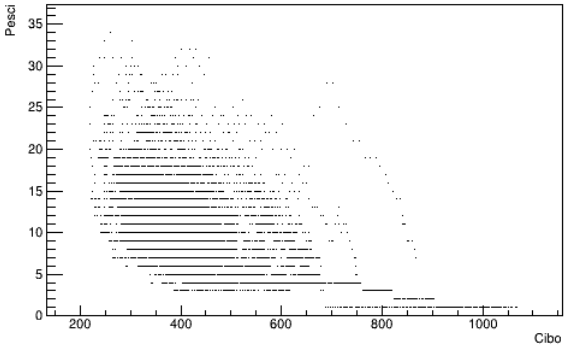

Figura 12: *Nessuna relazione tra pesci e cibo*

a nove intervalli riportante il valore medio di altruismo della popolazione nelle ultime 200 epoche nei rispettivi sistemi. Da questo ricaviamo il valore medio tra i nove campi $\bar{A}$ utilizzato per il confronto. È stato lasciato ad esempio un solo istogramma in Figura 13 mentre i valori di $\bar{A}$ sono confrontati in Tabella 2.

Parametro	Valore
Numero iniziale pesci	100
Numero iniziale cibo	180
Spawn rate cibo	10
Coeff. en. combattimento	2
Frazione banchi	0.2
Probabilità di mutazione	0.01

Tabella 1: *Impostazioni utilizzate*

Attivando la meccanica di lotta ma non quella di condivisione la popolazione sopravvive per troppo poco tempo per poter fare statistiche, in queste condizioni quindi non si può raggiungere un equilibrio.
Negli altri tre casi invece si raggiunge l'equilibrio ed è possibile confrontare i dati: la presenza di entrambe le meccaniche porta al valore di altruismo più alto mentre disattivarle entrambe porta a un valore più alto rispetto all'attivare solo la condivisione.

Naturalmente l'obiettivo di questa parte di analisi non è cercare di raggiungere l' $\bar{A}$ più

Condivisione	Lotta	$\bar{A}$
Non attiva	Non attiva	56.9
Attiva	Non attiva	51.2
Non attiva	Attiva	/
Attiva	Attiva	61.3

Tabella 2: *Valori osservati nelle diverse simulazioni*

alto possibile, bensì fare un confronto tra l'impatto delle diverse interazioni. Quello che emerge è che la condivisione in sé non aiuta solo i pesci più altruisti: infatti, con i parametri iniziali utilizzati, sembra che l'altruismo non incida sulla sopravvivenza una volta raggiunto l'equilibrio.
La condivisione ha però un ruolo fondamentale mentre è attiva la lotta: se la prima non è attiva la popolazione non sopravvive più di un centinaio di epoche, al contrario quando è attiva si raggiunge l'equilibrio e il valore $\bar{A}$ è il più alto tra i casi proposti.

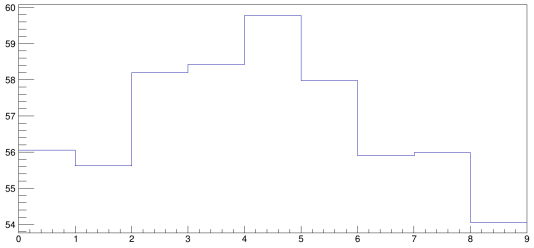

Figura 13: *Istogramma del valore medio di altruismo nei nove campi*

4.4. Relazione tra banchi e altruismo

Per questa analisi sono stati tenuti fissi i valori dei parametri inziali in Tabella 3 mentre sono state fatte più simulazioni con valori diversi della frazione di banchi sul cibo totale. In tutte le simulazioni erano attive sia la condivisione che la lotta. Come nel caso precedente sono stati confrontati gli $\bar{A}$ (calcolati allo stesso modo), riportati in Tabella 4.

Si nota immediatamente che l'abbondanza di banchi di pesci più piccoli non sia partico-

Parametro	Valore
Numero iniziale pesci	100
Numero iniziale cibo	180
Spawn rate cibo	10
Coeff. en. combattimento	2
Probabilità di mutazione	0.01

Tabella 3: *Impostazioni utilizzate*

Frazione banchi	$\bar{A}$
0	59.3
0.25	59.4
0.5	59.7
0.75	59.6
1	60.8

Tabella 4: *Valori osservati nelle diverse simulazioni*

larmente incidente sul valore medio di altruismo della popolazione. Infatti a meno di una leggera crescita di $\bar{A}$ l'unico vero impatto è stato in generale sulla maggiore dimensione della popolazione (al crescere della frazione di banchi) dato dalla maggiore quantità di cibo disponibile.

4.5. Relazione tra dispendio energetico della lotta e altruismo

Per questa analisi sono stati tenuti fissi i valori dei parametri inziali in Tabella 5 mentre sono state fatte più simulazioni con valori diversi del coefficiente di lotta. In tutte le simulazioni erano attive sia la condivisione che la lotta. Come nel caso precedente sono stati confrontati gli $\bar{A}$ (calcolati allo stesso modo) in Tabella 6.
Si mostra una correlazione importante tra il dispendio energetico per lottare e il valore $\bar{A}$. In particolare il "salto" maggiore lo si ha passando da un coefficiente $\beta = 0$ a $\beta = 0.5$, inoltre per ricreare una situazione limite si è impostato anche il valore del coefficiente $\beta = 10$, un valore così alto in generale determina la morte di entrambi i contendenti.

Parametro	Valore
Numero iniziale pesci	100
Numero iniziale cibo	180
Spawn rate cibo	10
Frazione banchi	0.2
Probabilità di mutazione	0.01

Tabella 5: *Impostazioni utilizzate*

Cefficiente lotta	$\bar{A}$
0	48.8
0.5	55.5
1	59.3
2	60.3
10	65.3

Tabella 6: *Valori osservati nelle diverse simulazioni*

4.6. Relazione tra velocità e altruismo

Per la realizzazione del grafico in Figura 14 sono stati messi in correlazione per ogni epoca e per ogni campo i valori medi di altruismo e velocità della popolazione.
Non è stata trovata alcuna particolare correlazione tra i due geni, anche utilizzando diverse impostazioni iniziali, in Tabella 7 sono specificate quelle relative al grafico proposto.

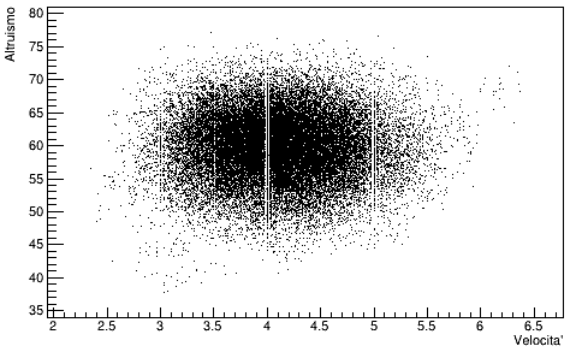

Figura 14: *Grafico a dispersione tra veloctà media ed altruismo medio*

Parametro	Valore
Numero iniziale pesci	100
Numero iniziale cibo	180
Spawn rate cibo	10
Coeff. en. combattimento	2
Frazione banchi	0.2
Probabilità di mutazione	0.25

Tabella 7: *Impostazioni utilizzate*

4.7. Relazione tra età e altruismo

Per la realizzazione del grafico in Figura 15 sono stati messi in correlazione per ogni epoca e per ogni campo i valori medi di altruismo ed età (epoche passate dalla nascita dell'individuo) della popolazione. I punti sono stati raccolti a partire da 400 epoche dopo l'inizio della simulazione (durata circa 5000 epoche) in modo da escludere le età artificialmente basse dovute alla generazione iniziale.
Impostando un'elevata probabilità di mutazione genetica (come mostrato in Tabella 8) è stato possibile fare in modo che venisse esplorato un ampio intervallo di valori del gene dell'altruismo, così da avere un più ampio numero di dati diversi su cui basare l'analisi.
Si nota che a bassi valori dell'altruismo l'età media della popolazione è minore, mentre spostandosi lungo il verso positivo delle ascisse la distribuzione dei punti si apre verso l'alto arrivando a toccare i massimi. Si conclude che il gene dell'altruismo contribuisca a fornire la possibilità di avere una popolazione più anziana, che non sarebbe possibile con valori minori.

Parametro	Valore
Numero iniziale pesci	100
Numero iniziale cibo	180
Spawn rate cibo	10
Coeff. en. combattimento	2
Frazione banchi	0.2
Probabilità di mutazione	0.25

Tabella 8: *Impostazioni utilizzate*

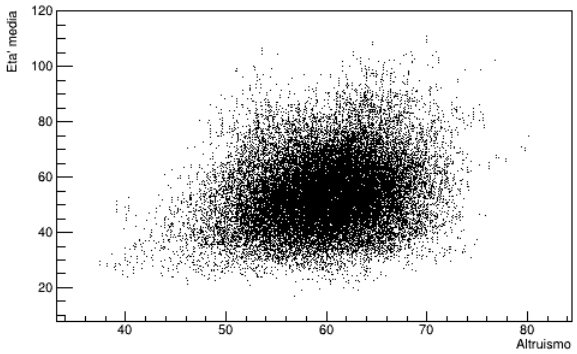

Figura 15: *Grafico a dispersione tra età media ed altruismo medio*

5. Conclusioni

È stato proposto un modello delle interazioni tra individui di una specie basato sulla trasmissione genetica di alcuni caratteri, regolato da nascita, riproduzione e morte degli individui che si sviluppano sotto la pressione evolutiva determinata dalla limitatezza delle risorse ovvero del cibo e di conseguenza dell'energia. Il modello si è rivelato capace di riprodurre situazioni che si riscontrano in natura come ad esempio il raggiungimento di una popolazione stabile, l'estinzione totale e fasi di crescita incontrollata.
In questo modello gli individui possono interagire cooperando o competendo per ottenere il cibo necessario alla vita e alla riproduzione. Tali interazioni sono governate dal grado di altruismo di ciascun individuo, rappresentato nel modello attraverso un gene.
Nello specifico, analizzando tale gene, è emerso che la condivisione in sé non sia particolarmente selettiva verso gli individui più altruisti poiché all'equilibrio non penalizza la sopravvivenza dei meno altruisti.
La condivisione diventa determinante quando gli individui hanno anche la possibilità di lottare: se essa non è attiva la popolazione si estingue in poco tempo, al contrario quando è attiva si raggiunge l'equilibrio ed il gene altruista è maggiormente sviluppato.

Riferimenti bibliografici

[1] The Evolution of Reciprocal Altruism BY Robert L. Trivers, The Quarterly Review of Biology, Volume 46, Number 1Mar., 1971, Stony Brook University

[2] Food Sharing and Nonhuman Reciprocal Altruism BY Karin Schneeberger, Encyclopedia of Evolutionary Psychological Science, Springer International Publishing Switzerland

[3] Reciprocal altruism in bats and other mammals BY Gerald S.Wilkinson, Ethology and Sociobiology, Volume 9, Issues 2–4, July 1988, Pages 85-100

[4] Selfish genes: a green beard in the red fire ant BY Laurent Keller Kenneth G. Ross, Nature, volume 394, pages 573–575 (1998)

[5] The Genetical Evolution of Social Behaviour. I, W. D. HAMILTON, J. Theoret. Biol. (1964) ‘7, 1-16

[6] An energy-based natural selection model, Noam Abadi, Guillermo Abramson, Preprint submitted to Physica A, November 16, 2020