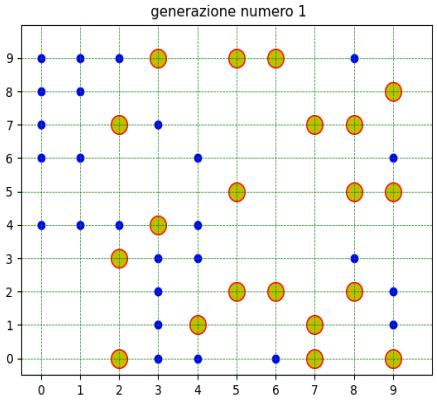
Algoritmi Genetici

Federico Astolfi, Tommaso Vassura

Giugno 2022

Sommario

In questo progetto una popolazione di creature viene distribuita in maniera aleatoria in una griglia bidimensionale. Tale ambiente presenta al suo interno delle torte disposte in posizioni casuali. Le creature hanno a propria disposizione quattro mosse (destra, su, sinistra, giù) per muoversi all'interno di questo reticolo e possiedono inoltre un'energia, la quale diminuisce di 1 a ogni mossa compiuta. L'unico modo in cui l'energia può venire reintegrata è raggiungendo una torta, e siccome una creatura con energia 0 viene considerata morta, l'unica possibilità per le creature di muoversi senza estinguersi è rappresentata dal raggiungimento delle torte distribuite intorno a loro.



Ogni creatura è in possesso di un "genoma", che consiste nell'associazione di una mossa a ogni possibile scenario in cui la creatura possa incorrere (e.g. una torta a destra, una torta su, vuoto a sinistra, vuoto giù), ovvero un elenco di 2^4 indicazioni su come muoversi nell'ambiente.
Nel corso del progetto più generazioni di tali creature vengono generate, e mentre quella iniziale è in possesso di mosse casuali ed energia pari a una certa soglia fissata, le generazioni successive hanno come energia la media dell'energia dei due genitori da cui discendono e come mosse una combinazione, ottenuta mediante crossover, delle mosse dei genitori.
Un altro scenario che viene poi preso in esame è il caso in cui oltre alle

torte siano presenti dei veleni, che abbassino l'energia della creatura che vi incorre.
Dal momento che i genitori vengono selezionati per la riproduzione di modo che la loro scelta sia tanto probabile quanto più la loro energia è alta rispetto a quella degli altri, lo scopo di questo progetto è di capire se si è in grado di far sviluppare a tali creature una sorta di istinto di sopravvivenza, il quale compaia a un certo punto nelle generazioni successive e mediante cui esse siano in grado di discriminare tra una scelta che permette loro di incrementare energia oppure di perderla.

Indice

1 Genetica

In questa sezione vengono motivate le scelte di carattere genetico, dettagliando le considerazioni che sono state fatte al fine massimizzare l'apprendimento e la stabilità delle popolazioni di individui. In quanto segue si intenderà che il problema proposto è stato risolto quando il programma produce una sequenza abbastanza lunga di generazioni di individui che oltre a sopravvivere imparano. Si è deciso di quantificare l'apprendimento di un individuo attraverso una quantità *greed* che si calcola contando in quanti dei 15 geni che contengono una torta (cioè tutte le sequenze di lunghezza quattro composte di 0 e 1 tranne la sequenza 0000, si veda sezione 2), l'informazione è "andare verso una delle torte". Per semplicità useremo *greed* dopo averlo normalizzato, da cui si ottiene: $\frac{1}{15} \leq greed \leq 1$. Si è proceduto seguendo lo schema euristico tipico di un algoritmo genetico:

1. Generazione casuale della prima popolazione di individui.
2. Applicazione della funzione di fitness agli individui appartenenti all'attuale popolazione.
3. Selezione degli individui migliori con probabilità proporzionali al fitness
4. Generazione di una nuova popolazione di individui ibridi a partire dagli individui scelti al punto 3.
5. Ripetizione della procedura a partire dal punto 2 utilizzando la nuova popolazione creata al punto 4.

Come specificato dal problema, la prima popolazione viene generata con mosse e coordinate sulla griglia casuali ed energia fissata; il fitness, in questo caso, è l'energia residua degli individui dopo il movimento. La creazione di individui figli, parte dalla selezione casuale di due genitori, estratti tra tutti i possibili con probabilità proporzionale all'energia rimasta. La media aritmetica delle energie dei genitori sarà l'energia posseduta da ogni figlio. Considerando che un individuo è tanto più adatto quanto si muove verso le torte limitrofe, e che questa è un'informazione localizzata in ogni singolo gene più che dal genoma nel suo insieme, si è optato per un crossover ad un punto. Questo non solo si impone per la propria semplicità ma, come si richiede ad un crossover adeguato, permette mediamente ai figli di ereditare le caratteristiche che hanno reso i genitori adatti, nel senso del fitness. Infine, per evitare degli ottimi locali e ampliare lo spazio di ricerca, abbiamo aggiunto la possibilità di mutazioni genetiche casuali: con una probabilità specificata dall'utente ogni nuovo individuo figlio potrebbe subire la mutazione. In tal caso viene estratto casualmente un gene e viene cambiata la mossa corrispondente al gene con probabilità uniforme (e.g. l'associazione "(torta, torta, vuoto, vuoto) : su" viene cambiata in "(torta, torta, vuoto, vuoto) : giù". Nella sezione 2 viene spiegata la scelta di struttura dati del genoma che è stata realmente usata nel programma). E' interessante

notare un tratto caratteristico del problema in esame che lo differenzia e complica rispetto ad alcuni esempi classici di utilizzo di algoritmi genetici. In effetti, come già notato, sono due gli aspetti da tenere in considerazione e ricercare per le generazioni: continuare a vivere e imparare (cioè aumentare *greed*). Tuttavia per alcune scelte dei parametri liberi (numero di individui, numero di torte ecc..), gli individui non hanno la possibilità di vivere abbastanza a lungo affinché la pressione evolutiva riesca a produrre un miglioramento e di conseguenza l'algoritmo genetico non si avvicina neanche ad una soluzione accettabile, perché non è possibile scegliere a priori quante generazioni fare. Per completezza citiamo anche la possibilità di inserire nell'ambiente bidimensionale del veleno, oltre alle torte. Quanto detto in questa sezione rimane invariato ad eccezione della lunghezza del genoma che diventa una struttura dati con 3^4 voci. Nella sezione 3 seguiranno le considerazioni sulla convergenza nella situazione semplice e in quella con il veleno.

2 Il programma

Passiamo dunque a trattare più nel dettaglio la parte computazionale del presente progetto.
Per semplicità, abbiamo definito l'**ambiente** costruendo una matrice contenente 0 per gli spazi vuoti e 1 per le torte (eventualmente anche 2 nel caso dei veleni), in cui, coerentemente con quanto descritto sopra, gli 1 (e i 2) vengono distribuiti casualmente all'interno della griglia. Abbiamo codificato il programma in Python, un linguaggio di programmazione orientato agli oggetti, che in quanto tale ci ha consentito di definire una *classe* **Creatura**, che abbiamo dotato di diversi attributi:

- **self.energia**, che rappresenta la "vita" della creatura. Una creatura con **self.energia** pari a 0 non è in grado di muoversi e viene considerata defunta. A ogni mossa compiuta dalla creatura, la sua energia viene decrementata di 1 se capita su una casella vuota, lasciata invariata se incorre in una torta, diminuita di 2 se inciampa in un veleno (nel caso tale sottrazione comporti che **self.energia** < 0, imponiamo **self.energia** $= \mathbf{0}$).

- **self.x** e **self.y**, ovvero delle coordinate random all'interno della griglia. Si fa notare che, visto che la creatura occupa una posizione all'interno di **ambiente** che è una matrice, con **self.x** si accederà alla riga in cui si trova la creatura e con **self.y** alla colonna corrispondente.

- **self.mosse**, ovvero un *dizionario* che rappresenta il genoma dell'individuo e contiene la mossa da compiere in ogni situazione possibile.
  Per comodità, abbiamo codificato l'informazione del movimento con 4 numeri: destra$\rightarrow$ 0, su$\rightarrow$ 1, sinistra$\rightarrow$ 2, giù$\rightarrow$ 3.
  Tali movimenti sono associati a stringhe composte da 4 numeri che indicano la presenza o meno delle torte (o dei veleni eventualmente):
  vuoto$\rightarrow$ 0, torta$\rightarrow$ 1, veleno$\rightarrow$ 2. Per proporre un esempio concreto, la

mossa **'0210' : 2** passa l'informazione che, nel caso in cui (leggendo da sinistra a destra) a destra ci sia una casella vuota, sopra ci sia un veleno, a sinistra una torta e giù una casella vuota, allora la creatura a cui questo gene è associato andrà a sinistra, ovvero sulla torta.

- **mate(self, other, mut_prob)**, ovvero un *metodo* che consente alla creatura **self** di accoppiarsi con un'altra **other**.
  In questo *metodo* si effettua il crossover tra **self** e **other** per ottenere una creatura figlia e successivamente si introduce, con una certa probabilità **mut_prob**, una mutazione all'interno del genoma, che sulla base di tale probabilità potrebbe modificare una singola mossa contenuta all'interno di **mosse**, sostituendo un'istruzione con un'altra (differente). L'energia della creatura figlia viene settata come la media di quelle dei due genitori, arrotondata per eccesso.

Abbiamo definito successivamente un metodo **movimento(creature, ambiente)**, che prende in input una creatura e una griglia ed effettua diverse operazioni.
In primis, nel caso in cui la creatura abbia energia sufficiente per muoversi, ne consente il movimento sulla base di quanto codificato nel suo genoma. Ancora una volta, si noti che i movimenti della creatura sono stati implementati tenendo conto che essa si sta muovendo su una matrice, in cui la prima coordinata indica la riga e la seconda la colonna. Successivamente, il *metodo* si occupa di rimuovere eventuali torte (o veleni) mangiati dalla creatura, di modo che non ne rimanga più traccia nell'ambiente.
Infine sono stati implementati i metodi riguardanti gli aspetti genetici:

- **rouletteSampling(list, fit)**: data una lista $list = \{C_i\}_i$ di creature e una $fit = \{f_i\}_i$ ($\forall i, f_i \geq 0$) con le rispettive energie estrae due genitori C_{i_1}, C_{i_2} con probabilità $\sum_i \frac{f_{i_1}}{f_i}$ e $\sum_i \frac{f_{i_2}}{f_i}$

- **crossover(dict1, dict2)**: dati due dizionari, restituisce un dizionario che rappresenta il genoma di un figlio ottenuto dai genitori con crossover ad un punto. Il punto in cui fare il taglio dei genomi non è un parametro della funzione per semplicità ed è inserito direttamente nell'implementazione del metodo.

- **getOffsprings(parents, npop, mutProb, scritte)**: è il metodo che si occupa di creare la generazione successiva a partire da **parents**. In particolare, usando i metodi citati sopra, crea **npop** figli, che hanno subito delle mutazioni con probabilità **mutProb**. L'ultimo parametro assume valori booleani e si occupa di far comparire le scritte o meno.

Per semplicità non dettagliamo le funzioni minori con scopi o implementazioni esigue.

3 I casi di studio

In questo capitolo ci occupiamo di analizzare alcuni casi particolarmente significativi tra le simulazioni svolte.

3.1 Rapporto ottimale

Lo scopo di questo esperimento è minimizzare il rapporto tra individui di ogni generazione e torte presenti nell'ambiente. Innanzitutto definiamo i nostri presupposti. Come anticipato, diremo di aver trovato una soluzione accettabile rispetto al problema di 'sviluppare un istinto di sopravvivenza' per quei parametri per cui si ottiene una sopravvivenza stabile nel tempo con una probabilità abbastanza alta. In particolare diremo che il programma converge (con determinati parametri) se con probabilità almeno 0.6 le popolazioni non si estinguono prima della centesima generazione. Dunque dei parametri con cui si ottiene convergenza faremo un'ulteriore analisi di non trivialità: controlleremo che gli individui abbiano effettivamente imparato e inoltre controlleremo che non sarebbe stato possibile ottenere lo stesso risultato di sopravvivenza con un approccio casuale al posto che genetico.

Le modalità dell'esperimento: fissiamo il limite inferiore di individui da cui partire a 20 e il limite superiore a 40. Quindi facciamo variare il numero di individui da 20 a 40 a passi di due e per ogni elemento $i \in$ **range(20, 40, 2)** $= [n_i]_i$ facciamo variare il numero di torte $j \in$ **range(60, 90, 3)** a 3 passi ogni iterazione, arrestandoci al primo valore j per cui la probabilità di successo è maggiore di 0.6; facciamo notare che ovviamente tale probabilità è stata ottenuta a posteriori statisticamente mediante prove ripetute.
In questo modo otteniamo al variare di i il minimo numero di torte t_i per cui si ha convergenza nel senso specificato sopra (figure 1).

Prima di procedere con l'interpretazione dei dati concludiamo l'esperimento calcolando il vettore dei rapporti $[v_i]_i = [\frac{t_i}{n_i}]_i$; ovviamente siamo interessati al miglior rapporto $min_i\{v_i\}_i$. I range di i, j si sono scelti con le seguenti considerazioni, per quanto riguarda il numero di individui: un numero troppo basso di individui generalmente non porta alla convergenza perché la prima generazione casuale ha un fitness troppo basso, mentre un numero troppo alto avvantaggia l'esplorazione nello spazio dei geni ma crea troppa competizione, dal momento che gli individui non hanno nessun meccanismo di collaborazione. Per quanto riguarda il numero di torte invece: un numero troppo basso semplicemente non porta alla convergenza e uno troppo alto disincentiva l'apprendimento data l'assenza di pressione evolutiva.
I risultati di questo esperimento rivelano che il vettore dei rapporti è decrescente in i $\forall i \leq 9$, anche se in realtà per 40 individui il programma non converge neanche con 90 torte. Il valore ottimale per questo set di dati è invece $v_9 = \frac{87}{38} = 2,28$. In pratica, pur di mettere torte a sufficienza, il vantaggio di avere una grande popolazione vince sullo svantaggio che gli individui competano per le stesse ri-

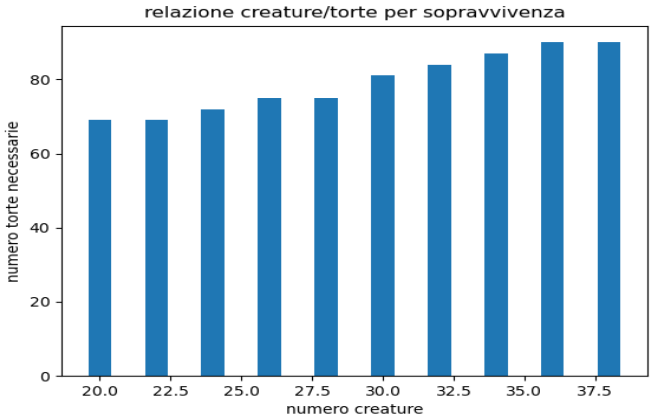
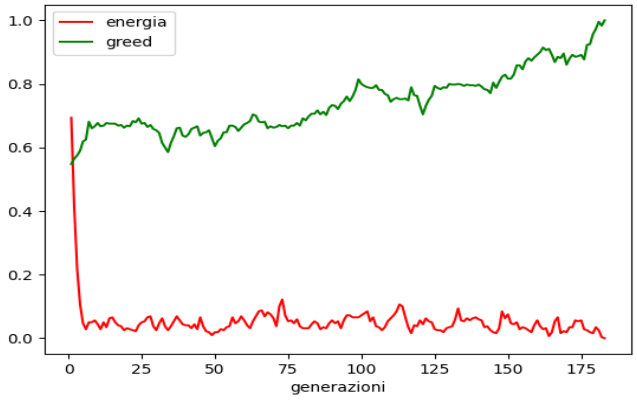


Figura 1: minimo t_i

sorse fino ad un valore di circa 40 individui. Ora seguiamo il controllo di non trivialità, assicurandoci che oltre a sopravvivere gli individui stiano imparando ad andare verso le torte, cioè che *greed* stia aumentando. Visualizziamo questo dato in un grafico:



Si nota che gli individui si sono estinti alla centosettantacinquesima generazione e *greed* si è avvicinato molto al valore massimo 1, rispetto al valore di partenza (abbiamo calcolato che il valore atteso di *greed* per generazioni casuali è 0,53). Infine, controlliamo che i parametri inseriti non siano così permissivi da permettere la sopravvivenza anche di popolazioni che si muovono casualmente: di questo otteniamo una conferma lanciando una versione modificata del programma in cui i figli vengono generati senza premiare i migliori, si ottiene una rapida estinzione: non ci sono abbastanza torte per permettersi di non imparare.

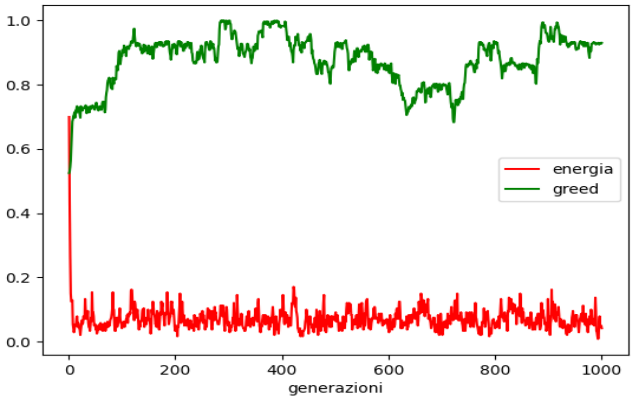


Figura 2: Grafico ottenuto facendo 5 mosse, 24 individui e 72 torte

3.1.1 Osservazioni sull'esperimento del rapporto ottimale

Nei casi con parametri che portano alla convergenza, notiamo due fenomeni su cui ci soffermiamo:

a *Greed* non converge ad 1 come si potrebbe sperare, ma piuttosto oscilla con picchi "vicini" ad 1;

b Si notano vari casi in cui, proprio in seguito ad un "avvicinamento" di *greed* al valore massimo, avviene l'estinzione della popolazione.

Nelle figure 2 e 3 dei grafici in cui si osservano rispettivamente i fenomeni a e b.

Innanzitutto ricordiamo che con *greed* si intende la media delle *greed* di ogni individuo di una generazione, quindi ci limitiamo a studiare il livello di apprendimento medio di ogni generazione. Il primo fenomeno dipende fortemente dalla natura aleatoria del problema: come abbiamo già discusso, parliamo di convergenza in un'accezione ampia che include gli che individui hanno avuto un apprendimento apprezzabile. In effetti, nei casi presi in esame *greed* tende velocemente a salire ed avvicinarsi al valore massimo grazie al meccanismo di selezione dei genitori: la selezione avviene usando come fitness l'energia, ma nelle fasi iniziali gli individui con energia maggiore sono quelli con *greed* migliore e questo motiva il miglioramento iniziale che era sperato. Una volta che l'apprendimento medio delle generazioni si avvicina al massimo teorico entra in gioco l'aleatorietà della disposizione delle torte. Infatti con un *greed* molto alto ed omogeneo il criterio di selezione in base all'energia smette di premiare chi raggiunge meglio le torte (perché ogni individuo lo fa molto bene) ed inizia a premiare chi ha avuto la maggiore "fortuna" di trovarsi le risorse nelle vicinanze.

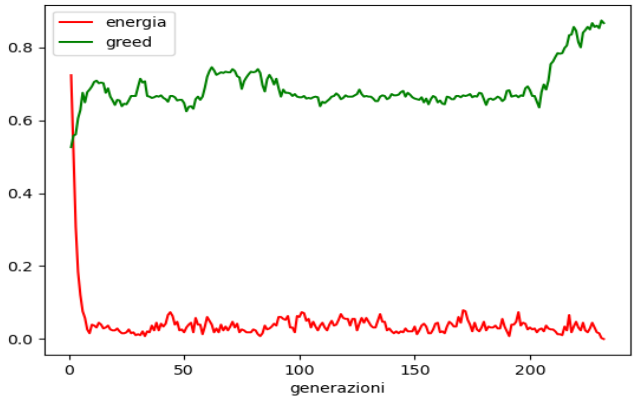


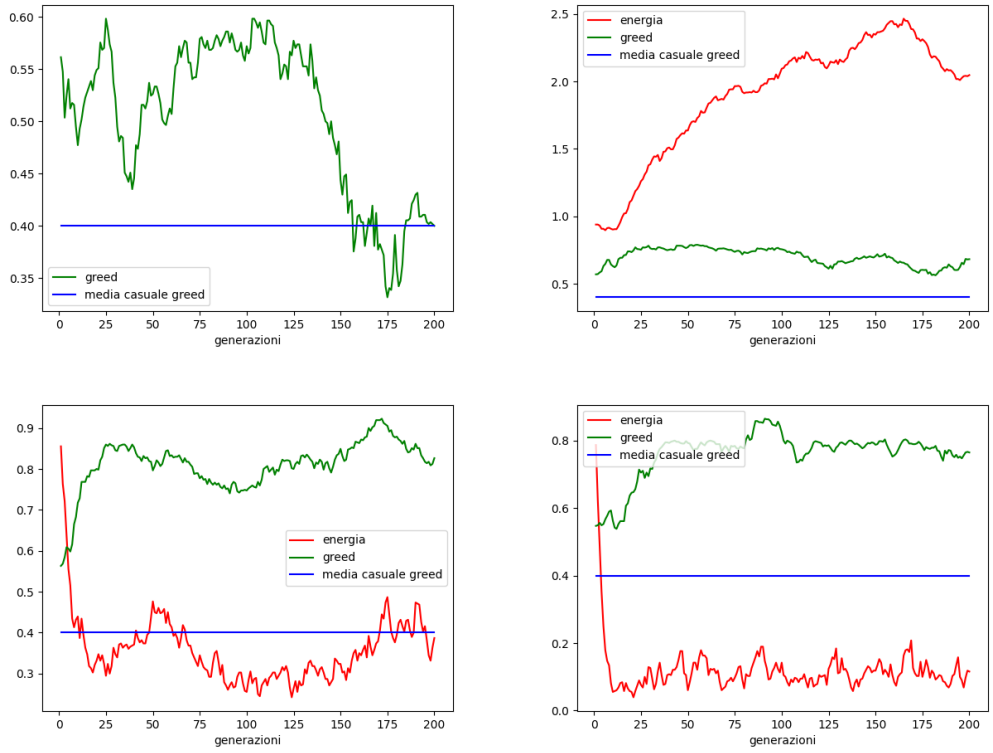
Figura 3: Grafico ottenuto facendo 4 mosse, 38 individui e 65 torte

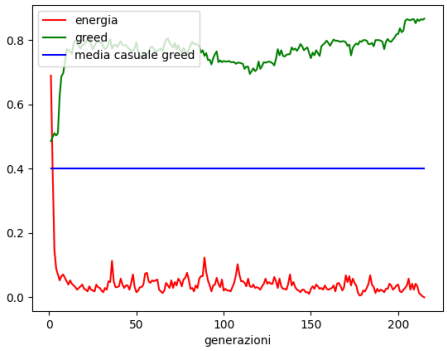
Così facendo negli individui figli si riscontra un abbassamento di apprendimento, perché tale valore si è reso meno indispensabile nella generazione precedente. Questo fenomeno di diminuzione dell'apprendimento si arresta quando di nuovo "avere *greed* elevato" torna ad essere un parametro mediamente più vantaggioso di "avere casualmente delle risorse nelle vicinanze".

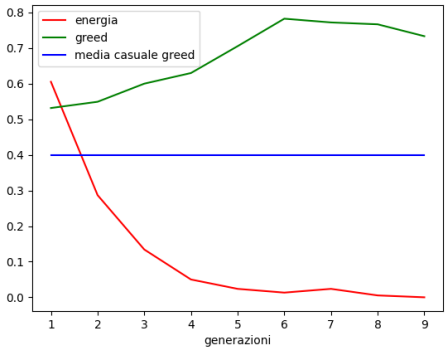
Il secondo interessante fenomeno riguarda invece una dinamica di competizione. Studiamo per un attimo i parametri in cui hanno prosperato fino alla trecentesima generazione gli individui riportati in figura 3. Per esplorare più configurazioni è stato lanciato il programma con la possibilità di fare solo 4 mosse invece che 5 come nell'esperimento del rapporto ottimale: in effetti con 4 mosse si riescono a trovare dei rapporti di convergenza migliori di quelli ottenuti nel test 3.1. Ne è un esempio il caso della figura 2, lanciato con 38 individui e 65 torte ($1.71 = \frac{65}{38} < 2.28$). Tuttavia quello che ci interessa in questo paragrafo non è migliorare il parametro ma piuttosto notare che ogni creatura ha a disposizione 1.71 torte e ha 4 mosse da fare. Siccome con ogni mossa l'energia di una creatura scende se non trova una torta e rimane invariata se la trova , deduciamo che ogni individuo, nel caso di perfetta equi-distribuzione perde circa due punti di energia dopo il movimento, quindi la media delle energia dei figli scende di *almeno* 2 punti, da cui segue un'estinzione molto veloce. Da un caso come quello di figura 3 osserviamo proprio che se *greed* si avvicina molto ad 1 le creature sono quasi tutte perfette, quindi si sottraggono le risorse a vicenda e perciò diventa più probabile il caso fatale dell'equi-distribuzione delle risorse (questo dipende anche dalla posizione delle torte, la quale potrebbe svantaggiare alcuni individui che rimangono lontani da tutto il cibo e di conseguenza salvare la popolazione attraverso il sacrificio di questi individui).

3.2 Le mosse

Il numero di mosse che le creature devono fare ad ogni generazione è un parametro importante per la capacità di sopravvivere e imparare. Si osserva banalmente che se gli individui devono fare meno mosse perdono meno energia ad ogni generazione ritardando l'estinzione. Tuttavia un numero maggiore di mosse può essere utile all'algoritmo genetico perché le creature si trovano ad affrontare, in ogni generazione, più configurazioni e questo permette al criterio di selezione basato sull'energia di discriminare meglio chi si è distinto per bravura; inoltre ridurre troppo le mosse (con rapporto risorse/ numero individui fissato) potrebbe ridurre troppo la pressione evolutiva disincentivando l'apprendimento. Verifichiamo queste ipotesi con qualche test: fissiamo un rapporto ρ di torte/individui, per esempio quello ottimale per 5 mosse cioè $\rho = \frac{87}{38} = 2.28$ calcolato in 3.1. Ora facciamo variare le mosse tra 1 e 6:







Siccome si è usato ρ ottimale per 5 mosse si ottiene che per 6 mosse non è sufficiente per la convergenza, infatti in questo test la popolazione si è estinta alla nona generazione. Infine come ci aspettavamo, per i casi banali con 1 e 2 mosse *greed* non aumenta.

3.3 Il veleno

Consideriamo ora il caso in cui sulla griglia sia presente anche del veleno; il genoma di un individuo ora contiene 81 geni e non 16 come nel caso precedente, dovendo considerare anche le disposizioni con i veleni. Oltre al già citato parametro *greed*, definiamo *fear* come la capacità di un individuo di schivare il veleno, cioè il numero di volte in cui i geni che contemplano la presenza del veleno consentono alla creatura di non passarci sopra. Per comodità normalizziamo anche *fear*. Dette X, Y le variabili aleatorie *greed* e *fear*, con un semplice calcolo si ottiene che

$$E[X] = 0.42 \quad \text{e} \quad E[Y] = 0.59$$

Questi valori ci serviranno per apprezzare, se sarà presente, un miglioramento degli individui. A priori ci aspettiamo che la convergenza sia più difficile da raggiungere per colpa della presenza dei malus di energia sparsi nella mappa e per la notevole crescita dei geni: infatti gli individui devono imparare molte configurazioni in più, molte delle quali del tipo "cosa fare in presenza di veleni". Il tentativo più naturale sarebbe quello di aumentare il numero di veleni per permettere che gli individui si trovino in uno spettro completo di situazioni, e che quindi possano essere puniti o premiati a seconda del proprio comportamento in modo da creare generazioni successive migliori. Tuttavia la presenza di veleni agisce direttamente sul parametro *energia* provocando spesso estinzioni rapide, prima che si inneschi un aumento di *fear* e *greed*. In effetti su una griglia 10x10 non si sono trovate delle configurazioni di parametri soddisfacenti e per questo motivo si è proceduto allargando la griglia ad una 20x20.
In ogni caso, procedendo appunto con un aumento della griglia e abbandonando per questo caso la ricerca del miglior rapporto tra torte e creatura, abbiamo

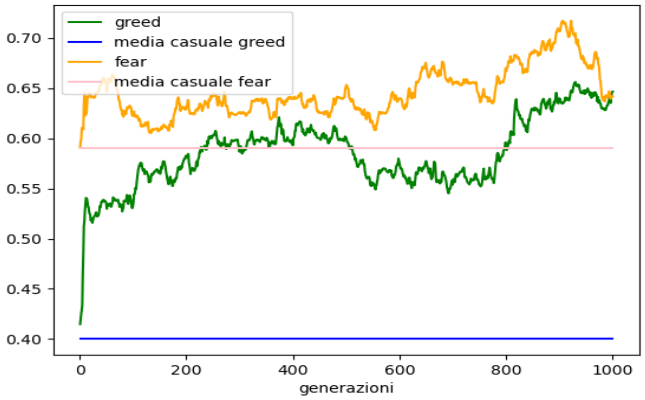


Figura 4: Grafico ottenuto facendo 5 mosse, 80 individui, 180 torte e 70 veleni

notato che si riescono a ottenere numerose situazioni soddisfacenti, ovvero in cui l'apprendimento sia evidente. Si consideri infatti la Figura 4.

Le metriche *fear* e *greed* subiscono una notevole crescita rispetto i loro valori iniziali, e tale fatto testimonia come la tecnica implementata permetta un riscontro effettivo dell'apprendimento delle creature. Notiamo qui una volta di più come però non sia possibile raggiungere un massimo assoluto di questo problema, in quanto l'aleatorietà e il gran numero di mosse della situazione impedisca ai picchi delle oscillazioni di raggiungere 1 e ivi di stazionarsi.

Citiamo ora un caso degenere che abbiamo reputato particolarmente interessante. Disponendo 250 torte, 70 veleni e 80 individui in uno spazio di 400 caselle, accade la situazione presentata nella Figura 5 : *fear* e *greed* risultano essere, con buona approssimazione, ottenibili l'uno dall'altro tramite una traslazione per un parametro "costante a tratti". In questo caso sembrerebbe dunque insensato avere due "metriche" della bontà delle creature ottenute, dal momento che la dipendenza l'una dall'altra è evidente e, sempre approssimativamente, costante a tratti.

Innanzitutto notiamo che l'esperimento ha avuto successo: le creature hanno imparato ad andare verso le torte e a schivare i veleni in modo soddisfacente, e ciò è testimoniato dal fatto che i valori si spostano notevolmente dalla media matematica, la quale è sfiorata solo da alcune oscillazioni aleatorie. Appurato dunque che tali parametri producano un risultato sensato, la natura del "problema" è data dalle relazioni numeriche tra i parametri, che in questo caso come in molti altri consentono che, in buona approssimazione, gli incrementi di *greed* e *fear* siano in relazione biunivoca. Dunque anche se queste due metriche codificano due informazioni diverse, *de facto* in numerose e simili situazioni due diverse caratteristiche finiscono per non aggiungere nulla di nuovo rispetto al comportamento della creatura.

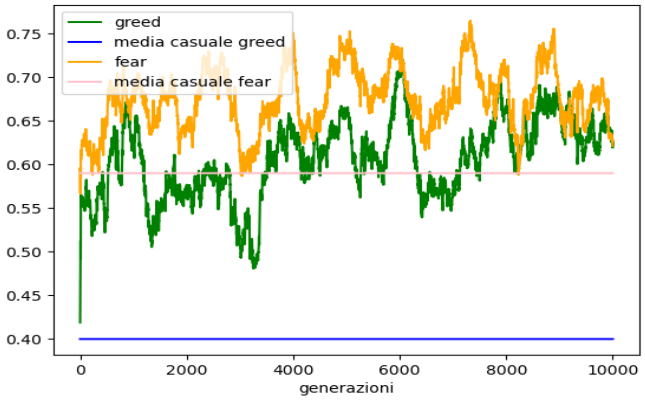


Figura 5: Caso degenere

4 Sviluppi ulteriori e limiti

4.1 8 direzioni

Si è implementata una versione del programma in cui la situazione generale rimane invariata a parte per la possibilità aggiuntiva per gli individui di "vedere" e muoversi in più direzioni, cioè le quattro già descritte con i più quelle intermedie: nord-est, nord-ovest, sud-ovest e sud-est. In questa versione le creature hanno il vantaggio di poter raggiungere più velocemente più risorse ma il loro genoma passa da una lunghezza di 2^4 a 2^8. Come per il caso del veleno, ma peggiorato notevolmente, una griglia 10 $\times$10 è troppa piccola per contenere abbastanza individui e abbastanza risorse. Incrementare gli individui è necessario per far fronte al notevole quantità di informazione genetica e aumentare le risorse è necessario per permettere agli individui di sopravvivere abbastanza a lungo perché si inneschi il processo di apprendimento.

4.2 Strategie

Si possono considerare delle strategie più fini per migliorare la capacità di sopravvivenza da parte delle popolazioni: in particolare si sono immaginate due strategie.

1. Competizione : nella sezione 3.1.1 abbiamo spiegato che in seguito ad un aumento consistente di *greed*, per rapporti di torte-individui minori del numero di mosse diventa più probabile l'estinzione. Una possibilità per abbassare ulteriormente il rapporto potrebbe essere forzare delle strategie di collaborazione in cui alcune creature vengano sacrificate e non gli si permette di accedere al cibo: quando i valori di *greed* diventano critici, questo non intaccherebbe il valore medio di apprendimento e ridurrebbe la

probabilità di estinzione. Ovviamente una possibilità più semplice sarebbe ridurre il numero di mosse, che per il momento vogliamo tuttavia tenere fissato.

2. Esplorazione: per tutto il progetto abbiamo affermato implicitamente che una creatura perfetta è quella con *greed* "massima". Di queste creature perfette ne abbiamo ottenute molte, ma notiamo che il genoma di una creatura perfetta non è un'unica configurazione tra tutte le possibili. Tra queste ce ne sono alcune migliori di altre? Per esempio, potremmo considerare come parametro aggiuntivo della bontà di una creatura se mediamente essa esplori più mappa delle altre creature perfette, cioè se essa raramente torni in posizioni in cui sia già stata. Un modo per fare questa valutazione potrebbe essere considerare quante volte la creatura presenta l'opposizione di due movimenti: se essa contiene tante volte l'istruzione "muoviti verso l'alto" quante quella "muoviti verso il basso" e analogamente con destra e sinistra, allora mediamente la creatura tenderà a tornare sui propri passi, facendo di questa una creatura leggermente peggiore di altre perché troverà meno risorse.

4.3 I limiti dell'apprendimento

Per ragioni che sono già state discusse, le situazioni con presenza di veleno hanno ottenuto risultati mediamente peggiori di quelli nella situazione semplice; in particolare i parametri *greed* e *fear* hanno avuto crescite più lente e valori massimi minori rispetto al caso senza veleno. Si è pensato di creare un test "realistico" con cui mettere alla prova alcune creature scelte. Come al solito si lancia il programma con un numero elevato di generazioni e di individui con presenza di veleno (su una griglia 20×20), si cerca di settare i parametri in modo da massimizzare la crescita di *greed* e *fear*. Di tutte le generazioni e individui abbiamo selezionato il migliore (scelto come quello con il massimo prodotto $greed \times fear$), il quale viene inserito in un circuito in cui deve avanzare il più possibile. Il circuito da completare (Figura 6) è costruito come un corridoio di spessore unitario, delimitato da veleni (croci rosse) e riempito di torte (pallini blu). L'individuo scelto smette di muoversi quando tocca una delle pareti di veleno. La natura del problema del circuito è tale per cui ogni errore commesso dalla creatura gli è fatale, e purtroppo solo poche delle creature selezionate sono abbastanza precise da muoversi lungo il circuito. Questo tentativo riconferma che il caso del veleno rende molto più difficile ottenere creature perfette. Infatti il completamento del labirinto, in realtà, non richiede che una decina di mosse "efficaci" codificate all'interno del genoma, ma nessuna creatura, seppur selezionata sulla base di *greed* e *fear* per generazioni e generazioni, è mai stata in grado di completarlo.

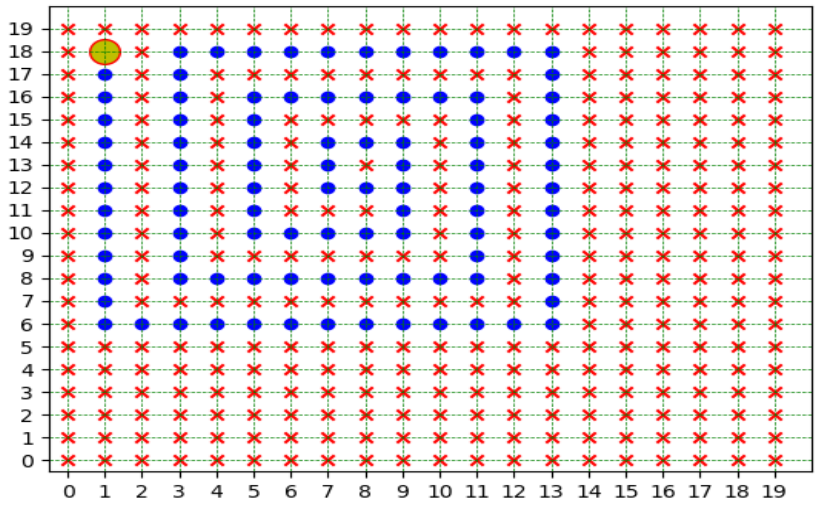


Figura 6: Test circuito