ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Corso di Laurea in Ingegneria e Scienze Informatiche

Evoluzione e Apprendimento per Rinforzo: Un Approccio Integrato per Agenti Autonomi in Contesti Videoludici

Tesi di laurea in:
Programmazione ad Oggetti

Relatore
Prof. Gianluca Aguzzi

Correlatore
Prof. Mirko Viroli

Candidato
Kimi Osti

III Sessione di Laurea
Anno Accademico 2024-2025

Sommario

I videogiochi, ambienti dinamici che richiedono di effettuare scelte in rapida successione, rappresentano un ottimo banco di prova per la valutazione dei sistemi di controllo autonomi. Particolarmente adatto in tale contesto è il Deep Reinforcement Learning, variante dell'Apprendimento per Rinforzo basato su Reti Neurali Profonde per la modellazione della conoscenza appresa. La capacità di adattamento a sistemi ad alta dimensionalità propria delle Reti Neurali Profonde viene però pagata al prezzo di un elevato costo computazionale e della necessità di definire numerosi parametri per la regolazione del processo di apprendimento. Si propone quindi un approccio combinato per il controllo di agenti di gioco autonomi, basato su Deep Reinforcement Learning per l'addestramento dei singoli individui e su un Algoritmo Genetico per ottimizzare i parametri dell'algoritmo di apprendimento. I due algoritmi sono poi direttamente integrati all'interno del mondo di gioco in cui agiscono, con l'obiettivo di misurarne il costo computazionale e valutare se possa essere fattibile integrare modelli analoghi all'interno di sistemi di gioco commerciali. Sebbene non particolarmente efficace dal punto di vita del costo computazionale—su un laptop da gioco di media fascia si misurano mediamente circa 13.25 Frame al Secondo, producendo quindi un flusso di gioco "scattoso" e poco piacevole—il modello combinato proposto si dimostra particolarmente efficace nell'ottimizzazione dei valori dei parametri e, di conseguenza, delle prestazioni dell'algoritmo di apprendimento, permettendo agli agenti di estendere il proprio tempo di vita mediamente fino a circa il 237% rispetto all'aspettativa derivata dall'analisi delle loro caratteristiche genetiche.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Fin dalla nascita dei primi calcolatori elettronici i videogiochi hanno riscontrato grande interesse, sia da parte degli sviluppatori che da parte degli utenti. Ad attirare gli utenti è sempre stata la possibilità di usare macchine primariamente impiegate lavorativamente per il proprio intrattenimento, mentre dal punto di vista degli sviluppatori i videogiochi hanno rappresentato, soprattutto nelle prime fasi, un modo per esplorare nuove tecniche di sviluppo in grado di affrontare e superare le limitazioni di memoria e capacità computazionale delle macchine a loro disposizione.

È infatti tramite la realizzazione di sistemi di gioco che sono stati proposti alcuni dei primi avanzamenti relativi alla grafica digitale—quindi ai processi di rendering e modellazione, sia bidimensionale che tridimensionale—e all'uso dell'audio per il coinvolgimento degli utenti, che hanno portato alla definizione delle interfacce e delle modalità di interazione uomo-macchina su cui si basano i sistemi moderni.

Machine Learning nei videogiochi. In tempi più recenti invece—durante i quali l'interesse degli utenti verso il mondo dei videogiochi non è sicuramente calato—l'attenzione degli sviluppatori si è spostata verso temi di più alto livello.

Se dal punto di vista strettamente tecnologico si possono infatti considerare sostanzialmente superati i problemi legati alla memoria e alle capacità computazionali dell'hardware che hanno spinto i primi avanzamenti in materia, la dinamicità degli ambienti di gioco e la naturalezza con cui si prestano allo studio del

comportamento e del coinvolgimento degli utenti hanno portato i videogiochi ad avere un ruolo sempre più centrale nella ricerca legata ai modelli comportamentali e di interazione.

In tal senso la facilità di misurazione delle prestazioni dei personaggi di gioco—spesso integrata nel motore di gioco stesso, qualora definisca un sistema di valutazione tramite punteggi—ha portato a studiare ampiamente l'uso del Machine Learning (ML) per l'addestramento di agenti autonomi di controllo dei personaggi. I mondi di gioco rappresentano infatti eccellenti spazi simulati—e di conseguenza economici—per la valutazione di sistemi di comportamento e di apprendimento autonomi.

Apprendimento per Rinforzo nei videogiochi. Particolarmente adatti a contesti di gioco sono i sistemi di controllo guidati da Reinforcement Learning o Apprendimento per Rinforzo (RL). Esso prevede infatti che gli agenti autonomi apprendano le proprie strategie interagendo direttamente con l'ambiente, senza richiedere dati di riferimento né la modellazione di una conoscenza pregressa del problema.

L'indipendenza dai dati propria del RL, che dipende invece fortemente dalla capacità del modello di valutare le proprie azioni, lo rende particolarmente adatto al dominio applicativo dei videogiochi, in cui risulta complesso definire strategie di riferimento—che spesso sono subottime o troppo poco generali anche se tratte da utenti esperti—ma estremamente semplice definire la qualità complessiva delle strategie comportamentali degli agenti—sfruttando ad esempio, se definiti, i concetti di punteggio o di vittoria previsti dalla maggior parte dei sistemi di gioco.

I recenti avanzamenti proposti nell'ambito delle Reti Neurali Profonde (DNNs) hanno poi rinnovato l'interesse verso il campo del Deep Reinforcement Learning (DRL), variante del RL che prevede appunto di sfruttare DNNs per la codifica della conoscenza appresa dal sistema di controllo, portando al raggiungimento di risultati notevoli come il superamento della prestazione di giocatori umani da parte di agenti autonomi [MKS+15].

In particolare i più recenti avanzamenti in ambito di visione artificiale hanno reso possibile la creazione di agenti autonomi basati sulla sola osservazione delle schermate di gioco [MKS+15, VDN26]. Raggiungere tale obiettivo è il primo passo

verso la creazione di agenti di gioco generali, ovvero indipendenti dal sistema in cui sono applicati ma piuttosto abbastanza flessibili da apprendere strategie efficaci idealmente per qualsiasi videogioco, basandosi solo sulle modalità di interazione comunemente usate dai giocatori umani.

Parametrizzazione dell'apprendimento. Uno dei problemi principali dei modelli di DRL è però l'elevata parametrizzazione del processo di apprendimento. Oltre a definire tutti i parametri necessari all'applicazione dell'algoritmo di RL applicato, si richiede infatti di definire i parametri regolatori dei passi di apprendimento e della struttura topologica della DNN usata per la modellazione della conoscenza acquisita.

La valutazione della qualità delle configurazioni parametriche proposte richiede però di svolgere interamente il processo di apprendimento, e successivamente delle prove di valutazione sul modello addestrato. Applicare un metodo di ricerca estensiva dei parametri richiederebbe quindi, dato l'elevato numero di parametri, l'addestramento di numerosi modelli per la sola valutazione delle configurazioni parametriche, con costi computazionali nella pratica insostenibili [MKS+15].

Risulta quindi fondamentale definire un metodo efficace ed efficiente per la ricerca educata dei parametri di apprendimento. Ciò permetterebbe infatti di ottimizzare la configurazione dei parametri riducendo il numero di modelli da addestrare, abbattendo il costo computazionale della ricerca senza intaccare però la qualità del modello.

Costo computazionale del Deep Reinforcement Learning. Un'ulteriore possibile criticità dei modelli di DRL sono gli elevati costi computazionali del processo decisionale e di apprendimento, causati dalla codifica della conoscenza all'interno delle DNNs. Esse sono infatti in grado di ridurre notevolmente le richieste di memoria di alcuni algoritmi di RL tradizionale, al prezzo però di un elevato costo computazionale a tempo di esecuzione.

Per questo motivo molto spesso i modelli di DRL sono implementati come controllori esterni all'ambiente con cui interagiscono, piuttosto che come agenti integrati al suo interno. È anzi pratica comune regolare il ritmo di esecuzione del motore di gioco per rispettare i tempi di esecuzione del modello di apprendimen-

to: fissata una frequenza di aggiornamento desiderata per l'ambiente di gioco, ad ogni passo decisionale del modello di DRL viene fatto corrispondere un passo di avanzamento del mondo di gioco della durata prestabilita, indipendentemente dal tempo realmente trascorso.

Obiettivo della ricerca. La ricerca proposta ha quindi come obiettivo quello di affrontare alcuni dei problemi noti del DRL, proponendo delle possibili soluzioni per l'addestramento efficace di agenti di gioco autonomi.

In primo luogo si vuole quindi affrontare il problema della parametrizzazione degli algoritmi di apprendimento. Per farlo si propone un modello basato sulla ricerca educata dei parametri dell'apprendimento, di cui saranno valutate efficacia e rapidità di convergenza. L'obiettivo è infatti lo sviluppo di un sistema di ottimizzazione in grado di esplorare rapidamente configurazioni parametriche vantaggiose, riducendo quindi il numero di modelli da addestrare rispetto a quelli necessari agli approcci basati su ricerca estensiva.

Dopodiché si vuole affrontare il tema della complessità computazionale dei modelli di DRL, con l'obiettivo di proporre un sistema di ottimizzazione combinato—ossia di ottimizzazione del DRL tramite ricerca educata, e di ottimizzazione delle strategie dei singoli individui tramite DRL—integrabile all'interno di un semplice motore di gioco senza deteriorare le prestazioni complessive del sistema.

Modalità di sviluppo. Per mantenere il massimo controllo sullo svolgimento del gioco e dei processi di apprendimento, si è deciso di sviluppare un semplice ambiente di gioco bidimensionale all'interno del quale svolgere le osservazioni.

I personaggi di gioco sono quindi degli agenti autonomi addestrati tramite DRL, il cui obiettivo è massimizzare il proprio tempo di sopravvivenza all'interno del mondo di gioco. La struttura del sistema di controllo degli agenti modella poi esplicitamente il concetto di attenzione: ogni agente è infatti guidato da due DNNs che lavorano in cascata, la prima per determinare l'oggetto verso cui è rivolta l'attenzione dell'agente e la seconda per determinare l'azione ideale da intraprendere dato il valore di attenzione. Così facendo si permette esplicitamente agli agenti di determinare strategie variate a seconda dell'oggetto su cui sono concentrati, aggiungendo quindi profondità ed espressività alle strategie apprese.

L'obiettivo di ridurre il costo computazionale dell'algoritmo di DRL impone però di usare DNNs semplici e a bassa dimensionalità per il controllo degli agenti. Di conseguenza anche i dati di input, ovvero le osservazioni ambientali, devono essere a bassa dimensionalità. A tale fine vengono forniti agli agenti dei dati di gioco pre-processati che ne rappresentino lo stato e la posizione, alleggerendo il processo decisionale senza sacrificare espressività del modello. Dall'altra parte il modello di apprendimento proposto risulta essere fortemente dipendente dalle caratteristiche del gioco, e quindi difficilmente adattabile alla soluzione di problemi differenti.

Infine, si integra all'interno del motore di gioco un Algoritmo Genetico (GA) per la ricerca educata dei parametri dell'apprendimento. Modellare tramite genomi i valori dei parametri di ciascun agente ed evolvere la popolazione selezionando gli individui più prestanti permette infatti di ottenere il risultato di ottimizzazione desiderato e di scartare immediatamente le configurazioni di parametri poco promettenti, riducendo quindi drasticamente la quantità di modelli da addestrare.

Risultati osservati. Il modello combinato si è dimostrato nel complesso efficace per l'addestramento degli agenti e l'ottimizzazione dei parametri dell'algoritmo di apprendimento.

Il GA applicato si è infatti dimostrato particolarmente robusto, garantendo convergenza verso configurazioni ottimali indipendentemente dalla qualità dei genomi della popolazione iniziale e a discapito di una forte instabilità osservata, dovuta in parte alla politica di mutazione applicata e in parte alla definizione di qualità su cui si basa il processo evolutivo.

Il modello di DRL ottimizzato tramite la ricerca educata dei parametri del GA ha poi permesso agli agenti di estendere il proprio tempo di vita mediamente fino al 237% del tempo di vita atteso in caso di completa inattività, sottolineando quindi la qualità generale delle strategie apprese. I risultati ottenuti dall'applicazione del modello combinato di ottimizzazione—se paragonati a quelli ottenuti applicando il solo modello di DRL, che ha permesso agli agenti di estendere il proprio tempo di vita solo del 152%—sottolineano quindi l'effetto positivo della ricerca educata dei parametri svolta dal GA, in grado di ottimizzarne i valori nell'arco di poche generazioni e quindi riducendo efficacemente il numero di modelli da addestrare.

Al contrario, la valutazione dell'efficienza del modello ha sottolineato come l'esecuzione proceda a un ritmo insufficiente per garantire un'esperienza utente positiva anche su un laptop da gioco, dotato quindi di componenti di fascia medio-alta. Una valutazione più approfondita delle prestazioni del modello di gioco lascia però margine di miglioramento, in primo luogo tramite l'ottimizzazione del processo di rendering e poi tramite eventuali migliorie introducibili nel processo di aggiornamento del mondo di gioco.

I risultati osservati non rappresentano quindi un punto di arrivo della ricerca negli ambiti affrontati, quanto piuttosto un promettente punto di partenza per eventuali ricerche future.

Struttura della Tesi. Nel Capitolo 2 saranno introdotti i concetti alla base del RL e dei GAs. Nel presentare i concetti di base di tali argomenti verranno presi come riferimento diversi lavori—di cui saranno riportati i risultati ed eventuali criticità—che hanno portato a delineare l'indirizzo di lavoro della Tesi stessa. Nel Capitolo 3 sarà descritto il progetto a supporto della Tesi, dettagliando in particolare gli algoritmi usati e le modalità di implementazione. Dopodiché nel Capitolo 4 saranno valutati i risultati ottenuti, sottolineando eventuali criticità riscontrate. Infine nel Capitolo 5 saranno presentate le conclusioni della Tesi e si proporranno in maniera più approfondita lavori futuri che possano estendere e migliorare i risultati ottenuti.

Capitolo 2

Motivazione e lavori correlati

2.1 Apprendimento per Rinforzo

Il RL è un metodo di ML basato sull'apprendimento tramite interazioni successive con un ambiente dinamico [KLM96]. Esso prevede quindi che un agente sia in grado di sperimentare e valutare autonomamente azioni diverse con l'obiettivo di trovare una strategia ottimale per la soluzione del problema che gli viene presentato; in questo senso il modello del RL è particolarmente interessante perché comparabile al modello di apprendimento del comportamento animale.

Per applicare efficacemente il RL è quindi fondamentale la definizione del concetto di "ambiente", ovvero il contesto di applicazione all'interno di cui l'agente è immerso. La modellazione dell'ambiente è fondamentale per il processo di apprendimento perché definisce le regole e i vincoli operativi dell'agente, e allo stesso momento fissa le modalità di valutazione delle azioni su cui si basa l'addestramento.

Il modello del RL si basa su tre componenti fondamentali:

- un insieme di possibili stati S che l'agente può assumere all'interno dell'ambiente;
- un insieme di azioni A che l'agente può compiere;
- un insieme di valori di ricompensa che l'agente può ricevere, solitamente identificato da un intervallo del tipo $[m, n] \in \mathbb{R}$.

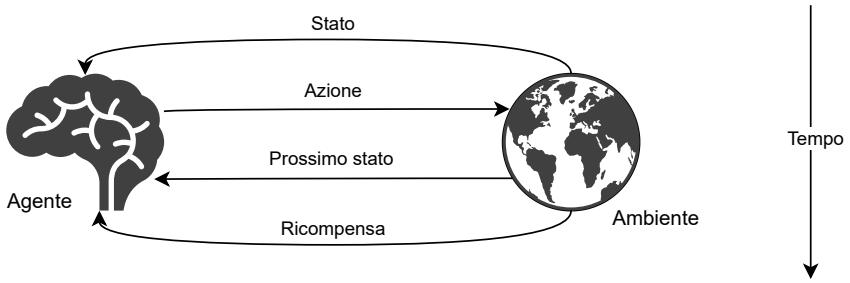

Figura 2.1: Schema rappresentativo del passo decisionale e di apprendimento dell'Apprendimento per Rinforzo. L'agente ricava una definizione del proprio stato osservando l'ambiente e la sfrutta per decidere un'azione da svolgere. Al passo successivo, dopo un certo tempo, l'ambiente restituisce una nuova rappresentazione dello stato—solitamente mutato—dell'agente corredata dalla ricompensa associata all'azione svolta, necessaria a raffinare le strategie future.

Il processo di apprendimento si basa quindi sull'interazione agente-ambiente. In particolare, in un dato istante t l'agente osserva l'ambiente e identifica il proprio stato s_t; dopodiché, tramite una strategia π dipendente dal suo algoritmo di apprendimento, decide l'azione a_t da svolgere in quel determinato istante; dall'azione svolta dipenderà la ricompensa r_t restituita dall'ambiente per l'azione svolta all'istante t. Il processo di apprendimento si basa sulla continua valutazione delle ricompense ricevute dall'ambiente, che quantificano la qualità dell'azione a all'istante t.

La qualità di una strategia π è rappresentata dalla somma di tutte le ricompense accumulate durante il periodo di interazione con l'ambiente, definito come "episodio". L'obiettivo del RL è quindi quello di apprendere la strategia ottimale π^* che permetta all'agente di massimizzare le ricompense ottenute, risolvendo di conseguenza al meglio il problema che gli viene posto. Nella valutazione di qualità delle singole azioni l'agente deve quindi essere in grado di valutare l'impatto delle stesse sulla qualità complessiva della strategia applicata, bilanciando la raccolta di ricompense istantanee e l'esecuzione di azioni lungimiranti che favoriscano l'accumulo di ricompense a lungo termine.

Indipendenza dai dati. Dal momento che l'apprendimento avviene tramite l'interazione con l'ambiente e che le ricompense ricevute per le proprie azioni sono di per loro indicative della qualità delle stesse, emerge il punto di forza principale del RL, ovvero che non richiede conoscenza strutturata del problema né dati di riferimento per completare il processo di addestramento. Quest'ultimo avviene infatti "a caldo", durante l'interazione stessa con l'ambiente. Per questo motivo il RL si adatta particolarmente bene alla soluzione di problemi in cui sia chiaramente definito l'obiettivo ma complesso definire una strategia. Due dei campi in cui viene maggiormente sfruttato il RL sono infatti la robotica [SLLN19] e i giochi, sia tradizionali [TT95, SSS+17] che elettronici [MKS+15].

La difficoltà di fornire agli agenti delle strategie ottimali di riferimento, che in altri modelli di ML comprometterebbe il processo di apprendimento, nel RL permette invece agli agenti stessi di esplorare liberamente l'ambiente in cui sono calati definendo autonomamente le proprie strategie comportamentali, talvolta sperimentando azioni "anticonvenzionali" secondo gli standard di insegnamento umano ma ugualmente efficaci. Tale comportamento è emerso ad esempio in uno studio che ha proposto un modello per l'apprendimento del Go basato completamente su RL senza nessuna forma di conoscenza pregressa [SSS+17], che si è dimostrato in grado di battere un diverso modello—basato invece su Imitation Learning, il cui processo di apprendimento richiede l'osservazione e l'emulazione delle strategie di giocatori esperti—già vincitore contro i più noti agenti di gioco autonomo del Go precedentemente proposti [SHM+16]. Durante il processo di apprendimento del modello basato unicamente su RL è infatti emerso che, oltre a tradizionali schemi tattici, esso è stato in grado di esplorare nuove mosse ugualmente efficaci [SSS+17] proprio perché svincolato da qualsiasi forma di conoscenza pregressa.

È quindi evidente come il superamento della richiesta di dati strutturati caratteristica di altri modelli di ML sia il principale punto di forza del RL, ma può rappresentare anche un importante punto di debolezza. Se la strategia iniziale π_0 non permettesse all'agente di esplorare in maniera efficace l'ambiente (ad esempio non tutti gli stati $s \in S$ vengono visitati), essa potrebbe rendere impossibile l'apprendimento della strategia ottimale π^*, in caso quest'ultima dipenda per esempio da stati o azioni che l'agente non ha sperimentato.

Exploration contro exploitation. Questo problema, non nuovo alla letteratura in materia [KLM96], prevede che si trovi un compromesso tra *exploration* ed *exploitation*. Exploration (o esplorazione) è la tendenza dell'agente a scegliere le proprie azioni con l'obiettivo di approfondire la propria conoscenza dell'ambiente, indipendentemente dalla qualità prevista. L'obiettivo di un'azione a_t di esplorazione è quindi la sperimentazione di una nuova strategia piuttosto che lo sfruttamento delle conoscenze apprese. Al contrario Exploitation (o sfruttamento) è la tendenza dell'agente a scegliere le proprie azioni sfruttando la conoscenza pregressa. L'obiettivo di un'azione a_t di sfruttamento delle conoscenze pregresse è quindi quello di massimizzare la ricompensa istantanea r_t indipendentemente dal livello di conoscenza che si ha dell'ambiente.

Appare però evidente che una strategia di solo sfruttamento delle conoscenze pregresse rischierebbe di non raggiungere mai la strategia ottimale π^* perché favorirebbe la ripetizione cieca di azioni già note rispetto all'esplorazione dell'ambiente, che permetterebbe invece al modello di svolgere valutazioni più generali e quindi efficaci. Dall'altra parte una strategia di sola esplorazione vanificherebbe completamente il processo di apprendimento, perché prediligerebbe ad ogni passo azioni di esplorazione senza mai sfruttare le conoscenze apprese osservando le ricompense ricevute. La qualità degli algoritmi di RL dipende quindi in maniera importante dalla capacità di esplorare l'ambiente pur garantendo a lungo termine la scelta di azioni redditizie.

Politica ε-greedy. Una soluzione nota al problema, proposta in diversi studi fra cui anche [MKS+15], è l'applicazione della politica di scelta delle azioni ε-greedy. Essa prevede di stabilire un parametro $\varepsilon \in [0, 1]$ che determina il bilanciamento applicato nel compromesso fra esplorazione e sfruttamento delle conoscenze pregresse. La politica prevede infatti che si scelga un'azione casuale di esplorazione con probabilità ε, e che altrimenti si scelga l'azione reputata più redditizia—da cui il nome "greedy"—con probabilità $1 - \varepsilon$.

A seconda dell'implementazione dell'algoritmo, il valore di ε può essere fisso o variabile nel tempo: un metodo solitamente utilizzato è quello di porre ε tendente a 1 nelle fasi iniziali per prediligere l'esplorazione, e farlo decadere fino a valori tendenti a 0 nelle fasi finali dell'apprendimento per sfruttare la conoscenza appresa.

Si rivela comunque efficace—specialmente in ambienti molto dinamici e variabili come possono essere i videogiochi [MKS+15]—mantenere anche nelle fasi finali una quota di azioni di esplorazione per spingere l'agente a sperimentare nuove strategie potenzialmente più adatte alle mutate condizioni dell'ambiente [KLM96].

Dipendenza stato-azione-ricompensa. Un altro problema da affrontare nel processo di apprendimento del RL è quello della dipendenza fra stati e azioni dell'agente. Questo problema, ricorrente nella gran parte dei domini applicativi, emerge in particolare negli ambienti dinamici in cui il RL è prediletto rispetto ad altri paradigmi di apprendimento. In primo luogo, è evidente come una stessa azione a possa avere risultati differenti a seconda dello stato s dell'agente al momento in cui viene eseguita. Prendendo ad esempio come dominio il gioco Space Invaders, studiato da [MKS+15], l'azione "sparare" è valutata positivamente se colpisce una nave nemica, mentre risulta dannosa per il giocatore se colpisce una delle barriere protettive, e la differenza di esito dipende primariamente dalla posizione del giocatore, ovvero una componente del suo stato. Per modellare questa dipendenza si propone la seguente definizione modificata di ricompensa:

$$R : S \times A \longrightarrow \mathbb{R} \tag{2.1}$$

Dipendenza stato-azione-stato. Nella direzione opposta, è vero anche che le azioni svolte influenzano lo stato dell'agente. Dato uno stato istantaneo s_t, l'azione a_t svolta dall'agente può influenzare più o meno direttamente la transizione verso lo stato s_{t+1}, e tale influenza può essere certa o probabilistica. Sempre riprendendo l'esempio di Space Invaders studiato in [MKS+15], l'azione "sparare" non modificherà sicuramente la posizione dell'agente, ma a seconda di altri elementi ambientali potrebbe causare variazioni dello stato, come ad esempio la distruzione di una nave nemica se il colpo va a segno: a seconda della definizione di stato (la sola posizione del giocatore, o anche altri elementi dell'ambiente) e dell'azione intrapresa la dipendenza diventa quindi certa o probabilistica. Un possibile modello per questa dipendenza è offerto dai Processi Decisionali di Markov (MDPs) [KLM96], che si basano su alcuni elementi fondamentali:

- un insieme degli stati S;

- un insieme delle azioni A;
- una funzione ricompensa come definita dall'Equazione (2.1);
- una funzione $T : S \times A \times S \longrightarrow [0, 1]$ che stabilisce, dati uno stato di partenza s_t, un'azione a_t e uno stato di arrivo s_{t+1} la probabilità che l'azione a_t provochi la transizione di stato $s_t \longrightarrow s_{t+1}$.

La modellazione tramite MDPs richiede però due assunzioni fondamentali. In primo luogo si richiede che l'ambiente sia completamente osservabile, ovvero che l'agente abbia piena cognizione dell'ambiente in cui agisce. Tale assunzione è necessaria a garantire che osservazioni uguali coincidano alla rappresentazione dello stesso stato, condizione non verificata se parti dell'ambiente sono oscurate alla visione dell'agente—la stessa osservazione potrebbe infatti coincidere a stati diversi a seconda della configurazione della zona di ambiente invisibile all'agente.

Inoltre si richiede che lo storico degli stati dell'agente non influenzi la funzione di probabilità delle transizioni, rendendo i MDPs utili alla modellazione solo di ambienti *memory-less*. Ad esempio i giochi da tavolo, a pari configurazione di tabellone, offrono le stesse opportunità di azione all'agente indipendentemente dallo storico delle azioni precedenti, e ciascuna azione determina in maniera diretta lo stato successivo [TT95, SSS+17]. Anche i videogiochi, a parità di stato e azione svolta, sono progettati per produrre sempre lo stesso effetto e sono quindi in tal senso modellabili tramite MDPs [MKS+15].

Ricompense a lungo termine. Grazie alla modellazione della dipendenza fra azioni e transizioni di stato che offrono i MDPs è possibile introdurre il concetto di ricompensa a lungo termine [KLM96]. Quantificare efficacemente le ricompense a lungo termine permette al modello di apprendimento di fare scelte più lungimiranti, talvolta sacrificando ricompense istantanee se necessario a massimizzare la ricompensa complessiva. In questo modo, gli agenti prediligono l'applicazione di una buona strategia π rispetto a una sequenza di azioni a_t localmente greedy.

Per quantificare l'impatto delle ricompense a lungo termine si introduce quindi la funzione ricorsiva di qualità $V : S \times A \longrightarrow \mathbb{R}$. Essa introduce il parametro γ, detto discount factor, e denota come a^* l'azione prevista dalla strategia applicata π per lo stato s' [KLM96].

$$V(s,a) = R(s,a) + \gamma \sum_{s' \in S} [T(s,a,s')V(s',a^*)] \tag{2.2}$$

Dal momento che l'equazione è definita in maniera ricorsiva e pertanto espandibile in una sommatoria infinita, si richiede che il valore di γ sia minore di 1 per garantire convergenza a un valore finito.

L'Equazione (2.2) valuta contemporaneamente le probabilità di transizione verso ciascuno stato successivo, la singola previsione di ricompensa e le scelte future dettate dalla strategia corrente. In questo modo garantisce che la scelta greedy dell'azione a_t sia ottimale anche per la strategia a lungo termine, tenendo conto delle conseguenze che la scelta istantanea ha sui comportamenti futuri dell'agente. La natura ricorsiva della funzione di qualità, utile a garantirne la precisione e a far coincidere le scelte localmente e complessivamente ottime, la rende però incalcolabile precisamente perché espandibile all'infinito. Una buona approssimazione si può comunque ottenere sostituendo all'interno dell'Equazione (2.2) un singolo valore finito al posto dell'espressione ricorsiva $V(s', a^*)$, ma per farlo sarebbe necessario mantenere in memoria i valori di qualità per tutte le coppie stato-azione ammissibili.

Q-learning. Un algoritmo di RL basato sulla modellazione esplicita delle dipendenze fra stati e azioni è il Q-learning. Esso si basa infatti sull'uso di una tabella a doppia entrata—su un asse gli stati, sull'altro le azioni—detta Q-table e denotata come Q, che modella esplicitamente la dipendenza introdotta dall'Equazione (2.1) associando ad ogni coppia stato-azione un valore di qualità. Il processo decisionale del Q-learning prevede quindi, dato uno stato s_t, di scegliere l'azione a giudicata più redditizia. La strategia π è quindi definita come segue:

$$\pi = \max_a Q(s_t, a) \tag{2.3}$$

La definizione del passo decisionale del Q-learning fa però emergere la prima criticità dell'algoritmo: la strategia applicata è composta interamente da scelte localmente ottimali, approccio che potrebbe apparentemente limitare la lungimiranza del modello.

Questo problema viene però affrontato e risolto dalla formulazione del passo di apprendimento, che consiste in un aggiornamento dei valori della Q-table basato su una valutazione di qualità delle azioni analoga a quella proposta dall'Equazione (2.2) in cui si modellano esplicitamente la dipendenza fra l'azione intrapresa e la transizione di stato osservata e il concetto di ricompensa a lungo termine.

Il passo di apprendimento per l'istante t—che viene svolto all'istante $t+1$, quando sono già noti il valore di ricompensa $r_t = R(s_t, a_t)$ e il nuovo stato s_{t+1}—viene espresso come:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{2.4}$$

Tale formulazione prevede di introdurre due parametri regolatori dell'apprendimento: il learning rate α, che regola l'impatto che le singole osservazioni hanno sulla conoscenza precedentemente acquisita, e il discount factor γ, che regola il peso che le ricompense future hanno rispetto a quella istantanea.

La formulazione ricorsiva proposta nell'Equazione (2.4)—$Q(s_t, a)$ dipende direttamente da $Q(s_{t+1}, a)$, che a sua volta dipende da $Q(s_{t+2,a})$ e così via—permette quindi di superare il problema delle scelte localmente greedy introdotto dalla definizione del passo decisionale, ma rende particolarmente importante la scelta del valore del parametro γ di bilanciamento delle ricompense a lungo e a breve termine.

La definizione del passo di apprendimento del Q-learning è quindi efficace nella modellazione delle ricompense a lungo termine, ma fa emergere quello che forse è il problema principale dell'algoritmo: l'aggiornamento dei valori di qualità avviene solo per le coppie stato-azione effettivamente esplorate, mentre le restanti coppie ammissibili restano ai valori iniziali, che sono solitamente posti a zero o generati casualmente per evitare condizionamenti.

L'algoritmo richiede quindi di modellare esplicitamente il compromesso fra esplorazione e sfruttamento delle conoscenze pregresse. Un metodo efficace può essere quello di adottare una politica ε-greedy con tasso di esplorazione variabile nel tempo e limitato all'estremo inferiore, al fine di mantenere una quota di azioni di esplorazione anche nelle fasi avanzate dell'apprendimento. Si definiscono quindi il tasso di esplorazione iniziale ε_0, il tasso di decadimento $\lambda_\varepsilon \in (0, 1)$ applicato come coefficiente moltiplicativo a ε ad ogni passo decisionale e il tasso di esplorazione

minimo ε_{min} e si riassume la definizione del Q-learning con politica di esplorazione ε-greedy nell'Algoritmo 1.

Algoritmo 1 Algoritmo Q-learning con politica ε-greedy

for all $s \in S$, $a \in A$ **do**
  $Q(s, a) \leftarrow 0$
end for
$\varepsilon \leftarrow \varepsilon_0$
$s_t \leftarrow s_0$
Define a_t, r_t, s_{t+1}
repeat
  With probability ε: $a_t \leftarrow$ random $a \in A$
  With probability $1 - \varepsilon$: $a_t \leftarrow \max_a Q(s_t, a)$
  if $\varepsilon > \varepsilon_{min}$ **then**
    $\varepsilon \leftarrow \lambda_\varepsilon \varepsilon$
  end if
  Perform a_t
  Observe s_{t+1}, r_t
  $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$
  $s_t \leftarrow s_{t+1}$
until end

Abbattimento della dimensionalità. La formulazione del Q-learning proposta, così come la gran parte degli algoritmi di RL, sebbene in grado di risolvere i problemi legati alle dipendenze fra stati e azioni e alla modellazione di strategie lungimiranti lascia aperto un importante problema di dimensionalità. La rappresentazione esaustiva della conoscenza modellata tramite Q-table richiede infatti uno spazio di memoria direttamente proporzionale alla dimensione dello spazio $S \times A$ delle coppie stato-azione ammissibili, rendendo tale approccio di fatto insostenibile per la gran parte dei domini reali.

Un dominio reale in cui emergono forti problemi di dimensionalità è ad esempio quello del backgammon, che si stima abbia 10^{20} stati ammissibili distinti [TT95]. Data l'impossibilità di modellare direttamente una mole di stati così grande, e data anche la difficoltà di simulare l'andamento della partita in tempo reale per valutare la qualità di una mossa—il lancio dei dadi rende il backgammon non deterministico, aumentando notevolmente il numero di possibili scenari da considerare rispetto

ad altri giochi deterministici come gli scacchi o la dama—si sono proposti degli approcci alternativi per la previsione del valore di qualità delle azioni di gioco. Nel 1988, alle Computer Olympiad viene presentato Neurogammon [Tes90], campione imbattuto contro altri agenti autonomi nella categoria del backgammon. Esso è basato su DNNs in grado di dedurre le probabilità di vittoria dati la configurazione del tabellone e altri dati di input pre-processati estraibili in tempo reale. Il suo addestramento è basato su una raccolta di dati prodotti da esperti del gioco, e lo ha portato ad essere il primo modello di ML a vincere una competizione per agenti di gioco autonomi.

Pochi anni dopo venne invece proposto TD-Gammon [TT95], modello interamente basato su RL e addestrato unicamente tramite sessioni di gioco contro se stesso, senza conoscenze pregresse. Quest'ultimo, anch'esso basato su DNNs addestrate a dedurre la probabilità di vittoria prevista, fu in grado di raggiungere risultati comparabili a quelli di Neurogammon sfruttando come input la sola codifica della configurazione del tabellone. Una seconda versione, il cui input fu esteso per supportare gli stessi dati pre-processati di Neurogammon (ma comunque addestrata senza conoscenza pregressa) fu in grado di migliorare ulteriormente le performance arrivando a un deficit di circa 0.2 punti a partita contro alcuni dei migliori giocatori umani dell'epoca.

Analizzando il comportamento di TD-Gammon, analisti esperti e giocatori professionisti notarono che aveva appreso, essendo "liberato" dalla conoscenza pregressa, delle strategie anticonvenzionali che si sono rivelate migliori di quelle reputate standard, portando alla rivisitazione di alcune tattiche di gioco consolidate [TT95].

Parte del successo di TD-Gammon, a detta dello stesso autore, è inoltre da attribuire alla casualità intrinseca del backgammon, che richiede il lancio di dadi. Grazie a ciò il modello, pur giocando sempre contro se stesso, è stato obbligato ad esplorare zone dello spazio degli stati che altrimenti non avrebbe probabilmente esplorato: in altri giochi più deterministici come gli scacchi il modello avrebbe infatti verosimilmente appreso nelle prime fasi una strategia basilare e poco generica, e si sarebbe auto-alimentato replicandola nelle partite contro se stesso senza riuscire a raggiungere la strategia ottimale [TT95]. Questa osservazione è in linea con la necessità precedentemente introdotta di stabilire un compromesso tra esplorazione e sfruttamento della conoscenza appresa, che in questo caso è stato

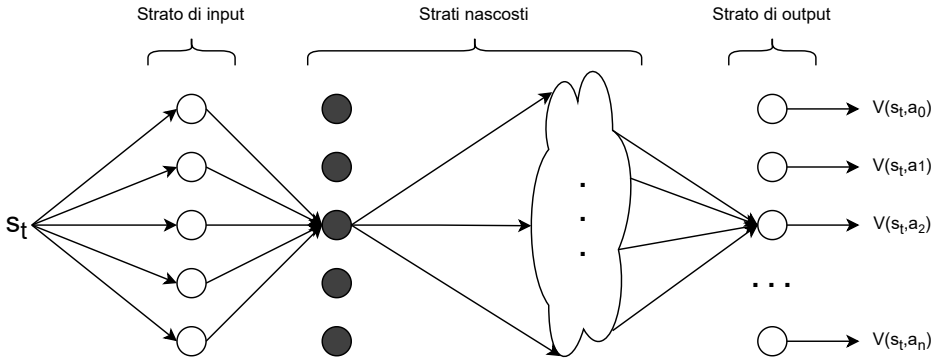

Figura 2.2: Schema rappresentativo del funzionamento di una Rete Neurale Profonda per il Deep Reinforcement Learning. Lo stato s_t viene riformulato in forma vettoriale ed espresso come stimolo di input per i neuroni del primo strato della rete. Ogni neurone propaga il proprio valore di attivazione a tutti i neuroni del livello successivo, e le trasformazioni svolte dagli strati nascosti vengono recepite dallo strato di output. Esso, composto da un neurone per ciascuna delle n azioni possibili, produce stimoli di attivazione corrispondenti ai valori di qualità attesi.

imposto dal dominio applicativo ma che in altri contesti deve essere esplicitamente modellato.

Deep Reinforcement Learning. Se quello di TD-Gammon ha rappresentato un grande avanzamento per il RL ma non è stato replicato in altri domini nel breve termine [KLM96], l'avanzamento della ricerca sulle DNNs ha recentemente rinnovato l'interesse verso il campo del DRL [AHSB18, WWL+24], alimentato dal raggiungimento di risultati notevoli come [MKS+15]. Gli algoritmi di DRL affrontano principalmente il problema della dimensionalità degli stati: sfruttando le DNNs per l'approssimazione funzionale dei valori di qualità delle azioni o delle strategie è infatti possibile scalare teoricamente a problemi arbitrariamente grandi, a patto di definire modelli di DNNs adeguati alla loro approssimazione.

Uno dei maggiori avanzamenti recentemente proposti in merito è stato la realizzazione di un modello di DRL in grado di battere dei tester umani nella gran parte della libreria dei giochi Atari 2600 usando i soli pixel delle schermate di gioco come dati di input e il punteggio come metrica di ricompensa [MKS+15]. L'algoritmo proposto nello studio viene chiamato Deep Q-Learning e si basa sul-

l'uso di una DNN per l'approssimazione funzionale dei valori di qualità per ogni coppia stato-azione come previsto dal Q-Learning. Il modello—che prende come input le schermate di gioco, pre-processate solo applicando una riduzione della dimensionalità—prevede che i primi strati convoluzionali della DNN estraggano dalle immagini le informazioni rilevanti alla definizione dello stato dell'agente, poi usato dagli ultimi strati per prevedere i valori di qualità attesi per ogni azione possibile. L'algoritmo sfrutta poi la politica ε-greedy per bilanciare esplorazione e sfruttamento delle conoscenze: posto un valore iniziale $\varepsilon_0 = 1$ e stabilito un numero fisso di azioni di pura esplorazione, si lascia decadere gradualmente il valore di ε durante l'apprendimento fino a raggiungere valori prossimi allo 0. Viene comunque sempre mantenuto $\varepsilon > 0$ per lasciare spazio a una quota di azioni di esplorazione anche nelle fasi più avanzate dell'apprendimento. Tramite questa politica viene modellato esplicitamente il compromesso tra esplorazione e sfruttamento delle conoscenze, superando i limiti sottolineati durante lo sviluppo del modello TD-Gammon [TT95].

Parametri dell'apprendimento. Durante il processo di apprendimento—nel caso del RL, e ancora di più nel caso del DRL in cui entrano in gioco anche gli iperparametri tipici delle DNNs—devono essere fissati i valori di alcuni parametri fondamentali che possono influenzare l'efficacia dell'algoritmo. In particolare si sottolineano:

- exploration rate ε, che determina il bilanciamento tra esplorazione e sfruttamento della conoscenza. In particolare, da esso derivano 3 parametri: exploration rate iniziale ε_0, exploration rate minimo ε_{min} e tasso di decadimento λ_ε;
- discount factor γ, che determina quanto peso viene dato alle azioni future nel calcolo del valore di qualità della singola azione. Porre $\gamma = 0$ significa prediligere azioni localmente ottime senza lungimiranza, mentre al contrario $\gamma \simeq 1$ significa prediligere azioni lungimiranti a ricompense istantanee;
- learning rate α, proprio dei modelli di apprendimento "a step"—quali alcuni algoritmi di RL tradizionale, come il Q-Learning, e i metodi di addestramento delle DNN—e fondamentale per garantire il raggiungimento di una

soluzione ottima. Valori troppo piccoli di α, oltre a rallentare il processo di apprendimento nel suo complesso, potrebbero portare a ignorare errori piccoli ma significativi, condizionando la convergenza verso soluzioni subottime. Al contrario valori troppo elevati, seppur in grado di velocizzare il processo di apprendimento, potrebbero impedire il mantenimento della soluzione ottima o complicarne l'esplorazione, rendendo il modello più suscettibile alle singole valutazioni di errore;

- topologia della rete, iperparametro fondamentale di qualsiasi algoritmo basato su DNNs che determina il livello di generalità del modello e di conseguenza la sua adeguatezza ad esprimere la complessità del problema;

- altri parametri propri dell'algoritmo scelto e della sua implementazione: nel caso del modello per i giochi Atari, che lavora con Deep Q-Learning, sono per esempio la dimensione dei batch per il singolo passo di apprendimento della DNN, la frequenza dei passi di apprendimento rispetto ai passi decisionali e altri parametri strettamente legati alle modalità di aggiornamento dei pesi della DNN [MKS+15].

Risulta quindi evidente come l'ottimizzazione dei parametri dell'algoritmo scelto ne possa migliorare o compromettere l'efficacia. Dall'altra parte però, gli stessi ricercatori che hanno implementato l'apprendimento per i giochi Atari si sono affidati a una "ricerca informale" degli iperparametri effettuata su pochi giochi scelti a priori—senza garanzie di efficacia di tale approccio—perché una ricerca estensiva degli stessi sarebbe stata computazionalmente insostenibile [MKS+15]. Inoltre, lo stesso set di iperparametri ottimizzato su quei pochi giochi scelti secondo criteri qualitativi è stato applicato a tutti i modelli di apprendimento, indipendentemente dal gioco affrontato: non sarebbe quindi impensabile provare a migliorare le prestazioni del modello senza proporre un nuovo algoritmo, ma semplicemente ottimizzando le scelte degli iperparametri per ogni singolo dominio applicativo.

2.2 Algoritmi Genetici

Un problema aperto è la definizione di modalità computazionalmente sostenibili per la ricerca e l'ottimizzazione degli iperparametri dei modelli di apprendimento. Una soluzione è sicuramente offerta dai GA, che associati al DRL possono offrire un metodo di ricerca educata dei parametri guidato dalla valutazione dei risultati dei modelli di DRL stessi [SLLN19]. Uno dei punti di forza della ricerca tramite GAs rispetto a una tradizionale ricerca estensiva è che il numero di modelli da addestrare contemporaneamente diminuisce notevolmente, dal momento che si riescono a scartare immediatamente le configurazioni poco promettenti.

I GAs sono infatti una classe di algoritmi che, imitando il processo di trasmissione genetica della riproduzione biologica, mira ad evolvere una popolazione mantenendo e ottimizzando i tratti degli individui più prestanti. Si basano su alcuni concetti fondamentali:

- l'insieme di individui di cui si vogliono ottimizzare le caratteristiche è detto popolazione. Data una popolazione iniziale, da essa deriveranno nuove generazioni di individui a cui verranno trasmessi solo i caratteri dominanti;
- ciascun individuo ha un genoma, ovvero una collezione di geni che possono avere valori binari, discreti o continui, che codifica tutte le sue caratteristiche rilevanti;
- si definisce una funzione fitness che associa a ogni individuo un valore numerico che ne rappresenta la "qualità". Gli individui con valore di fitness più elevato, secondo il modello evolutivo darwiniano, saranno più propensi a trasmettere il proprio genoma alle generazioni successive. Dalla definizione della funzione fitness (che deve essere in grado di modellare esattamente le metriche di qualità desiderate) dipende quindi in larga parte la qualità dell'algoritmo proposto, dal momento che è essa a guidare la selezione dei genomi ai fini evolutivi.

Il momento cardine in cui entra in gioco il GA è la generazione di un nuovo individuo, qualunque sia il motivo che l'ha innescata. Generalmente un GA prevede un processo a tre fasi per determinare il genoma dell'individuo: selezione, ricombinazione e mutazione.

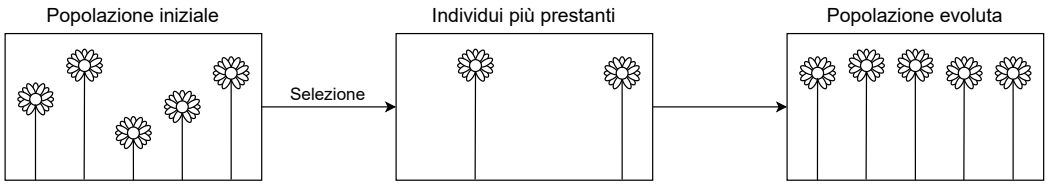

Figura 2.3: Schema generale di funzionamento degli Algoritmi Genetici. Data una popolazione iniziale, solitamente generata con caratteristiche casuali o frutto di un'ottimizzazione preliminare, si applica una selezione basata sui valori di fitness—nel caso esemplificato la lunghezza dello stelo—che permette di generare una popolazione evoluta che mantenga le sole caratteristiche degli individui più prestanti.

La selezione prevede di individuare all'interno della popolazione esistente i genitori per il nuovo individuo, e per farlo si affida in maniera più o meno diretta al valore di fitness degli individui della popolazione: metodi che selezionano in ogni caso gli individui più prestanti convergono rapidamente ma sfavoriscono l'esplorazione di nuove combinazioni potenzialmente vantaggiose, mentre al contrario metodi che dipendano in maniera troppo blanda dai valori di fitness rischiano di fallire nella selezione degli individui con genomi più favorevoli, rallentando quindi il processo evolutivo della popolazione.

La ricombinazione, come l'omonimo processo biologico, prevede di mescolare i genomi dei genitori per generare una nuova combinazione di geni da assegnare al figlio. Ai fini della ricombinazione si può lavorare a livello di sequenze di geni—applicando quindi uno o più punti di taglio, e trasmettendo intere sezioni di genoma in maniera unitaria al nuovo individuo—oppure di singolo gene—scegliendo quindi puntualmente da quale genitore ereditare ogni gene, senza un numero di punti di taglio prestabiliti—ma non si agisce mai al di sotto di tale granularità. In questa fase si garantisce quindi che ogni gene del nuovo individuo sia identico al corrispondente del genitore da cui è stato ereditato.

La mutazione poi, processo fondamentale per garantire l'esplorazione dello spazio dei genomi ammissibili, prevede di modificare puntualmente singoli geni del figlio per proporre un valore differente rispetto a quello ereditato dai genitori. Si devono quindi definire il tasso di mutazione—ovvero la probabilità di colpi-

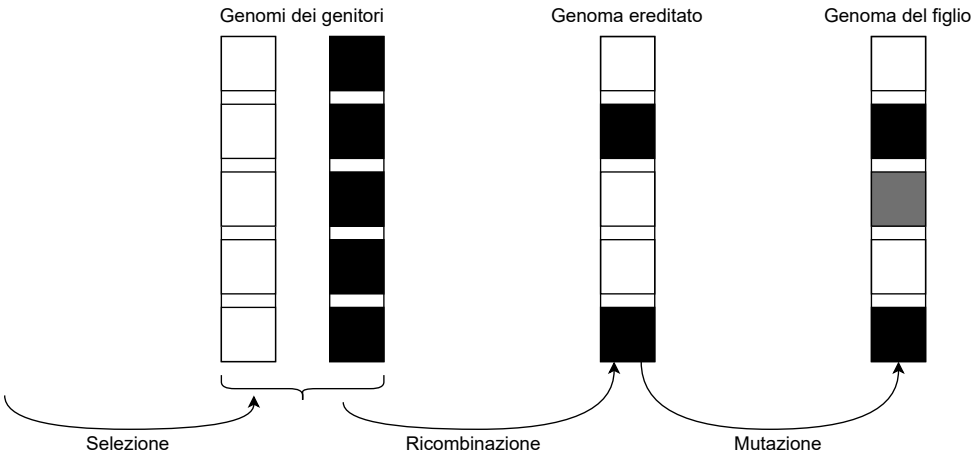

Figura 2.4: Schema di applicazione del singolo passo evolutivo dell'Algoritmo Genetico, necessario per determinare le caratteristiche del nuovo individuo. Tramite selezione sono determinati i genitori del nuovo individuo, dalla ricombinazione dei due genomi dei genitori è determinato il genoma ereditato, che può poi essere soggetto a mutazioni puntuali dei singoli geni per determinare le caratteristiche del nuovo individuo.

re ciascun gene, parametro che ha una funzione simile al tasso di esplorazione ε per il RL—e un metodo per determinare, quando applicate, l'aggressività delle mutazioni.

Ottimizzazione dei parametri con GA. Recenti studi hanno esplorato la possibilità di applicare GAs all'ottimizzazione dei parametri degli algoritmi di DRL, dimostrando come la ricerca educata possa essere una soluzione sostenibile a tale problema [SLLN19, BPK25]. In particolare lo studio [BPK25] paragona diversi metodi di selezione, ricombinazione e mutazione applicati allo stesso problema commentandone l'efficacia e la rapidità di esecuzione su una popolazione di dimensione prefissata, ponendo quindi l'accento anche sulla valutazione del costo computazionale del processo evolutivo.

2.3 Creatures: vita artificiale come videogioco

Nel 1996 CyberLife, compagnia inglese di sviluppo di videogiochi, rilasciò sul mercato Creatures, un simulatore di vita artificiale per PC, PlayStation e Game Boy Advance. Integrati direttamente all'interno del gioco sono presenti agenti autonomi di ML controllati da reti neurali, evoluti lungo le generazioni tramite l'applicazione di un GA. Tale esempio, oltre a rappresentare un unicum mai più replicato in sistemi di gioco commerciali, risulta particolarmente rilevante perché in grado di affrontare efficacemente i limiti di capacità computazionale degli hardware dell'epoca.

Il gioco. La dinamica di gioco prevede che all'utente venga fornito un set iniziale di uova di "Norns" (le creature del gioco) con un genoma unico dipendente dalla copia fisica del gioco di cui si è in possesso. Gli individui nati da tali uova sono la popolazione iniziale per il mondo dell'utente, e l'obiettivo è quello di mantenere ed estendere la popolazione facendo in modo che gli agenti interagiscano fra loro per riprodursi. Vista la dinamica di riproduzione, il GA di Creatures non prevede un sistema di selezione dei genitori, che vengono invece determinati direttamente dalla loro strategia di comportamento—la capacità di apprendere le azioni necessarie alla riproduzione fa quindi già da metodo di selezione e valutazione dei genitori, senza dover definire esplicitamente una funzione fitness. Sono invece normalmente applicati un sistema di ricombinazione e mutazione dei genomi per determinare le caratteristiche dei nuovi individui [GC98]. Il genoma delle creature non determina poi soltanto i parametri del loro apprendimento—come invece solitamente avviene in altri studi più recenti [SLLN19], che sfruttano gli agenti di DRL come sistemi di controllo esterni—ma codificano anche le caratteristiche fisiologiche degli agenti, che ne possono influenzare le modalità di interazione con l'ambiente.

Il cervello degli agenti. Le reti neurali di controllo proposte in Creatures non sono DNNs completamente connesse—come proposto nella gran parte degli altri studi, anche contemporanei al rilascio del gioco [MKS+15, Tes90, TT95]—ma piuttosto hanno una struttura a lobi, necessità dettata dai limiti di memoria e capacità computazionale dei sistemi supportati. Ogni rete neurale ha infatti una topologia

Figura 2.5: Modellino del mondo di gioco di Creatures, custodito al Centre for Computing History di Cambridge, Regno Unito[1]. Durante lo sviluppo del gioco, per superare i costi e le problematiche legate alla modellazione 3D digitale, i modelli del mondo e dei personaggi vennero realizzati fisicamente, fotografati e digitalizzati per essere usati come assets di gioco. Ciò ha contribuito a determinare lo stile caratteristico delle grafiche di Creatures.

predefinita, al cui interno sono attivate un numero limitato di connessioni [GC98]: la conoscenza appresa è quindi codificata primariamente dalle connessioni attive piuttosto che dai pesi delle stesse, con un risultato di fatto analogo a quello ottenuto con una DNN in cui la maggior parte dei pesi sia uguale a zero.

I lobi dei cervelli di Creatures sono quindi dei raggruppamenti di neuroni che svolgono una funzione comune. Ci sono perciò dei lobi dedicati alla lettura delle osservazioni del mondo che traducono lo stato degli agenti in stimoli di attivazione, dei lobi di pensiero che permettono di stabilire delle connessioni mediate fra neuroni di altri lobi, e dei lobi di output che traducono gli stimoli ricevuti dagli altri neuroni in stimoli di attivazione con cui determinare le azioni da svolgere. Riprendendo quindi il modello delle reti neurali di Creatures, e rapportandolo a sistemi moderni in cui si può supporre di poter implementare DNNs completamente connesse senza particolari problemi computazionali, si possono reinterpretare i lobi come singoli strati di un'unica DNN dove i lobi dedicati alla lettura degli sti-

[1] `https://www.computinghistory.org.uk/det/42694/Creatures-Development-Model/`

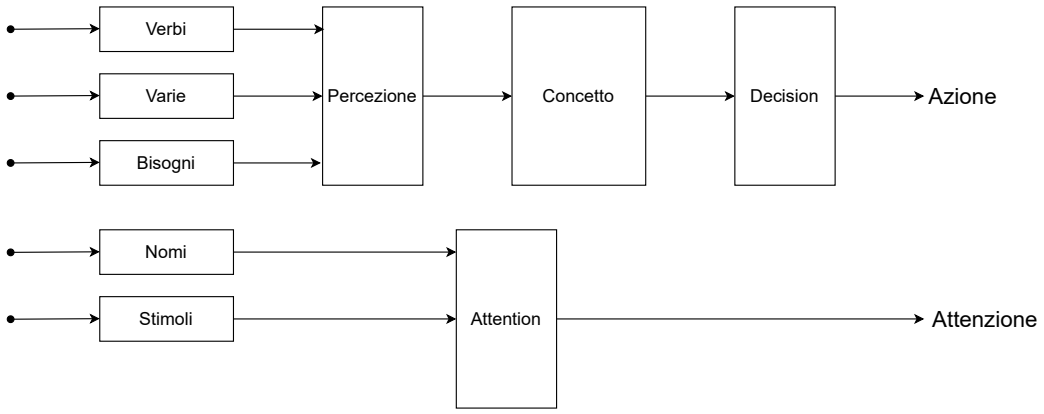

Figura 2.6: Schema di funzionamento del cervello dei "Norns" [GC98], in cui si dettaglia la struttura e la collaborazione fra lobi. Sebbene il contributo di Attention non venga sfruttato direttamente dal lobo Decision, la modalità di attuazione del comportamento degli agenti dipende dal binomio azione-attenzione, riflettuto nella struttura dell'input basato sul binomio nome-verbo oltre che sugli stimoli ambientali.

moli del mondo sarebbero lo strato di input e i lobi dedicati alla decisione delle azioni da intraprendere sarebbero lo strato di output.

In tal senso ci si concentra particolarmente sui lobi Attention e Decision, rispettivamente responsabili della determinazione del tipo di oggetto su cui l'agente sofferma la propria attenzione e sull'azione da svolgere nei confronti di tale oggetto [GC98]. Tale approccio permette ai "Norns" di stabilire un flusso di ragionamento basato sul binomio soggetto-azione piuttosto che sul solo concetto di azione, permettendogli quindi di apprendere strategie ulteriormente raffinate che si basino su una pre-elaborazione dello stato per l'estrazione di informazioni complesse (nel caso di Creatures un tipo di oggetto che attira l'attenzione dell'agente) che possono essere usate per selezionare l'azione da svolgere. Si cita in questo senso anche uno studio contemporaneo alla realizzazione di questa Tesi che propone un modello di DRL basato su Attention per il gioco di Enduro—appartenente alla libreria Atari 2600—in grado di superare le prestazioni del modello proposto da [MKS+15] basandosi anch'esso sull'analisi delle schermate come unico dato grezzo di input [VDN26].

Il modello di apprendimento. Alla nascita di un "Norn", all'interno del suo cervello vengono formate delle connessioni con un valore di forza prestabilito. Ad ogni passo decisionale viene poi applicato al valore di forza di tutte le connessioni non attivate un coefficiente di decadimento, che può a lungo termine portare all'atrofia della connessione. Tale processo, che ricalca il processo di decadimento delle connessioni dei cervelli biologici, prevede che al raggiungimento del livello di atrofia la connessione venga eliminata e al suo posto ne venga creata una nuova in maniera casuale. Le connessioni che vengono attivate ricevono invece un indebolimento o un rafforzamento che dipende dagli effetti che l'azione ha sui parametri vitali dell'agente, modellando di fatto un sistema di ricompense analogo a quello dei sistemi di RL tradizionali. Anche in questo caso le connessioni che raggiungono il limite minimo per il valore di forza sono eliminate e al loro posto ne vengono casualmente generate di nuove [GC98].

2.4 Direzione di ricerca della Tesi

L'obiettivo della ricerca svolta in questa Tesi, riprendendo il risultato ottenuto da Creatures e integrando tecnologie e contributi più recenti, è quindi quello di valutare l'efficacia e l'efficienza di un sistema di gioco che simuli un ambiente di vita artificiale per agenti autonomi addestrati con DRL i cui parametri—vitali e di apprendimento—sono ottimizzati tramite un GA. In particolare si pone l'attenzione su alcuni temi fondamentali:

- la realizzazione di un sistema di controllo basato su DRL e sulla collaborazione di due DNNs: una prima rete con il ruolo di Attention, che processa lo stato dell'agente per estrarre una rappresentazione semanticamente espressiva dell'oggetto su cui dirigere la propria attenzione, e una seconda rete che in cascata sia in grado di determinare, basandosi sul prodotto della rete Attention e su una visione più limitata del proprio stato, l'azione ottimale da intraprendere;

- l'osservazione dell'impatto che l'evoluzione tramite GA ha sul rendimento del modello di apprendimento, estendendo il processo evolutivo anche a ca-

ratteristiche fisiologiche degli agenti per valutare l'impatto che anche tali parametri hanno sulla qualità delle strategie apprese;

- l'integrazione del sistema di controllo degli agenti autonomi all'interno del motore di gioco stesso e la valutazione dell'efficienza del modello di ottimizzazione proposto, per valutare la possibilità di integrare sistemi analoghi in videogiochi commerciali.

Capitolo 3

Contributo

Il progetto proposto riprende la direzione esplorata da Creatures nel 1996 [GCM97] e si basa sull'integrazione all'interno del motore di gioco di agenti autonomi addestrati tramite DRL basato su Attention, i cui parametri vitali e di apprendimento sono ottimizzati grazie al contributo di un GA.

Si introduce quindi in primo luogo il funzionamento del sistema implementato, dettagliando in particolare la struttura del mondo di gioco e le possibili modalità di interazione fra utente e agenti autonomi. Dopodiché, si introducono le caratteristiche principali degli agenti e dei parametri che ne regolano il ciclo di vita.

L'attenzione si sposta poi sulla modellazione del sistema di controllo degli agenti, e nella Sezione 3.1 viene dettagliata l'implementazione dell'algoritmo di DRL usato per l'addestramento, di cui sono introdotti i parametri coinvolti dal processo di ottimizzazione del GA.

Nella Sezione 3.2 sono poi dettagliati la struttura del genoma degli agenti, la definizione della funzione fitness e il funzionamento generale del GA responsabile dell'evoluzione della popolazione.

Infine, nella Sezione 3.3 sono introdotte le modalità di implementazione del progetto, soffermandosi sulle tecnologie usate e sulle scelte più strettamente legate allo sviluppo del sistema.

Motore di gioco. Il gioco si svolge in un mondo bidimensionale a vista dall'alto all'interno di cui gli agenti si possono muovere liberamente. La partita viene ini-

ziata a mappa vuota, e l'utente ha la possibilità di richiedere in qualsiasi momento la generazione di un individuo con genoma casuale. La popolazione generata su richiesta dell'utente fa da popolazione base per l'applicazione del GA, che viene innescato alla morte degli agenti per generarne di nuovi e tutelare la dimensione minima richiesta della popolazione.

La dinamica di gioco standard prevede poi che gli agenti apprendano autonomamente con l'obiettivo di sopravvivere il più a lungo possibile. Sebbene non sia strettamente necessario per lo svolgimento del gioco, l'utente può comunque selezionare un agente per monitorarne lo stato e interagirvi. L'interazione si basa sulla possibilità di inviare all'agente stringhe di testo fino a 8 caratteri e segnali di ricompensa. In questo modo è possibile fornire una componente di input con cui guidare il processo decisionale degli agenti, sfruttando poi il sistema di ricompense per influenzarne il processo di apprendimento ed instaurare una forma molto semplice e primitiva di comunicazione basata sul linguaggio naturale.

Agenti di gioco. Durante la propria vita ciascun agente di gioco va incontro a un naturale processo di decadimento dei propri parametri vitali—fame, vita e stanchezza—che devono essere mitigati per garantire la sopravvivenza dell'agente stesso. La fame e la stanchezza, inizialmente poste a zero, aumentano a ritmo costante fino al raggiungimento di una soglia critica; quando anche solo uno di tali bisogni raggiunge il limite di sopportazione dell'agente, comincia a decadere il suo valore di vita. Se tale valore decade fino a raggiungere lo zero, l'agente muore e viene eliminato dal campo di gioco.

Per regolare i propri parametri vitali l'agente, che viene generato in un'area centrale della mappa detta *playground*, deve raggiungere delle aree di ristoro poste ai margini del mondo di gioco e interagirvi. In particolare, ai lati del playground sono poste due aree di recupero per la vita, mentre ai quattro angoli sono poste due aree per il recupero della fame e due per il recupero della stanchezza come illustrato dalla Figura 3.1.

Sistema di controllo degli agenti. Gli agenti di gioco possono muoversi nelle 4 direzioni principali o interagire con l'ambiente. In caso di interazione, se l'agente si trova all'interno di uno dei punti di ristoro l'effetto è quello di regolare comple-

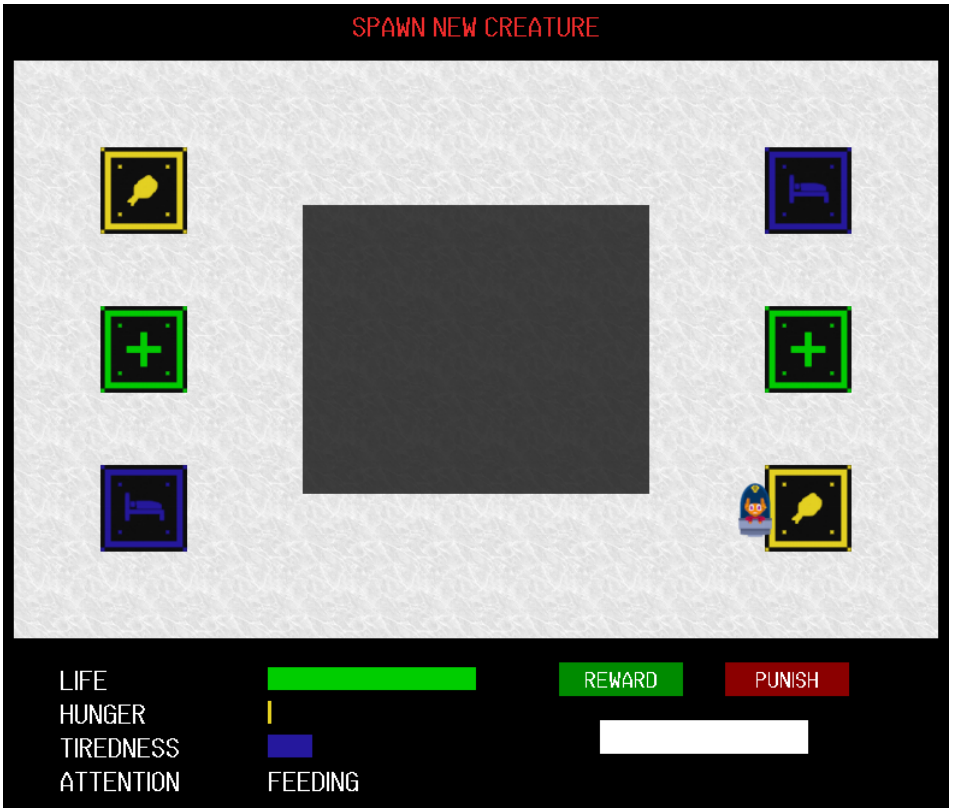

Figura 3.1: Cattura del mondo di gioco eseguita durante l'esecuzione, selezionando l'agente per attivare la dinamica di monitoraggio e interazione dell'utente. Nella parte inferiore dello schermo si vedono i valori dei parametri vitali dell'agente—vita, fame e stanchezza—l'oggetto dell'attenzione dell'agente e i comandi per l'interazione: un campo per l'inserimento di testo e due bottoni per l'invio di ricompense. Nella parte superiore si vede invece il bottone per la generazione di un nuovo individuo. All'interno della mappa si notano il playground—la zona scura in posizione centrale—e le aree di ristoro: in verde per il recupero della vita, in blu per il recupero della stanchezza e in giallo per il recupero della fame.

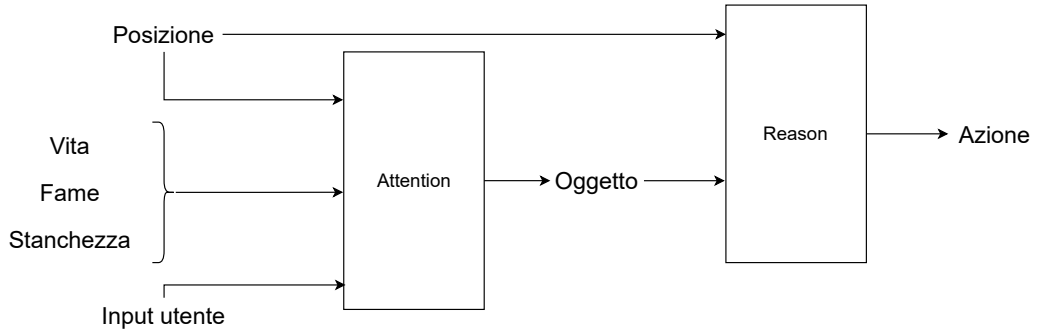

Figura 3.2: Schema di funzionamento del processo decisionale degli agenti di gioco, che modella esplicitamente la collaborazione fra i due Percettroni Multistrato di controllo.

tamente la funzione vitale corrispondente, mentre l'azione è inefficace se l'agente si trova in un qualsiasi altro punto della mappa.

Il comportamento degli agenti è controllato da due Percettroni Multistrato (MLPs)—ovvero DNNs senza connessioni all'indietro, in cui ogni neurone riceve stimoli da tutti i neuroni dello strato precedente e invia i propri stimoli elaborati a tutti i neuroni dello strato successivo—chiamati Attention e Reason, che lavorano in cascata come illustrato dalla Figura 3.2. In particolare Attention lavora per prima per determinare il tipo di punto di ristoro verso cui l'agente dirige la propria attenzione e Reason sfrutta la decisione di Attention per determinare l'azione più fruttuosa da svolgere.

La rete Attention basa il proprio processo decisionale sui valori dei parametri vitali dell'agente, sulla distanza da ciascun tipo di punto di ristoro e sulla stringa di input eventualmente indicata dall'utente. In questo modo l'agente sfrutta la completa visione che ha di sé e dell'ambiente circostante per dirigere la propria attenzione, che può essere però influenzata anche dall'apporto di input fornito dall'utente.

La rete Reason basa invece il proprio processo decisionale sull'oggetto dell'attenzione determinato dalla rete Attention e sulla posizione dell'agente rispetto ai punti di ristoro. In questo modo è in grado di scegliere l'azione più fruttuosa dalla propria posizione rispetto all'oggetto della propria attenzione, che rappresenta

una sintesi significativa della visione più ampia del proprio stato che ha la rete Attention.

L'input testuale dell'utente non ha influenza diretta sul processo decisionale della rete Reason, e costituisce quindi una forma di comunicazione basata esclusivamente sul nome e limitata al concetto di attenzione. Le ricompense somministrate dall'utente vengono invece recepite anche dalla rete Reason, seppur in forma mitigata, per permettere all'utente di guidare il processo di apprendimento degli agenti.

3.1 Algoritmo di apprendimento

Il processo di apprendimento di entrambe le reti avviene tramite DRL, con un algoritmo basato su Q-learning simile a quello proposto per il dominio dei giochi dell'Atari 2600 [MKS+15].

Abbattimento della dimensionalità. Sebbene in grado di affrontare i problemi del bilanciamento tra scelte localmente ottime e lungimiranti e del compromesso fra esplorazione dell'ambiente e sfruttamento delle conoscenze apprese, l'implementazione del Q-learning dettagliata nell'Algoritmo 1 presenta infatti nella sua formulazione tradizionale un problema importante di scalabilità: la dimensione della Q-table dipende direttamente dalla dimensione dello spazio $S \times A$ delle coppie stato-azione. Nel dominio studiato, in cui sia la posizione degli agenti che i valori dei parametri vitali sono valori continui, e in cui qualsiasi stringa di lunghezza inferiore agli 8 caratteri è un possibile input, mantenere in memoria i valori di qualità per ogni scelta in ogni stato diventerebbe insostenibile.

Si propone quindi un'implementazione modificata del Q-learning che sfrutti le capacità di approssimazione funzionale delle DNNs per calcolare i valori di qualità piuttosto che mantenerli in memoria. Si vuole quindi addestrare la DNN a prevedere, dato uno stato $s \in S$, i valori di qualità previsti per ogni azione $a \in A$.

Il processo decisionale del modello rimane sostanzialmente invariato. Denotando con θ_t i pesi della DNN all'istante t, dopo aver osservato lo stato s_t si prevedono i valori di qualità $Q(s_t, a, \theta_t) \forall a \in A$ e, con politica ε-greedy, l'agente sceglie se eseguire l'azione ottimale a^* o un'azione di esplorazione a scelta casualmente.

Addestramento della Rete Neurale Profonda. Al contrario, il passo di apprendimento deve essere riformulato per conformarsi ai metodi di addestramento delle DNNs. Ricalcando il passo di apprendimento dell'Algoritmo 1 si definisce qui il valore e di errore della singola previsione come

$$e = R(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a, \theta_t) - Q(s_t, a_t, \theta_t) \tag{3.1}$$

Al dato di errore si applica poi per il calcolo del valore di costo—fondamentale per l'addestramento della DNN, che ha come obiettivo la minimizzazione di tale risultato—la funzione Huber [Hub64]. Quest'ultima è stata scelta perché combina le formulazioni dell'Errore Assoluto Medio (MAE) e dell'Errore Quadratico Medio (MSE) per ridurre l'impatto degli *outliers*—ovvero i valori molto lontani dalla media dei risultati osservati, che causano solitamente grandi errori di previsione—sul processo di apprendimento senza però ridurre l'espressività degli errori meno pronunciati. In questo modo permette di mitigare l'instabilità propria dell'algoritmo Q-learning—sottolineata anche in altri studi, ma che con i dovuti accorgimenti può essere quasi completamente superata [MKS+15]—senza però ridurre l'espressività delle osservazioni svolte.

L'ottimizzazione dei pesi delle DNNs, basata sui valori di errore prodotti dalla funzione Huber, viene poi svolta con il metodo di discesa del gradiente Adaptive Moment Estimation (Adam) [KB15]. Tale metodo risulta particolarmente efficace perché sfrutta l'accumulo dei gradienti passati per determinare dei learning rate differenziati per ciascun peso da ottimizzare. Inoltre introduce degli accorgimenti per mitigare il condizionamento dovuto all'inizializzazione a zero dei valori dei gradienti accumulati, velocizzando la convergenza e riducendone l'instabilità.

Apprendimento per batch. La definizione proposta per la singola iterazione del Q-learning, che prevede di aggiornare i valori della Q-table ad ogni passo decisionale, rappresenta uno dei punti di forza dell'algoritmo nella sua formulazione tradizionale. Nel passaggio alla formulazione basata su DNN tale definizione presenta però due problemi fondamentali.

Il primo, legato al costo computazionale, risiede nell'impossibilità pratica di applicare un passo di apprendimento ad ogni passo decisionale: il passo di adde-

stramento di una DNN è infatti particolarmente oneroso perché richiede di ricavare l'errore della previsione applicando la funzione costo, di calcolare i gradienti dell'errore rispetto a tutti i pesi della rete e infine di applicare il passo di ottimizzazione dei pesi aggiornando i valori degli stessi.

Il secondo è invece un problema di generalità: se nella formulazione tradizionale del Q-learning aggiornare a ogni osservazione è fruttuoso perché la Q-table è una rappresentazione esaustiva dello spazio dei valori di qualità, nella formulazione del Q-learning basata su DNN i valori di qualità di ogni azione vengono derivati dallo stato tramite una funzione approssimante espressa dai pesi della rete, che quindi deve essere generale e il più possibile indipendente dalle perturbazioni istantanee.

Si introduce quindi il concetto di *apprendimento per batch*. Definendo come batch un insieme di osservazioni usato per il singolo passo di apprendimento, si definiscono due momenti principali dell'algoritmo. Il primo, dedicato alla raccolta dati, prevede che venga innescato il solo processo decisionale della DNN e che si memorizzino le osservazioni che derivano dalle interazioni con l'ambiente. Il secondo prevede poi di svolgere il singolo passo di apprendimento su un batch di osservazioni, con l'obiettivo di raccogliere valutazioni di errore più generali e pertanto più significative.

L'apprendimento per batch prevede quindi necessariamente—dovendo raccogliere un numero di osservazioni sufficiente alla creazione del singolo batch—che i passi di addestramento siano meno frequenti dei passi decisionali. Ciò permette di stabilizzare, oltre al processo di apprendimento, anche il comportamento dell'agente, che in questo modo ha la possibilità di svolgere diverse azioni con la stessa configurazione di pesi della DNN.

Experience Replay Buffer. L'efficacia dell'apprendimento per batch si basa però sul presupposto che le osservazioni del batch siano fra loro indipendenti, condizione non verificata nel caso del Q-learning. Dal momento che il valore di qualità di una singola azione dipende anche dal valore previsto per le azioni successive, apprendere su un batch di azioni processate nello stesso ordine in cui sono state svolte potrebbe portare il modello ad auto-condizionarsi rendendo meno stabile il processo di apprendimento.

Per superare questo problema si introduce l'*Experience Replay Buffer*, uno spazio di memoria—idealmente illimitato, ma che nella pratica viene limitato a una dimensione molto maggiore a quella del batch ottenendo un risultato equivalente—in cui l'agente salva lo storico di tutte le proprie osservazioni dell'ambiente. Al momento dell'apprendimento viene quindi estratto un campione casuale dell'Experience Replay Buffer per popolare il batch su cui addestrare la DNN. Grazie a questo accorgimento le osservazioni contenute in ogni batch diventano realmente indipendenti e si può sfruttare la massima efficacia di tale metodo di apprendimento.

L'introduzione del metodo di apprendimento per batch e dell'Experience Replay Buffer richiede di definire altri tre parametri dell'algoritmo di apprendimento: la dimensione dell'Experience Replay Buffer, la dimensione del batch e il periodo di addestramento τ, ovvero il tempo che intercorre fra passi di addestramento successivi.

Modello Target. La definizione di errore proposta nell'Equazione (3.1) fa emergere però un ulteriore problema di auto-condizionamento: il calcolo delle ricompense future—che a tutti gli effetti sono una componente del valore di qualità reale, obiettivo del processo di apprendimento della DNN—avviene sfruttando la stessa configurazione di pesi θ_t usata per il calcolo del valore di qualità previsto. Ciò, sommato al fatto che lo stato s_{t+1} su cui viene svolta la previsione di qualità futura è determinato almeno in parte dalla coppia stato-azione (s_t, a_t), potrebbe influenzare in maniera sostanziale il processo di apprendimento impedendo all'agente di raggiungere la strategia ottimale.

Per mitigare tale forma di condizionamento si introduce il *modello Target*, una DNN di supporto usata esclusivamente per le previsioni di qualità delle scelte future. L'obiettivo è quello di permettere il calcolo delle ricompense future con una configurazione di pesi differente e indipendente da quella della DNN principale, fortemente influenzata dallo stato istantaneo dell'apprendimento, rendendo i valori ottenuti ulteriormente più consistenti e regolari.

Volendo comunque evitare di innescare un secondo processo di addestramento per la DNN Target—che richiederebbe ingenti risorse computazionali, e sarebbe svolto sulle stesse osservazioni e con lo stesso obiettivo di quello svolto per la DNN

principale—si possono aggiornare i suoi pesi effettuando una copia della configurazione del modello principale all'interno del modello Target. In questo modo, salvo l'istante in cui effettivamente sono copiati i pesi, le due DNNs non hanno mai la stessa configurazione e le previsioni svolte si possono reputare indipendenti, senza azionare un secondo processo di apprendimento ma piuttosto riusando in maniera efficace la conoscenza appresa dalla DNN principale.

Si propone quindi la seguente formula modificata per il calcolo del valore di costo c, sempre basata sull'applicazione della funzione Huber, dove si denotano con θ' i pesi della DNN Target:

$$c = Huber(R(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a, \theta'_t) - Q(s_t, a_t, \theta_t)) \tag{3.2}$$

Introdurre il modello Target richiede però di stabilire un ulteriore parametro dell'apprendimento: il periodo τ' che regola la frequenza di aggiornamento dei pesi della DNN Target stessa. Dal momento che i pesi da assegnare sono direttamente copiati dalla configurazione della DNN principale si pone $\tau' > \tau$ per evitare di sprecare risorse computazionali copiando più volte la stessa configurazione nel modello Target.

Parametri del Deep Q-learning. Si riassume quindi nell'Algoritmo 2 l'implementazione proposta di Deep Q-learning—ovvero Q-learning basato su DNN. Si sottolineano in particolare tutti i parametri i cui valori non sono fissati:

- il tasso di esplorazione iniziale ε_0;
- il coefficiente di decadimento del tasso di esplorazione λ_ε;
- il tasso di esplorazione minimo ε_{min};
- il discount factor γ per le ricompense future;
- il learning rate α per il metodo di ottimizzazione Adam;
- il periodo di aggiornamento della DNN principale τ;
- il periodo di aggiornamento della DNN Target τ'.

Algoritmo 2 Deep Q-learning con politica ε-greedy

Initialize θ, θ'
Initialize empty *ERB* ▷ Experience Relay Buffer
$\varepsilon \leftarrow \varepsilon_0$
$s_t \leftarrow s_0$
Define a_t, r_t, s_{t+1}
repeat
  With probability ε: $a_t \leftarrow$ random $a \in A$
  With probability $1 - \varepsilon$: $a_t \leftarrow \max_a Q(s_t, a, \theta)$
  if $\varepsilon > \varepsilon_{min}$ **then**
    $\varepsilon \leftarrow \lambda_\varepsilon \varepsilon$
  end if
  Perform a_t
  Observe r_t, s_{t+1}
  Store (s_t, a_t, r_t, s_{t+1}) in *ERB*
  if τ since last Main Model update **then**
    Pick a batch of random samples from *ERB*
    for all random samples **do**
      $e_t = r_t + \gamma \max_a Q(s_{t+1}, a, \theta') - Q(s_t, a_t, \theta)$
      Perform Adam optimization with rate α on $Huber(e_t)$
    end for
  end if
  if τ' since last Target Model update **then**
    $\theta' \leftarrow \theta$
  end if
  if *ERB* is full **then**
    Clear *ERB* content
  end if
until end

L'ottimizzazione dei valori di tali parametri è responsabilità del GA, mentre al contrario la topologia delle DNNs coinvolte, la dimensione dei batch e la dimensione degli Experience Replay Buffer sono fissate a priori.

Sistema di ricompense. Ai fini dell'addestramento degli agenti il sistema di gioco prevede due contributi di ricompensa principali: il primo è determinato dall'utente e somministrato tramite il meccanismo di interazione con gli agenti di gioco, mentre invece il secondo è determinato dall'ambiente e viene recepito ad ogni passo decisionale. Dal momento che il comportamento degli agenti è controllato da due DNNs che implementano diverse funzioni del processo decisionale, l'intensità della ricezione delle ricompense è differenziata fra le due reti per facilitare a ciascuna l'apprendimento della propria funzione.

Il contributo di ricompensa somministrato dall'utente è circoscritto alla dinamica di interazione attivabile selezionando un agente di gioco. Può essere quindi usato per guidare l'apprendimento in senso lato, ma la sua funzione primaria è quella di influenzare la rete Attention per stabilire una forma di comunicazione tra agente e utente. Tale contributo quindi, che di default vale zero a meno di interazioni esplicite dell'utente, viene recepito interamente dalla rete Attention e solo in forma attenuata dalla rete Reason.

Il contributo di ricompensa determinato dall'ambiente tiene invece conto di due componenti separate. La prima è legata allo stato dei parametri vitali dell'agente: in caso di perdita della vita o di aumento di fame e stanchezza esso riceve un segnale negativo, mentre al contrario ne riceve uno positivo in caso di mitigazione di un bisogno—a patto che comunque non si sia registrata una perdita di vita. La seconda è legata invece alla posizione dell'agente: in caso di allontanamento dall'oggetto della propria attenzione viene ricevuto un segnale negativo, mentre al contrario ne viene ricevuto uno positivo in caso di avvicinamento.

Dal momento che il contributo di ricompensa ambientale è strettamente legato alle azioni dell'agente, viene recepito interamente dalla rete Reason e solamente in forma attenuata dalla rete Attention. La ricezione di questo contributo da parte della rete Attention deve comunque essere garantita in primo luogo perché è l'unico contributo di ricompensa che l'agente riceve nella dinamica di gioco standard—che non richiede necessariamente l'intervento dell'utente—e in secondo luogo perché il

suo valore è comunque in parte determinato dall'oggetto dell'attenzione dell'agente. Il valore complessivo di ricompensa ambientale quantifica infatti la qualità della collaborazione fra le due reti di controllo, essendo basato sulla valutazione delle transizioni di stato causate dalle azioni scelte dalla rete Reason in combinazione con gli oggetti dell'attenzione determinati dalla rete Attention.

3.2 Algoritmo genetico

Il sistema di gioco prevede che alla morte di un agente, quando la dimensione della popolazione scende al di sotto della soglia stabilita dall'utente, venga azionato un GA in grado di generare un nuovo individuo che erediti ed evolva le caratteristiche degli individui più prestanti tra quelli in vita.

Il genoma. Il primo elemento necessario per la modellazione di un GA è il *genoma* degli individui, ovvero la codifica strutturata delle caratteristiche interessate dal processo evolutivo, composto da 18 geni a valori continui limitati ciascuno all'interno di un intervallo di valori ammissibili. I 18 geni contengono il valore di tutti i parametri dell'apprendimento precedentemente elencati per ciascuna delle due reti di controllo—quindi 7 geni per la rete Attention e altrettanti per la rete Reason—e 4 geni addizionali che rappresentano la velocità di movimento dell'agente, il tasso di decadimento della vita e i tassi di aumento di fame e stanchezza.

La scelta di modellare all'interno del genoma anche alcuni parametri vitali non direttamente legati al processo di apprendimento è coerente con l'integrazione degli agenti all'interno del motore di gioco—che diventa quindi sostanzialmente un motore di vita artificiale—e permette di studiare l'impatto che anche questi hanno sul processo di apprendimento. Non è infatti da escludere che un individuo di per sé più prestante, e quindi più rapido o più resistente, sia avvantaggiato indipendentemente dai parametri dell'algoritmo di apprendimento perché maggiormente in grado di esplorare l'ambiente di gioco.

La funzione fitness. Un ulteriore elemento da definire prima di modellare le modalità di applicazione del GA è la *funzione fitness*, che dato un individuo della

popolazione restituisce un valore numerico che ne rappresenta la qualità: maggiore è il valore restituito dalla funzione, maggiore è la qualità dell'individuo preso in considerazione. La definizione della fitness è quindi cruciale per l'efficacia del GA in quanto stabilisce la metrica di valutazione delle prestazioni di ogni individuo, e pertanto deve modellare correttamente gli obiettivi dell'evoluzione.

L'evoluzione guidata dal GA nel contesto di gioco ha come obiettivo la massimizzazione dell'efficacia del modello di apprendimento. Essa si può misurare tramite due metriche fondamentali: il tempo di vita complessivo e il valore medio dei parametri vitali dell'agente.

Il tempo di vita è usato come metrica primaria perché si suppone che gli agenti più longevi—indipendentemente dalla maggiore resistenza determinata dai geni regolatori dei parametri vitali—lo siano in quanto capaci di apprendere una strategia di sopravvivenza efficace.

Al contrario il valore medio dei parametri vitali degli agenti è usato in modo da garantire, a pari tempo di vita, la selezione di agenti mediamente più sani, supponendo che ciò sia frutto di una strategia appresa in grado di tutelare e mitigare i valori dei parametri vitali prima del raggiungimento delle soglie critiche.

Si propone quindi la seguente definizione di fitness (denotata con f), dove si indicano con $\overline{l}$ il valore medio del parametro della vita, e con $\overline{h}$ e $\overline{t}$ i complementari dei valori medi di fame e stanchezza rispettivamente. L'uso dei complementari per fame e stanchezza è dovuto al fatto che tali bisogni sono soggetti a un processo di decadimento ascendente—ovvero i valori ottimali sono prossimi allo zero, e i valori pessimi sono elevati—ed è necessario per far coincidere le medie ottimali a valori alti di fitness.

$$f = T_l \frac{\overline{l} + \frac{\overline{h}+\overline{t}}{2}}{2} \tag{3.3}$$

La formulazione proposta nell'Equazione (3.3) per il calcolo del valore di fitness si basa quindi sulla media pesata dei valori medi dei parametri vitali—in cui la vita ha peso uguale ai restanti bisogni combinati—moltiplicata per il tempo di vita complessivo. Il tempo di vita complessivo diventa quindi la metrica predominante per la selezione, escludendo i "neonati"—che altrimenti avrebbero valori medi per i bisogni vitali prossimi all'estremo ideale—e prediligendo la selezione di agenti

longevi indipendentemente dallo stato di salute per premiare l'apprendimento di strategie di sopravvivenza efficaci.

Selezione. Il modello proposto per la selezione dei genitori è un algoritmo di selezione a roulette con accettazione stocastica del risultato [LL12]. La classe di algoritmi a roulette, che prende il nome dall'omonimo gioco francese, prevede di rappresentare le possibili scelte per i genitori all'interno di un intervallo continuo, suddiviso in un numero di sotto-intervalli associati ciascuno a un individuo selezionabile, solitamente di ampiezza variabile e proporzionale al valore di fitness dell'individuo rappresentato. Il processo di selezione consiste poi nell'estrazione di un valore casuale all'interno dell'intervallo complessivo, che identifica con il proprio sotto-intervallo di appartenenza il genitore scelto.

La definizione dei sotto-intervalli ad ampiezza variabile, che da un lato permette di definire probabilità diverse di selezione per ciascun individuo, rende però computazionalmente oneroso risalire al sotto-intervallo di appartenenza del valore estratto. Si propone quindi un algoritmo modificato di selezione basato su intervalli ad ampiezza fissa—che abbatte quindi a un O(1) la complessità della valutazione del sotto-intervallo di appartenenza—con accettazione probabilistica del risultato dell'estrazione [LL12]. Esso permette quindi istantaneamente di individuare il potenziale genitore, e di accettarlo finalizzando la selezione con una probabilità proporzionale al suo valore di fitness.

Denotando con B l'insieme degli individui in vita e come $b \in B$ il singolo individuo, si definisce la probabilità di accettazione del risultato come segue:

$$P(b) = \frac{f(b)}{\sum_{b' \in B} f(b')} \tag{3.4}$$

La probabilità di accettazione proposta dallEquazione (3.4), basandosi sul presupposto che $f(b) > 0 \, \forall \, b \in B$—sempre verificato, dal momento che i valori medi per i parametri vitali sono inizializzati al massimo valore ammissibile e il tempo di vita è sempre maggiore di zero a meno dell'istante in cui l'agente viene generato—garantisce di non escludere né garantire la selezione di nessun genitore.

Ciò permette, pur favorendo la selezione di individui con elevati valori di fitness, l'esplorazione dello spazio dei genomi ammissibili alla ricerca di configurazioni

potenzialmente efficaci che possono emergere ricombinando e mutando i genomi di individui poco prestanti.

Ricombinazione. La ricombinazione applicata ai genomi dei genitori per determinare quello ereditato è uniforme e bilanciata. Ricombinare in modo uniforme significa che ogni gene viene ereditato singolarmente, senza individuare un numero di punti di taglio prestabilito né sequenze di geni da ereditare in maniera unitaria. Tale metodo di ricombinazione è quello che più di tutti favorisce l'esplorazione dello spazio dei genomi ammissibili, dal momento che non pone vincoli alle permutazioni di geni ottenibili.

Ricombinare in modo bilanciato significa poi che, per ciascun gene, la probabilità di eredità da ciascun genitore è del 50%, diversamente dai metodi sbilanciati che prevedono di assegnare probabilità diverse ai genitori, eventualmente dipendenti dai valori di fitness. Tale metodo favorisce l'esplorazione di nuove configurazioni dal momento che non esclude né sfavorisce l'eredità dai genitori meno prestanti, che comunque potrebbero custodire valori di singoli geni localmente ottimali.

Mutazione. Dopo aver selezionato e ricombinato i genomi, ogni gene del nuovo individuo è comunque uguale a quello ereditato da uno dei genitori. Per esplorare efficacemente lo spazio dei genomi ammissibili e favorire la convergenza verso configurazioni ottimali, si applica puntualmente a ciascun gene nua mutazione con probabilità $P = 0.1$. Essendo il genoma composto da 18 geni distinti, ciascuno soggetto alla stessa probabilità di mutazione in maniera indipendente, la probabilità di avere un genoma completamente immutato è pari a circa 0.15. Ciò significa che mediamente circa un genoma su 7 risulta invariato, mentre gli altri 6 ricevono un qualche tipo di mutazione.

Quando viene applicata una mutazione, il suo valore viene estratto casualmente da una distribuzione normale centrata in zero con ampiezza pari a quella dell'intervallo dei valori ammissibili per il gene coinvolto. In questo modo si prediligono mutazioni lievi, con l'obiettivo di compensare almeno parzialmente l'aggressività della politica di mutazione applicata. Il valore di mutazione viene poi sommato al valore ereditato, limitando comunque il valore ottenuto all'interno dell'intervallo di ammissibilità per il gene coinvolto.

3.3 Modalità di implementazione

Il codice del progetto è disponibile in un repository GitHub[2] ed è scritto in Python.

Per l'implementazione del sistema si è deciso di non affidarsi a un motore di gioco strutturato, reimplementando piuttosto le funzionalità centrali come la gestione del main loop, del rendering e dell'input utente. Si è deciso di re-implementare un semplice motore di gioco, nonostante il costo legato allo sviluppo, per garantire di avere il massimo controllo sull'andamento degli aggiornamenti dell'ambiente. Inoltre, l'indipendenza da sistemi strutturati ha permesso di avere più libertà per l'implementazione delle singole entità di gioco, facilitando l'integrazione del modello di apprendimento studiato.

La realizzazione di un motore di gioco personalizzato ha infatti permesso di gestire alla granularità del singolo passo anche i processi di addestramento e previsione delle DNNs di controllo degli agenti, condizione necessaria per lo studio approfondito delle prestazioni del modello proposto.

[2]Repository pubblico disponibile al link `https://github.com/kimiosti/ArtieLife`

Capitolo 4

Valutazione dei risultati

La valutazione dei risultati ottenuti si basa sull'analisi dell'efficacia dell'applicazione contemporanea di due sistemi di ottimizzazione: il GA che lavora fra generazioni per creare individui sempre più prestanti, e il DRL che lavora a livello del singolo individuo per raffinarne la strategia comportamentale e massimizzarne il tempo di vita. Valutare quindi ciascun algoritmo in maniera indipendente permette di fomulare un giudizio più approfondito sull'impatto che entrambi hanno sul rendimento complessivo del sistema, analisi che sarebbe impossibile osservando un semplice dato unidimensionale di qualità per il modello combinato.

Infine si vuole valutare l'impatto che l'applicazione dei due algoritmi ha sulle prestazioni complessive del motore di gioco. In tal senso si può considerare predominante la componente di complessità del modello di DRL, assimilando al contrario il costo computazionale del GA a quello richiesto dalle naturali dinamiche di gioco e di rendering. Si rivela quindi sufficiente una valutazione assoluta del sistema nel suo complesso senza paragoni di riferimento.

L'ambiente di valutazione. Per permettere di valutare separatamente il GA e il modello di DRL è stato necessario modificare il progetto aggiungendo delle opzioni per attivare o disattivare ciascun algoritmo in maniera indipendente. Per rendere più rapida la raccolta dati si è inoltre aggiunta la possibilità di avviare il gioco senza interfaccia grafica ed eventualmente con più mondi in parallelo, definendo all'avvio una dimensione fissa per la popolazione.

Ottimizzazione	FPS	$E(T_l)$	$\frac{T_l}{E(T_l)}$	Individui inefficaci
Nessun algoritmo	29.73	110.39	1.22	171(81.43%)
Algoritmo Genetico	29.91	263.09	1.89	83(39.52%)
Deep Reinforcement Learning	9.23	107.05	1.52	89(42.38%)
Approccio combinato	9.60	176.29	2.37	18(8.57%)

Tabella 4.1: Tabella riassuntiva delle prestazioni del modello nelle configurazioni studiate, attivando o disattivando i contributi di ottimizzazione del Deep Reinforcement Learning e dell'Algoritmo Genetico. Si denota con FPS la media dei Frame al Secondo—metrica di efficienza complessiva del sistema—con $E(T_l)$ il tempo di vita atteso in caso di inattività—metrica di valutazione dell'ottimizzazione dei parametri vitali—con $\frac{T_l}{E(T_l)}$ il rapporto fra il tempo di vita effettivo e il tempo di vita atteso—metrica di valutazione dell'efficacia della strategia appresa—e sono indicati come "inefficaci" quegli individui che hanno appreso una strategia che non gli ha permesso di estendere il proprio tempo di vita oltre il 101% rispetto all'aspettativa in caso di inattività.

Per valutare l'efficacia del GA, esso viene paragonato a un algoritmo che assegna a ogni agente un genoma casuale. L'obiettivo è quello di esplorare uniformemente e in maniera esaustiva lo spazio dei genomi ammissibili per raccogliere un dato di efficacia attesa il più possibile rappresentativo di un individuo medio.

Per la valutazione dell'algoritmo di DRL si sfruttano invece agenti con comportamento completamente casuale, simulando di fatto una strategia ε-greedy con tasso di esplorazione fisso e uguale a 1. In questo modo si può valutare l'impatto che i soli passi decisionali di exploitation hanno sulla capacità di sopravvivenza degli agenti. Ai fini della valutazione del modello di apprendimento, si sono fissate la dimensione dell'Experience Replay Buffer a 1000 sia per la rete Attention che per la rete Reason—cercando un equilibrio fra esigenze di memoria ed efficacia del sistema implementato—e la dimensione dei batch a 32, anch'essa per entrambe le DNNs di controllo.

Le simulazioni per tutte le configurazioni di attivazione degli algoritmi sono state svolte su un singolo mondo, senza interfaccia utente e con una popolazione base di 7 individui. L'esecuzione è stata svolta su un laptop con CPU Intel Core i3-1500G1, GPU integrata e 8GB di RAM.

La durata delle simulazioni non è stata fissata a priori, ma determinata in

maniera adattiva per garantire la generazione di almeno 210 individui—ovvero 30 generazioni—su cui svolgere la valutazione.

4.1 Algoritmo genetico

Modalità di valutazione. La valutazione del GA richiede di quantificare la capacità del modello evolutivo di convergere verso valori ottimali dei geni. Per evitare possibili perturbazioni, in tutti i test relativi alla valutazione del solo GA il contributo del DRL è disattivato. Pertanto le ottimizzazioni studiate coinvolgono solamente i geni regolatori dei parametri vitali degli agenti: il tasso di decadimento della vita λ_l, il tasso di aumento della fame λ_h, il tasso di aumento della stanchezza λ_t e la velocità di movimento.

Metriche di valutazione. Una prima forma di valutazione qualitativa dell'efficacia del GA è l'osservazione dei grafici di andamento dei valori dei singoli geni coinvolti nel processo di ottimizzazione. Si associa quindi a ciascun agente un ID progressivo, che permette così di rappresentare in un grafico cartesiano ogni coppia agente-valore del gene sottolineando eventuali tendenze evolutive.

Si prevede di non poter osservare tendenze evidenti in caso di mancata applicazione del GA, dal momento che i genomi sono determinati in maniera casuale, mentre al contrario si prevede che l'attivazione del processo evolutivo induca una tendenza verso i valori ottimali per i geni coinvolti.

Una valutazione invece quantitativa della qualità del GA—dal momento che viene studiato solamente in relazione all'ottimizzazione dei geni che regolano i parametri vitali degli agenti—si ottiene calcolando il *tempo di vita atteso* di ogni agente. Tale valore, denotato con $E(T_l)$, rappresenta il tempo di sopravvivenza stimato per l'agente in caso di inattività completa e dipende direttamente dal genoma dello stesso. In particolare dipende dai geni λ_h e λ_t per il calcolo del tempo minimo di raggiungimento dello stato di salute critico, e dal gene λ_l per il calcolo del tempo necessario al decadimento completo della vita dell'agente.

Denotando quindi il valore iniziale della vita come l_{max} e i valori massimi di sopportazione per fame e stanchezza come h_{max} e t_{max} rispettivamente, si definisce la seguente formula per il calcolo del tempo di vita atteso:

$$E(T_l) = \min(\frac{h_{max}}{\lambda_h}, \frac{t_{max}}{\lambda_t}) + \frac{l_{max}}{\lambda_l} \tag{4.1}$$

Efficacia di base. La Figura 4.1 rappresenta i grafici di andamento dei geni regolatori delle funzioni vitali degli agenti in caso di mancata attivazione del GA. Come previsto, già da una prima analisi qualitativa risulta evidente che i valori dei geni oscillano senza tendenze evidenti, dal momento che sono determinati in maniera completamente casuale.

Diversamente, sembra essere osservabile una minore dispersione per i valori del tempo di vita atteso. A fronte di un intervallo di ammissibilità teorico compreso fra circa 58.33 secondi e 325 secondi—ricavato applicando l'Equazione (4.1) ai valori minimi e massimi ammissibili per i geni coinvolti—i valori effettivamente osservati sono infatti contenuti fra un minimo di circa 60.10 secondi e un massimo di circa 247.96 secondi, con una media di circa 110.39 secondi.

Inoltre il 50% dei valori registrati per il tempo di vita atteso è compreso nell'intervallo $[84.01, 132.53]$, centrato in 108.27 e ampio solamente un quarto rispetto all'intervallo dei valori osservati. Nonostante sia centrato in un valore vicino alla media complessiva dei valori osservati, l'ampiezza dell'intervallo lascia dedurre che la distribuzione dei valori del tempo di vita atteso non sia uniforme. Ciò accade perché tale misura dipende da tre geni i cui valori sono generati casualmente in maniera uniforme, e quindi per generare un valore di tempo di vita atteso prossimo a uno degli estremi di ammissibilità è necessario che tutti i geni coinvolti siano prossimi allo stesso estremo di ammissibilità, mentre qualsiasi altra combinazione introduce delle compensazioni che riportano il valore derivato verso il centro dell'intervallo di ammissibilità.

Infine si nota che il valore medio osservato per il tempo di vita atteso è di circa 110.39 secondi, mentre l'intervallo dei valori osservati è centrato in 154.03 secondi. Tale condizionamento verso il basso deriva dalla definizione stessa del tempo di vita atteso, che dipende dal tempo minimo di raggiungimento della soglia critica per fame e stanchezza. Pertanto basta che uno solo fra i geni λ_h e λ_t abbia un valore elevato per abbattere il tempo di vita atteso, mentre entrambi devono avere un valore contenuto per generare un individuo potenzialmente più longevo,

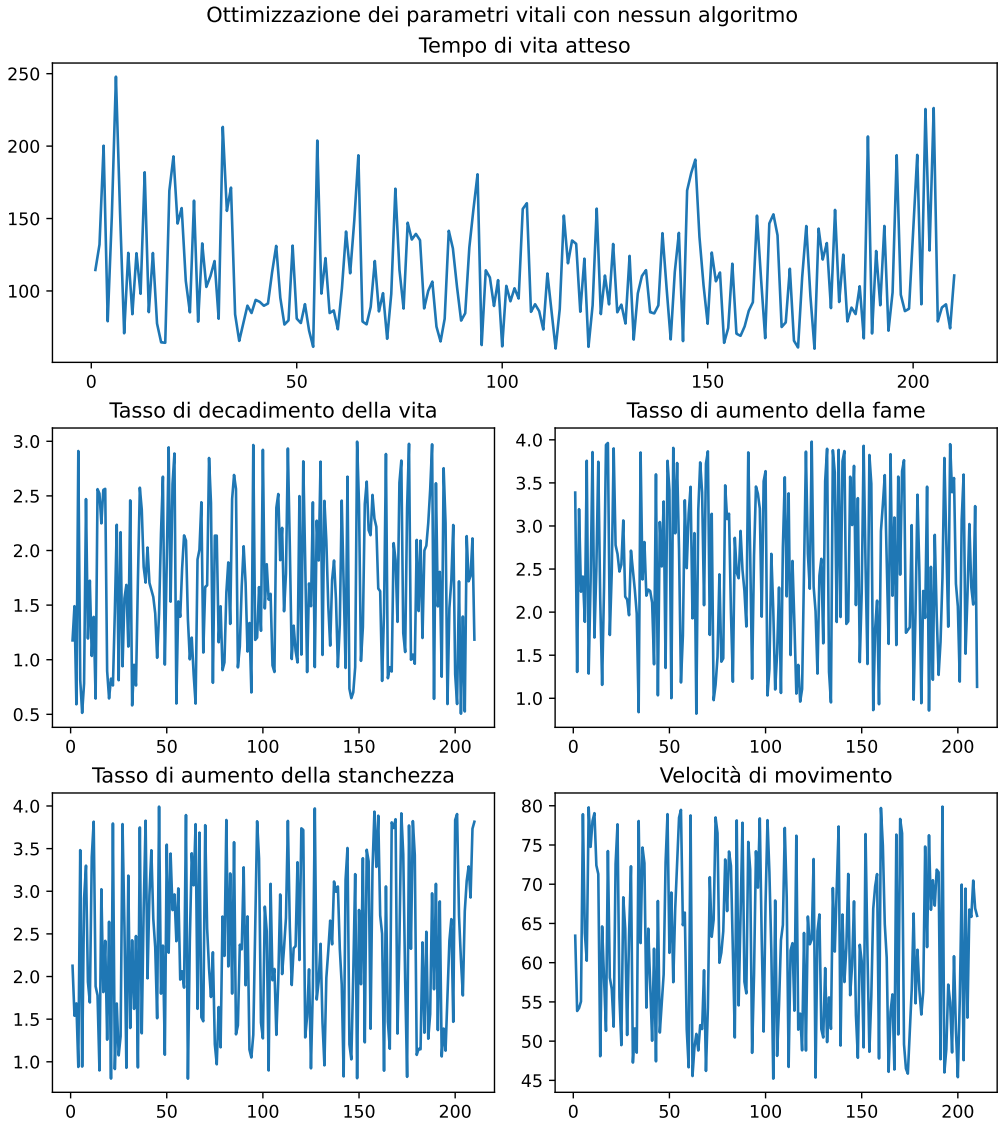

Figura 4.1: Grafici di valutazione dell'andamento dei geni regolatori dei parametri vitali e del tempo di vita atteso lungo le generazioni in assenza del contributo di ottimizzazione dell'Algoritmo Genetico. Sull'asse x sono rappresentati gli ID degli agenti—rappresentando quindi le generazioni in ordine cronologico—a cui viene fatto corrispondere sull'asse y il valore registrato dall'agente per la misura osservata.

aumentando quindi la probabilità di generare individui con genoma sfavorevole.

Efficacia dell'algoritmo. La Figura 4.2 rappresenta l'andamento dei geni regolatori delle funzioni vitali degli agenti in caso di applicazione del solo GA. Osservando qualitativamente i grafici si possono notare, a meno di instabilità anche abbastanza evidenti, delle tendenze generali: per il gene regolatore della velocità di movimento sono prediletti valori medio-alti, mentre i geni λ_h, λ_t e λ_l tendono a valori bassi creando di conseguenza una tendenza crescente nel tempo di vita atteso.

Le instabilità osservate possono dipendere da diversi fattori. Il primo è sicuramente l'aggressività della politica di mutazione implementata dal GA, che prevede che circa l'85% dei genomi prodotti sia soggetto a mutazioni di qualche natura. Se da un lato ciò favorisce l'esplorazione dello spazio dei genomi ammissibili, dall'altro rende sicuramente più instabile il processo di convergenza, dal momento che le mutazioni possono colpire anche geni già ottimizzati o prossimi al valore ottimale.

A contribuire ulteriormente all'instabilità del processo di convergenza è la definizione della funzione fitness, che premia gli individui a seconda del loro tempo di vita effettivo, indipendentemente dal tempo di vita atteso. Se da un lato, in assenza del contributo del DRL, gli agenti con tempo di vita atteso maggiore hanno più possibilità di esplorare casualmente l'ambiente e di conseguenza hanno maggiore probabilità di attuare una strategia che gli permetta di estendere ulteriormente il proprio tempo di vita, possono esserci degli individui particolarmente longevi anche a discapito di un genoma sfavorevole. In tal senso le instabilità nella convergenza dei valori del tempo di vita atteso sono giustificate dal fatto che l'ottimizzazione dei geni regolatori dei parametri vitali sia una conseguenza dell'applicazione del solo GA senza contributo del DRL piuttosto che il suo obiettivo primario.

Per quantificare l'efficacia del processo evolutivo è però necessario analizzare i valori osservati per il tempo di vita atteso paragonandoli a quelli raccolti nell'analisi dell'efficacia di base. Il valore minimo osservato è di circa 60.39 secondi—quindi sostanzialmente invariato rispetto al dato senza ottimizzazione—mentre il valore massimo osservato cresce fino a 325 secondi. L'intervallo dei tempi di vita attesi osservati è quindi centrato attorno a un valore di circa 192.70 secondi, a fronte

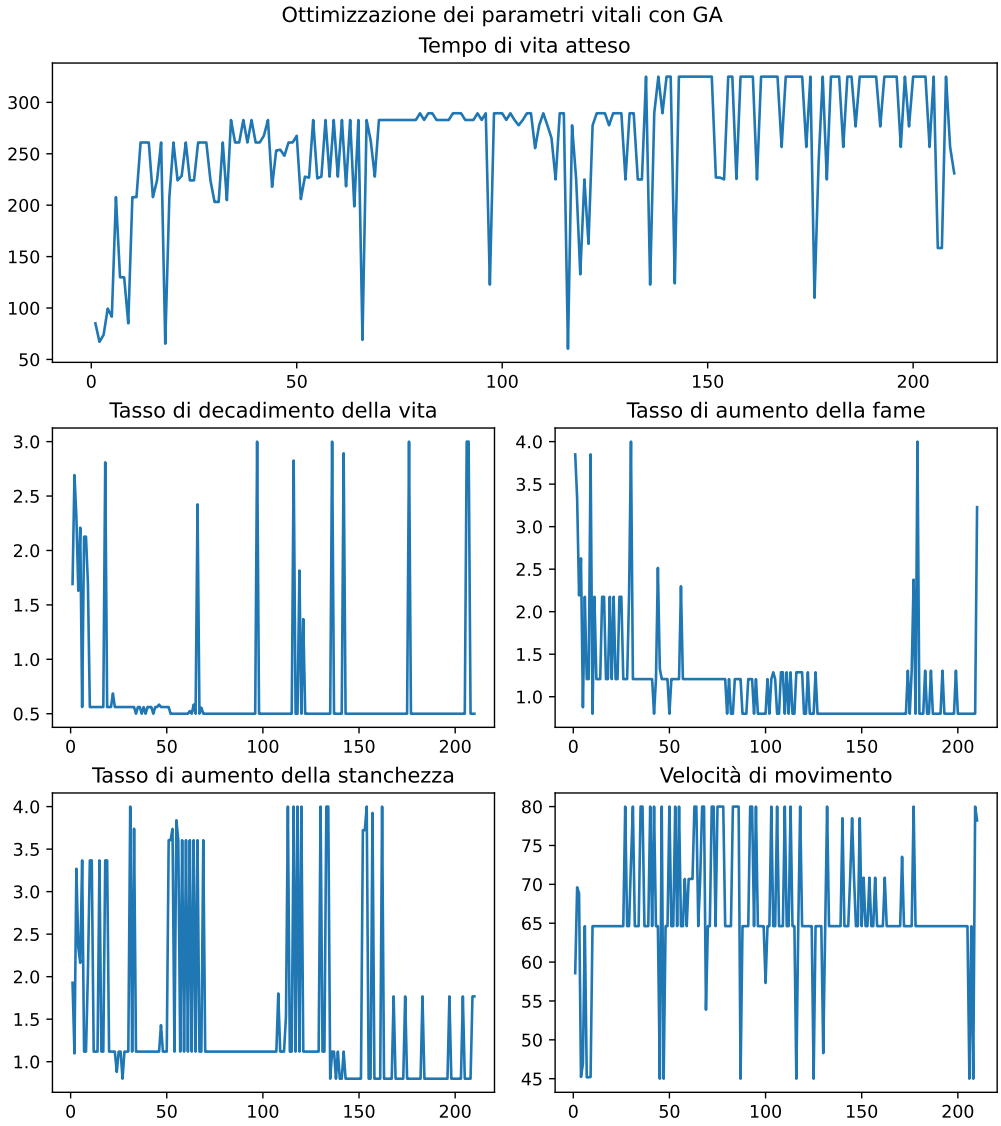

Figura 4.2: Grafici di valutazione dell'andamento dei geni regolatori dei parametri vitali e del tempo di vita atteso lungo le generazioni applicando il contributo di ottimizzazione dell'Algoritmo Genetico. Sull'asse x sono rappresentati gli ID degli agenti—rappresentando quindi le generazioni in ordine cronologico—a cui viene fatto corrispondere sull'asse y il valore registrato dall'agente per la misura osservata.

di una media pari a circa 263.09 secondi e quindi di circa 2.38 volte più grande rispetto a quella osservata nello scenario non ottimizzato.

Il primo risultato del GA, che emerge in maniera immediata osservando il valore medio rispetto al centro dell'intervallo dei valori osservati, è quindi l'annullamento del condizionamento verso il basso del tempo di vita atteso osservato in precedenza. La selezione educata dei genomi da trasmettere—contrapposta all'estrazione casuale dei valori dei geni—permette infatti di selezionare individui con configurazioni genetiche tali da annullare l'effetto negativo che la definizione stessa del tempo di vita atteso ha sul valore osservato, producendo anzi una forma di condizionamento verso l'alto per cui la media dei valori osservati è maggiore rispetto al centro dell'intervallo di distribuzione degli stessi.

Analizzando invece il valore massimo osservato si nota che l'ottimizzazione evolutiva ha raggiunto l'estremo superiore di ammissibilità per il tempo di vita atteso, mai raggiunto nell'osservazione non ottimizzata. In particolare il primo individuo con genoma ottimale è stato l'agente 134, appartenente alla ventesima generazione, e da quel momento sono stati generati un totale di 52 individui con lo stesso tempo di vita atteso. Ciò significa che, indipendentemente dai genomi della popolazione iniziale, il processo evolutivo guidato dal GA è stato in grado di selezionare, trasmettere e mutare i genomi in maniera efficace fino al raggiungimento, nelle fasi avanzate dell'evoluzione, della configurazione genetica ottimale. Il fatto che poi 52 degli ultimi 77 individui generati—quindi circa il 67.53% delle ultime 11 generazioni—abbia mantenuto il genoma ottimale nonostante l'aggressività della politica di mutazione è ulteriore indice dell'efficacia del GA per quanto riguarda il mantenimento delle caratteristiche dominanti.

Per valutare l'andamento del processo evolutivo, si possono poi paragonare le medie del tempo di vita atteso per la prima e l'ultima generazione osservate. La media per la prima generazione è di circa 107.82 secondi, paragonabile a quella complessivamente ottenuta nell'osservazione non ottimizzata, e deriva dalla generazione casuale dei genomi per la popolazione di base. Diversamente la media del tempo di vita atteso per la trentesima generazione è di circa 244.40 secondi, prossima al valore massimo osservato in assenza di ottimizzazione. Ciò significa che il processo evolutivo è riuscito a superare ampiamente i risultati del modello casuale, dimostrando che l'ottimizzazione dei genomi lungo le generazioni è garantita

indipendentemente dalle caratteristiche della popolazione iniziale.

Criticità dell'algoritmo. Seppure il GA si sia dimostrato nel complesso efficace, emergono alcune evidenti criticità prevalentemente legate alla stabilità dello stesso. In primo luogo si riprende l'ultima analisi svolta sull'andamento dei valori medi del tempo di vita atteso fra generazioni e si nota che il valore osservato per la trentesima generazione (244.40 secondi), seppur indicativo di un processo evolutivo generalmente efficace, è lievemente inferiore alla media complessiva (263.09 secondi). Tale risultato potrebbe essere indice di inefficacia del processo evolutivo, dal momento che la regressione delle ultime generazioni potrebbe indicare che l'algoritmo non è in grado di preservare i risultati di ottimizzazione raggiunti. Un ulteriore indicatore in tal senso è che l'individuo peggiore per tempo di vita atteso è stato l'agente 116, appartenente alla diciassettesima generazione.

La causa di tali problemi risiede probabilmente nella politica di mutazione del GA, che colpisce la gran parte dei genomi. In tal senso il processo di ottimizzazione continua a cui è sottoposta la popolazione—il processo evolutivo guidato dal GA non è infatti mai interrotto, indipendentemente dalla qualità dei genomi ottenuti—è naturalmente soggetto a periodiche flessioni di qualità dovute alla continua esplorazione casuale dello spazio dei genomi ammissibili che agisce parallelamente alla selezione degli individui più prestanti. La valutazione dell'efficacia del GA deve quindi prendere atto delle naturali oscillazioni della qualità e soffermarsi sulla valutazione della capacità del GA stesso di superare i passi di esplorazione dannosi per la qualità della popolazione. In particolare si sottolinea che il valore minimo osservato per il tempo di vita atteso non è mai stato replicato dopo l'agente 116—indice del fatto che il genoma sfavorevole è stato scartato appena dopo la sua esplorazione—e che il valore medio per il tempo di vita atteso delle 5 generazioni prima dell'ultima è di circa 303.61 secondi—quindi ampiamente superiore al valore medio complessivo, confermando la tendenza positiva del processo evolutivo indipendentemente dalla valutazione di qualità dei genomi della sola trentesima generazione.

4.2 Algoritmo di apprendimento

Modalità di valutazione. La valutazione dell'algoritmo di apprendimento richiede di quantificare l'impatto che l'attivazione del modello di DRL ha sulla capacità degli agenti di sopravvivere all'interno dell'ambiente di gioco. Per valutare il solo apporto del modello di apprendimento le osservazioni vengono svolte disattivando il contributo di ottimizzazione del GA, e quindi determinando in maniera casuale il genoma di ciascun agente.

Per questo motivo l'efficacia ottenuta in questa fase di analisi è da intendersi come una valutazione media della qualità dell'algoritmo di DRL all'interno degli intervalli di ammissibilità dei geni regolatori dei parametri dell'apprendimento, piuttosto che come una misura assoluta della qualità ideale dell'algoritmo. Per ottenere una misura di questo tipo sarebbe infatti necessario svolgere una ricerca—quantomeno approssimativa—dei parametri dell'apprendimento per fissarne il valore.

Metriche di valutazione. L'obiettivo principale del modello di apprendimento è quello di permettere a ogni singolo agente di sviluppare una strategia di sopravvivenza il più possibile efficace all'interno del mondo di gioco. La misura più direttamente influenzata dalla strategia di comportamento degli agenti è quindi il tempo di vita effettivo, che risulta essere direttamente proporzionale alla qualità della strategia di comportamento appresa.

La metrica del tempo di vita effettivo, usata dalla funzione fitness per valutare la qualità assoluta del singolo individuo, è però poco adatta alla valutazione del solo modello di DRL. Ciò perché il valore del tempo di vita effettivo dipende strettamente dal valore del tempo di vita atteso, che a sua volta dipende dai geni regolatori delle funzioni vitali dell'agente. Se dal punto di vista evolutivo tenere conto, seppur indirettamente, del tempo di vita atteso dell'individuo può essere funzionale alla progressione della popolazione, la valutazione dell'algoritmo di apprendimento richiede di definire una metrica differente in grado di superare tale dipendenza.

Si introduce quindi il *rapporto vita-aspettativa*, che riassume in un valore adimensionale la durata effettiva della vita dell'agente rispetto al suo tempo di vita

	$E(T_l)$	T_l	$\frac{T_l}{E(T_l)}$
Media	110.39	137.88	1.22
Minimo	60.10	60.12	1.00
Primo quartile	84.01	85.39	1.00
Mediana	98.76	107.23	1.00
Terzo quartile	132.53	147.04	1.00
Massimo	247.96	1089.89	6.39

Tabella 4.2: Riassunto dei valori medio, minimo e dei 4 quartili per il tempo di vita atteso, il tempo di vita effettivo e il rapporto vita-aspettativa ottenuti dall'analisi delle osservazioni svolte senza i contributi né dell'Algoritmo Genetico né del modello di Deep Reinforcement Learning.

atteso come definito dall'Equazione (4.1). In questo modo, paragonando il tempo di vita effettivo al tempo di vita atteso in caso di inattività, si può quantificare direttamente l'impatto che la strategia comportamentale applicata ha sulle capacità di sopravvivenza dell'agente all'interno del mondo di gioco. Si definisce quindi il rappporto vita-aspettativa come segue:

$$\frac{T_l}{E(T_l)} \tag{4.2}$$

Efficacia di base. La Tabella 4.2 contiene una rappresentazione schematica della media e della distribuzione dei valori osservati per il tempo di vita atteso, il tempo di vita effettivo e il rapporto vita-aspettativa nel caso sia disattivato il contributo di ottimizzazione del DRL.

Il primo dato da analizzare per quantificare la qualità delle strategie messe in atto dagli agenti è il valore medio per il rapporto vita-aspettativa. Esso, pur non essendo di per sé esaustivo, rappresenta infatti una valutazione media dell'efficacia delle strategie sperimentate da tutti gli agenti, e si prevede che rifletta più o meno direttamente la qualità dell'algoritmo di controllo proposto. Il valore di efficacia medio osservato per il modello di comportamento casuale è di circa 1.22.

L'apparente efficacia della strategia di controllo casuale viene però immediatamente ridimensionata se si analizza la distribuzione dei valori osservati per il rapporto vita-aspettativa. Dei 210 agenti osservati infatti solo 39—corrispondenti

	$E(T_l)$	T_l	$\frac{T_l}{E(T_l)}$
Media	107.05	169.37	1.52
Minimo	59.34	59.45	1.00
Primo quartile	77.42	85.68	1.00
Mediana	94.57	119.56	1.09
Terzo quartile	126.79	199.24	1.75
Massimo	238.90	916.61	5.33

Tabella 4.3: Riassunto dei valori medio, minimo e dei 4 quartili per il tempo di vita atteso, il tempo di vita effettivo e il rapporto vita-aspettativa ottenuti dall'analisi delle osservazioni svolte con il solo contributo del modello di Deep Reinforcement Learning.

quindi a circa il 18.57% del totale—hanno registrato un rapporto vita-aspettativa superiore a 1.01. Ciò significa che mediamente meno di un agente su 5 è riuscito a estendere il proprio tempo di vita anche solo dell'1% rispetto alle previsioni in caso di completa inattività.

Dall'altra parte la media del rapporto vita-aspettativa è pesantemente impattata dalla presenza di outliers pronunciati come il valore massimo, pari a circa 6.39. La presenza di individui così longevi è da attribuire a diversi fattori, in cui sicuramente giocano un ruolo fondamentale la semplicità del mondo e della dinamica di gioco. La struttura della mappa, bidimensionale e senza ostacoli intermedi a meno di collisioni con altri agenti, la rende facilmente esplorabile specialmente agli individui geneticamente più veloci, rendendo relativamente probabile il raggiungimento delle aree di ristoro anche ad agenti guidati da un modello di comportamento casuale. Allo stesso modo l'interazione permette all'agente, quando si trova all'interno di un punto di ristoro, di mitigare i propri bisogni, e la limitatezza del set di azioni possibili sommata alla facilità di esplorazione del mondo di gioco rende relativamente probabile l'attuazione di una strategia complessivamente efficace anche da parte di un modello di controllo casuale.

Efficacia dell'algoritmo. La Tabella 4.3 contiene invece una rappresentazione schematica della media e della distribuzione dei valori osservati per il tempo di vita atteso, il tempo di vita effettivo e il rapporto vita-aspettativa in caso di applicazione del solo modello di DRL.

Il primo dato analizzato è il valore medio per il rapporto vita-aspettativa, che in caso di applicazione del DRL misura circa 1.52. Ciò significa che il tempo di vita effettivo viene esteso mediamente del 52% rispetto al tempo di vita atteso, incremento più che doppio rispetto a quello osservato con il modello di comportamento casuale, che estendeva mediamente il tempo di vita di ciascun agente solo del 22%.

La distribuzione dei dati per il rapporto vita-aspettativa osservata applicando il DRL conferma la migliore efficacia del modello di apprendimento rispetto al modello di controllo casuale. Il numero di agenti in grado di estendere la durata della propria vita per più dell'1% cresce infatti fino a 121, più che triplicando il risultato precedente.

Tale miglioramento era deducibile anche dall'osservazione dei quartili della distribuzione per il valore del rapporto vita-aspettativa: la mediana—uguale a circa 1.09—indica già un netto miglioramento rispetto ai risultati ottenuti dal modello casuale, dal momento che indica come la metà degli agenti sia riuscita ad estendere il proprio tempo di vita di almeno il 9% rispetto al tempo di vita atteso. Il terzo quartile poi è quello in cui emerge la differenza più marcata: mentre nel caso del modello di controllo casuale vale 1—indicando quindi che il 75% degli agenti non è stato in grado di estendere il proprio tempo di vita—il contributo di ottimizzazione del DRL porta lo stesso valore a crescere fino a 1.75, escludendo quindi di fatto che la media ottenuta sia falsata dalla presenza di outliers.

Criticità dell'algoritmo. Sebbene i risultati ottenuti dal modello di DRL siano nettamente migliori di quelli ottenuti dal modello casuale, la quantità di agenti che non sono stati in grado di apprendere una strategia in alcun modo efficace—in totale 89 individui per cui il rapporto vita-aspettativa è minore di 1.01—solleva inevitabilmente delle perplessità relativamente all'effettiva efficacia del modello di apprendimento.

In questo senso è però fondamentale sottolineare che le valutazioni sono state svolte in assenza del contributo di ottimizzazione del GA, e che l'efficacia dell'algoritmo di DRL proposto dipende direttamente dalla capacità che l'agente ha di esplorare l'ambiente in cui si trova. Pertanto è credibile pensare che l'estrazione casuale dei valori dei geni possa aver reso meno efficace il processo di apprendi-

mento per quegli agenti a cui siano stati assegnati genomi sfavorevoli. Per trovare riscontro di questa possibilità si analizza il valore che più di tutti influenza la capacità dell'agente di esplorare lo spazio circostante, ovvero il tempo di vita atteso. Un agente con un lungo tempo di vita atteso ha infatti la possibilità di sperimentare azioni per un lasso di tempo più lungo prima di incorrere in condizioni critiche di salute ed eventualmente poi nella morte, rendendo inevitabilmente più agevole l'esplorazione dell'ambiente.

Analizzando gli 89 agenti che non sono riusciti ad apprendere una strategia efficace, il valore medio per il tempo di vita atteso vale circa 98.79 secondi, ampiamente sotto la media del campione osservato. Al contrario il valore medio per il tempo di vita atteso dei 5 agenti più performanti—che registrano un rapporto vita-aspettativa medio di circa 4.65—misura circa 149.76 secondi, superando ancora più abbondantemente il valore medio complessivo.

Tale analisi sembra confermare che ci possa essere una correlazione, seppur indiretta, fra i valori dei geni regolatori dei parametri vitali e la qualità della strategia appresa, e lascia intendere che la valutazione del modello di DRL con genomi casuali sia condizionata da una forte instabilità dei valori dei parametri—sia di apprendimento che di regolazione delle interazioni ambientali—che porta a un degrado dell'efficacia generale.

4.3 Approccio combinato

Modalità di valutazione. La valutazione dei risultati ottenuti applicando contemporaneamente le ottimizzazioni del GA e del DRL ha come obiettivo quello di quantificare l'efficacia complessiva del modello proposto. Per farlo però si prendono, come dati di efficacia di base con cui paragonare i valori osservati, i risultati ottenuti dall'applicazione dei due algoritmi studiati in maniera indipendente piuttosto che i risultati ottenuti dal modello completamente casuale.

Tale scelta è stata fatta con l'obiettivo di formulare giudizi più approfonditi e significativi sul funzionamento del modello integrato, individuando da una parte possibili problemi introdotti dall'integrazione dei due metodi di ottimizzazione, e sottolineando dall'altra come l'applicazione contemporanea dei due algoritmi

permetta di superare i principali problemi emersi negli scenari di applicazione indipendente precedentemente analizzati.

Metriche di valutazione. La valutazione ha come obiettivo principale la misura dell'impatto che l'ottimizzazione tramite GA ha sulle prestazioni del modello di DRL. La metrica principale usata per la valutazione del modello è quindi, riprendendo la definizione presentata nella Sezione 4.2, il rapporto vita-aspettativa.

Parallelamente si sfruttano, per facilitare la valutazione degli effetti del GA sul sistema, dei grafici di andamento di alcuni valori fondamentali riprendendo l'approccio usato nella Sezione 4.1. Al contrario però dell'analisi precedente non è possibile sfruttare il tempo di vita atteso come metrica quantitativa della qualità del GA, dal momento che il processo evolutivo coinvolge l'intero genoma e non soltanto la regolazione dei parametri vitali.

Efficacia di base. La valutazione dell'efficacia di base deve quindi tenere conto di due componenti distinte. La prima, più complessa da rapportare alla valutazione congiunta, consiste nei risultati ottenuti dall'applicazione indipendente del GA. La seconda, più direttamente rapportabile alle prestazioni del modello completo, è quella che invece deriva dall'osservazione delle prestazioni del solo modello di DRL.

La Figura 4.3 riprende la stessa visualizzazione dell'andamento del tempo di vita atteso della Figura 4.2, e la affianca alla visualizzazione dell'andamento lungo le generazioni dei valori del tempo di vita effettivo e del rapporto vita-aspettativa. Mentre è osservabile una tendenza verso valori elevati per il tempo di vita atteso, già motivata nella Sezione 4.1, gli altri due grafici di andamento non presentano tendenze evidenti. Ciò accade perché gli agenti sono controllati da un modello di comportamento casuale, che non è quindi in grado di sfruttare le condizioni favorevoli offerte dall'ottimizzazione del genoma per aumentare le proprie probabilità di sopravvivenza.

La Tabella 4.4 presenta invece una sintesi dei dati analoga a quella proposta nella Sezione 4.2, stavolta applicata ai dati di efficacia delle strategie comportamentali raccolti applicando la sola ottimizzazione tramite GA. È particolarmente interessante in questo senso notare che il valore medio osservato per il rapporto

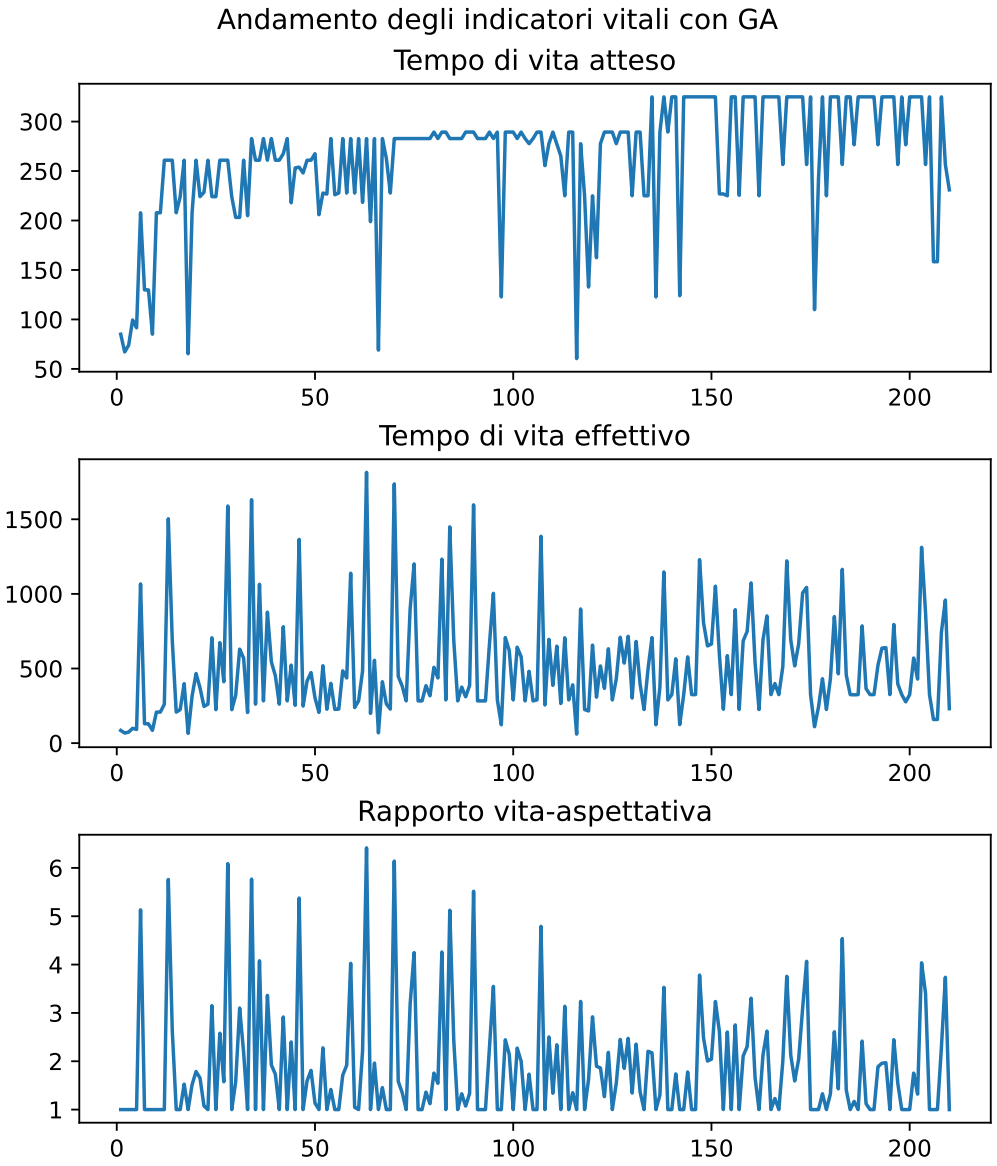

Figura 4.3: Grafici di valutazione dell'andamento del tempo di vita atteso, del tempo di vita effettivo e del rapporto vita-aspettativa lungo le generazioni applicando il solo contributo di ottimizzazione dell'Algoritmo Genetico. Sull'asse x sono rappresentati gli ID degli agenti—rappresentando quindi le generazioni in ordine cronologico—a cui viene fatto corrispondere sull'asse y il valore registrato dall'agente per la misura osservata.

	$E(T_l)$	T_l	$\frac{T_l}{E(T_l)}$
Media	263.09	510.79	1.89
Minimo	60.39	60.43	1.00
Primo quartile	227.74	282.86	1.00
Mediana	282.84	396.07	1.41
Terzo quartile	289.39	664.52	2.30
Massimo	325.00	1814.45	6.42

Tabella 4.4: Riassunto dei valori medio, minimo e dei 4 quartili per il tempo di vita atteso, il tempo di vita effettivo e il rapporto vita-aspettativa ottenuti dall'analisi delle osservazioni svolte con il contributo di ottimizzazione dell'Algoritmo Genetico.

vita-aspettativa è maggiore rispetto a quello osservato in caso di applicazione del solo DRL, e anche i dati di distribuzione sembrano confermare la maggiore efficacia del modello.

Analizzando i quartili dei valori di efficacia del modello ottimizzato tramite GA e confrontandoli con quelli riportati nella Tabella 4.3 ottenuti dal modello di DRL si vede infatti come in ogni fascia della distribuzione il modello basato su GA registri una prestazione migliore rispetto a quello basato su DRL. La differenza più marcata fra i dati raccolti dai due modelli risiede però nei valori del tempo di vita atteso: già dal primo quartile il modello ottimizzato tramite GA riporta infatti dei dati di tempo di vita atteso prossimi al valore massimo osservato dal modello basato su DRL.

Ciò sembrerebbe rafforzare le deduzioni già introdotte nella Sezione 4.2, secondo cui all'aumentare del tempo di vita atteso degli individui aumenta anche il loro rendimento comportamentale indipendentemente dal modello di controllo attivato. Estendere infatti il tempo di vita atteso significa permettere all'agente una maggiore esplorazione dell'ambiente che lo porta, in caso di ottimizzazione tramite DRL, a raffinare le strategie apprese oppure, in caso di applicazione della strateiga casuale, a incontrare sequenze di azioni ottimali, specialmente in un mondo con una struttura così semplice.

Come valore di efficacia del sistema basato su DRL si riprende invece soltanto il contenuto della Tabella 4.3. Sarebbe infatti poco signficativo rappresentare grafici di tendenza dei valori osservati indipendentemente dalla metrica analizzata,

	$E(T_l)$	T_l	$\frac{T_l}{E(T_l)}$
Media	176.29	442.94	2.37
Minimo	64.92	66.20	1.00
Primo quartile	88.28	173.78	1.53
Mediana	225.00	382.36	2.27
Terzo quartile	238.39	659.01	3.05
Massimo	325.00	1402.64	6.23

Tabella 4.5: Riassunto dei valori medio, minimo e dei 4 quartili per il tempo di vita atteso, il tempo di vita effettivo e il rapporto vita-aspettativa ottenuti dall'analisi delle osservazioni svolte con il contributo sia del Deep Reinforcement Learning che dell'Algoritmo Genetico.

dal momento che il contributo di ottimizzazione del GA è disattivato e quindi i parametri di ogni agente sono determinati in modo casuale.

Efficacia del modello di apprendimento. La Tabella 4.5 riassume i dati di efficacia delle strategie comportamentali prodotte dal modello di DRL con ottimizzazione evolutiva dei parametri tramite GA. Per quantificare la qualità del modello combinato di apprendimento si osservano principalmente il valore medio e la distribuzione per il dato del rapporto vita-aspettativa.

Il valore medio per il rapporto vita-aspettativa, pari a circa 2.37, indica che gli agenti guidati da DRL ottimizzato tramite GA sono stati mediamente in grado di estendere il proprio tempo di vita del 137% rispetto alle previsioni di aspettativa ottenute studiando il genoma. Tale risultato indica un livello di efficacia di circa 2.63 volte superiore rispetto a quello ottenuto dal solo modello di DRL, grazie a cui gli agenti sono stati in grado di estendere mediamente il proprio tempo di vita solo del 52%.

Analizzando poi i dati di distribuzione per il rapporto vita-aspettativa, l'efficacia del modello combinato risulta ancora più evidente. Se nella Sezione 4.2 si era infatti citata come possibile criticità dell'algoritmo la presenza di 89 agenti—che rappresentano circa il 42.38% della popolazione osservata di 210 individui—incapaci di apprendere una strategia in grado di estendere il tempo di vita effettivo oltre al 101% rispetto al tempo di vita atteso, tale problema viene quasi completamente superato dal modello combinato di applicazione del DRL e del GA, grazie a cui

solo 18 agenti—circa l'8.57% della popolazione osservata di 210 individui—hanno registrato un valore inferiore a 1.01 per il rapporto vita-aspettativa.

L'osservazione dei quartili della distribuzione per il rapporto vita-aspettativa conferma poi ulteriormente la qualità del modello combinato rispetto al solo modello di DRL. Il primo quartile ottenuto applicando l'approccio combinato di ottimizzazione è infatti pari a circa 1.53, valore già superiore alla media complessiva ottenuta dal solo modello di DRL. Ciò significa che il 75% degli agenti guidati da DRL ottimizzato tramite GA ha ottenuto un valore di efficacia superiore alla media degli agenti guidati da DRL con genoma casuale.

Efficacia dell'ottimizzazione dei parametri vitali. Tali risultati permettono di affermare che l'ottimizzazione guidata dal GA ha un impatto positivo sull'efficacia del modello di apprendimento. Si vogliono quindi analizzare le tendenze di alcuni valori fondamentali per giustificare il risultato osservato. In primo luogo si studiano i valori ottenuti per il tempo di vita atteso al fine di analizzare l'effetto che l'ottimizzazione dei parametri vitali degli agenti ha sul loro processo di apprendimento. A fronte di un valore medio per il tempo di vita atteso di circa 176.29 secondi—maggiore del terzo quartile di distribuzione osservato dal modello basato solo su DRL, pari a circa 126.79 secondi—e basandosi sulle osservazioni già svolte nella Sezione 4.2, è infatti naturale indagare una possibile forma di correlazione fra il tempo di vita atteso e l'efficacia della strategia appresa.

Nella Figura 4.4 vengono quindi presentati i valori del tempo di vita atteso, del tempo di vita effettivo e del rapporto vita-aspettativa per ogni agente della popolazione studiata. Osservando l'andamento della misura del tempo di vita atteso si nota una tendenza abbastanza chiara verso valori medio-alti—fino a raggiungere un massimo di 325, pari all'estremo superiore di ammissibilità—nonostante la scarsa qualità dei genomi della popolazione iniziale, sottolineando ulteriormente la robustezza del GA proposto. È interessante però notare come il valore ottimale del tempo di vita, dopo essere stato raggiunto, venga in realtà rapidamente abbandonato per convergere nuovamente verso valori lievemente inferiori. Osservando però il grafico di tendenza del rapporto vita-aspettativa si nota, in corrispondenza dell'aumento del tempo di vita atteso fino all'estremo superiore, una flessione del rendimento del modello di apprendimento. È quindi verosimile concludere che il

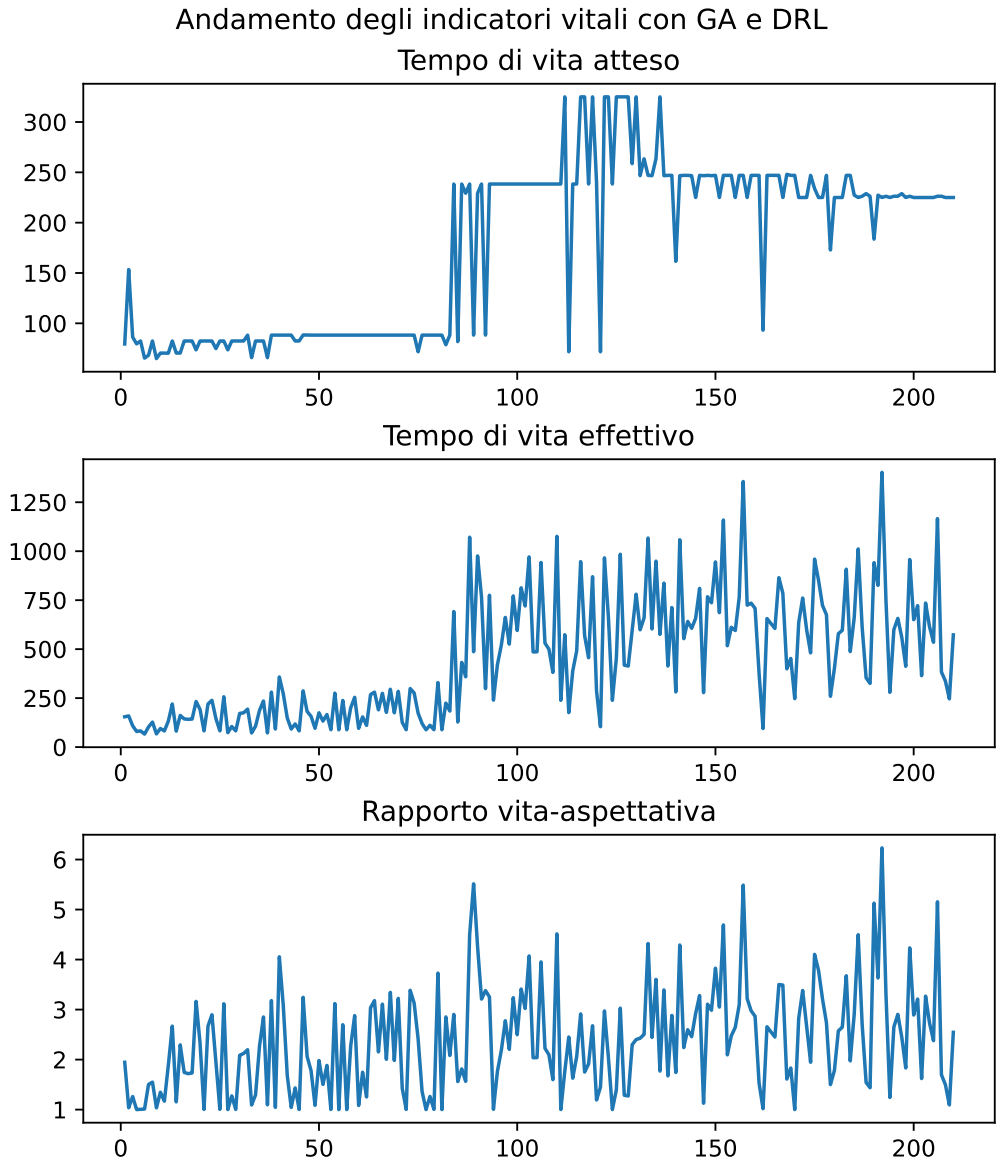

Figura 4.4: Grafici di valutazione dell'andamento del tempo di vita atteso, del tempo di vita effettivo e del rapporto vita-aspettativa lungo le generazioni applicando i contributi di ottimizzazione sia del Deep Reinforcement Learning che dell'Algoritmo Genetico. Sull'asse x sono rappresentati gli ID degli agenti—rappresentando quindi le generazioni in ordine cronologico—a cui viene fatto corrispondere sull'asse y il valore registrato dall'agente per la misura osservata.

valore ottimale dei geni regolatori dei parametri vitali sia in realtà stato scartato per prediligere genomi più favorevoli al processo di apprendimento, dimostrando come i due processi di ottimizzazione si possano influenzare reciprocamente.

Studiando invece l'andamento del rapporto vita-aspettativa, metrica di valutazione del processo di apprendimento, non si notano tendenze evidenti. Dall'altra parte però si nota una correlazione con i valori del tempo di vita atteso. Attorno alla dodicesima generazione—composta dagli agenti numerati fra il 78 e l'84—si nota infatti un repentino aumento del tempo di vita atteso, che da valori attorno al 100 passa a valori attorno al 250. In corrispondenza di questo aumento, probabilmente causato da una mutazione favorevole, si osserva un cambiamento nella distribuzione dei valori del rapporto vita-aspettativa: i picchi negativi diminuiscono per frequenza, mentre al contrario i picchi positivi crescono sia per frequenza che per valore. Si analizzano quindi i valori del rapporto vita-aspettativa e del tempo di vita atteso delle prime 11 generazioni e successivamente delle generazioni dalla tredicesima in poi con l'obiettivo di verificare la correlazione.

Le prime 11 generazioni, composte da 77 individui, registrano un tempo di vita atteso medio di circa 83.92 secondi, di poco inferiore alla metà del valore medio complessivo. Considerando poi il rapporto vita-aspettativa, si osservano un massimo di circa 4.06 e un valore medio di circa 1.91, abbondantemente al di sotto della media dell'intero campione—ma comunque meglio rispetto alla media complessiva ottenuta applicando il solo modello di DRL, presumibilmente a causa dell'ottimizzazione dei parametri di apprendimento svolta dal GA. Fra le prime 11 generazioni si osservano inoltre, riprendendo le valutazioni svolte in precedenza, 12 individui con rapporto vita-aspettativa minore di 1.01, corrispondenti a circa il 15.58% della popolazione osservata.

Le generazioni che vanno dalla tredicesima alla trentesima, composte da 126 individui, registrano invece un tempo di vita atteso medio di circa 236.50 secondi, pari a circa 1.34 volte il valore medio complessivo. Considerando poi anche in questo caso il rapporto vita-aspettativa, si osservano un massimo—corrispondente al massimo complessivo per il campione—di circa 6.23 e un valore medio di circa 2.66. Si osservano in queste generazioni invece soltanto 4 agenti con rapporto vita-aspettativa minore di 1.01—pari quindi a circa il 3.17% della popolazione osservata—e dall'altra parte 15 agenti—corrispondenti a circa l'11.90% della popo-

lazione studiata—con rapporto vita-aspettativa maggiore di 4.06, valore massimo osservato nelle prime 11 generazioni. I dati confermano quindi la correlazione fra il tempo di vita atteso e il rapporto vita-aspettativa, sottolineando come l'ottimizzazione tramite GA dei geni regolatori dei parametri vitali sia in grado di aumentare l'efficacia del modello di apprendimento pur non influenzando direttamente i suoi parametri, ma agendo piuttosto sulle modalità di interazione fra agente e ambiente.

Efficacia dell'ottimizzazione dei parametri di apprendimento. La Figura 4.5 contiene i grafici di andamento dei principali geni regolatori dei parametri della rete Attention, e la Figura 4.6 contiene le rappresentazioni corrispondenti per la rete Reason di controllo delle azioni degli agenti. Già dalla sola osservazione qualitativa dei grafici si notano forti instabilità che impediscono di individuare tendenze evidenti. Ciò può essere dovuto alla formulazione proposta per il GA, caratterizzato da una politica di mutazione particolarmente aggressiva, e dal fatto che l'ottimizzazione agisce parallelamente anche sui geni regolatori dei parametri vitali degli agenti. Evolvere primariamente i geni regolatori dei parametri vitali degli individui significa infatti sostanzialmente estenderne il tempo di vita, metrica predominante per la valutazione del valore di fitness. Tale effetto è rafforzato poi dalla correlazione osservata fra tempo di vita atteso ed efficacia del processo di apprendimento, che rende quindi ulteriormente conveniente l'ottimizzazione dei geni regolatori dei parametri vitali.

A contribuire ulteriormente all'instabilità del processo di ottimizzazione dei parametri dell'apprendimento può essere anche la definizione degli intervalli di ammissibilità degli stessi. Si sono infatti proposti intervalli relativamente ristretti e centrati attorno a valori reputati ottimali in altri studi basati su modelli di DRL analoghi. Tale approccio, che da un lato aumenta certamente l'efficacia complessiva del sistema, potrebbe impedire però una convergenza marcata nel caso in cui tutte—o quasi—le configurazioni di valori ammissibili per i parametri dell'apprendimento avessero prestazioni di fatto equivalenti. In tal caso infatti, indipendentemente dai valori dei geni regolatori dei parametri di apprendimento, le prestazioni degli agenti resterebbero sostanzialmente invariate e l'unico contributo di ottimizzazione efficace—a maggior ragione alla luce della correlazione fra tempo di vita atteso ed efficacia dell'apprendimento sottolineata in precedenza—diventerebbe

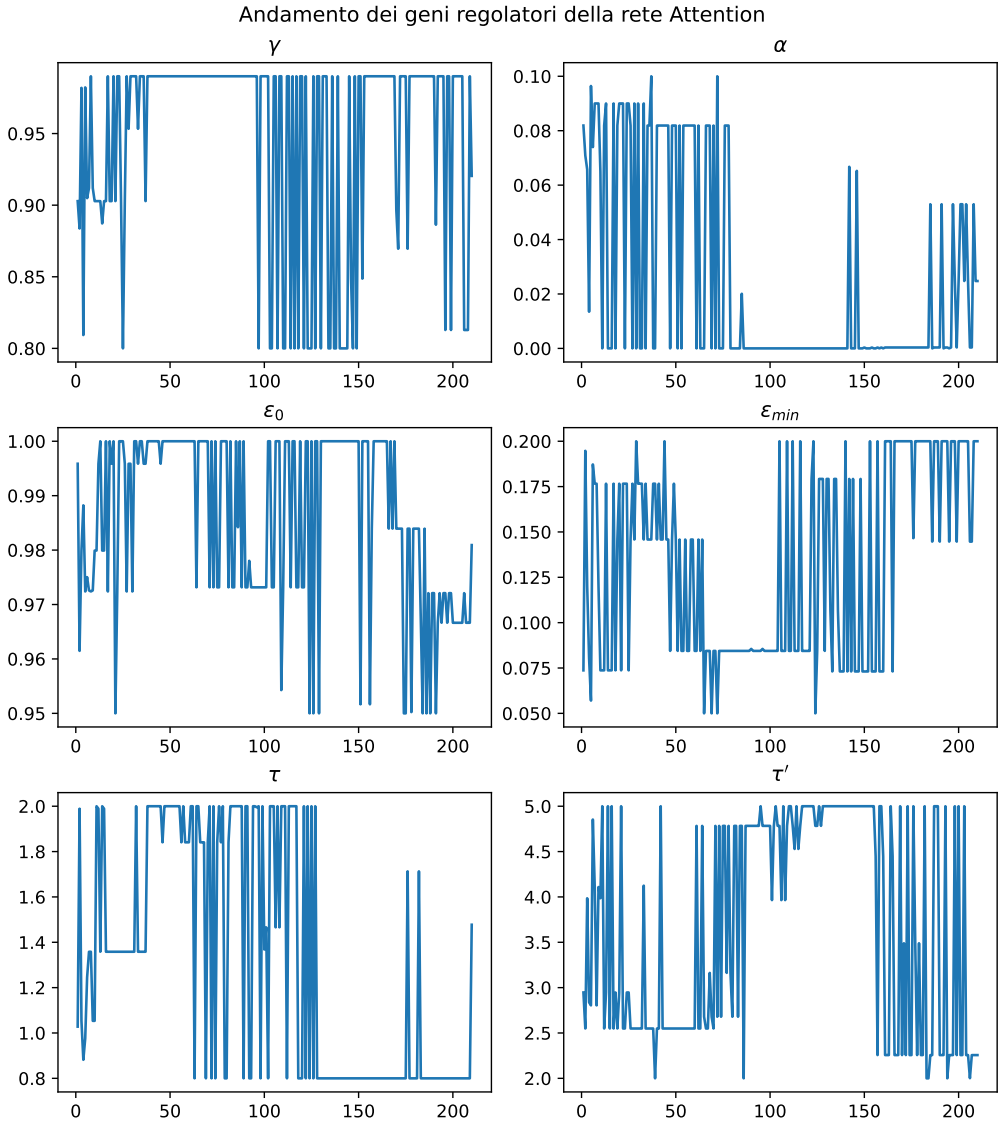

Figura 4.5: Grafici di tendenza per i geni regolatori del discount factor γ, del learning rate α, del tasso di esplorazione iniziale ε_0, del tasso di esplorazione minimo ε_{min}, e dei periodi di aggiornamento τ della rete principale e τ' della rete Target ottimizzati tramite Algoritmo Genetico per il lobo Attention. Sull'asse x sono rappresentati gli ID degli agenti—rappresentando quindi le generazioni in ordine cronologico—a cui viene fatto corrispondere sull'asse y il valore registrato dall'agente per la misura osservata.

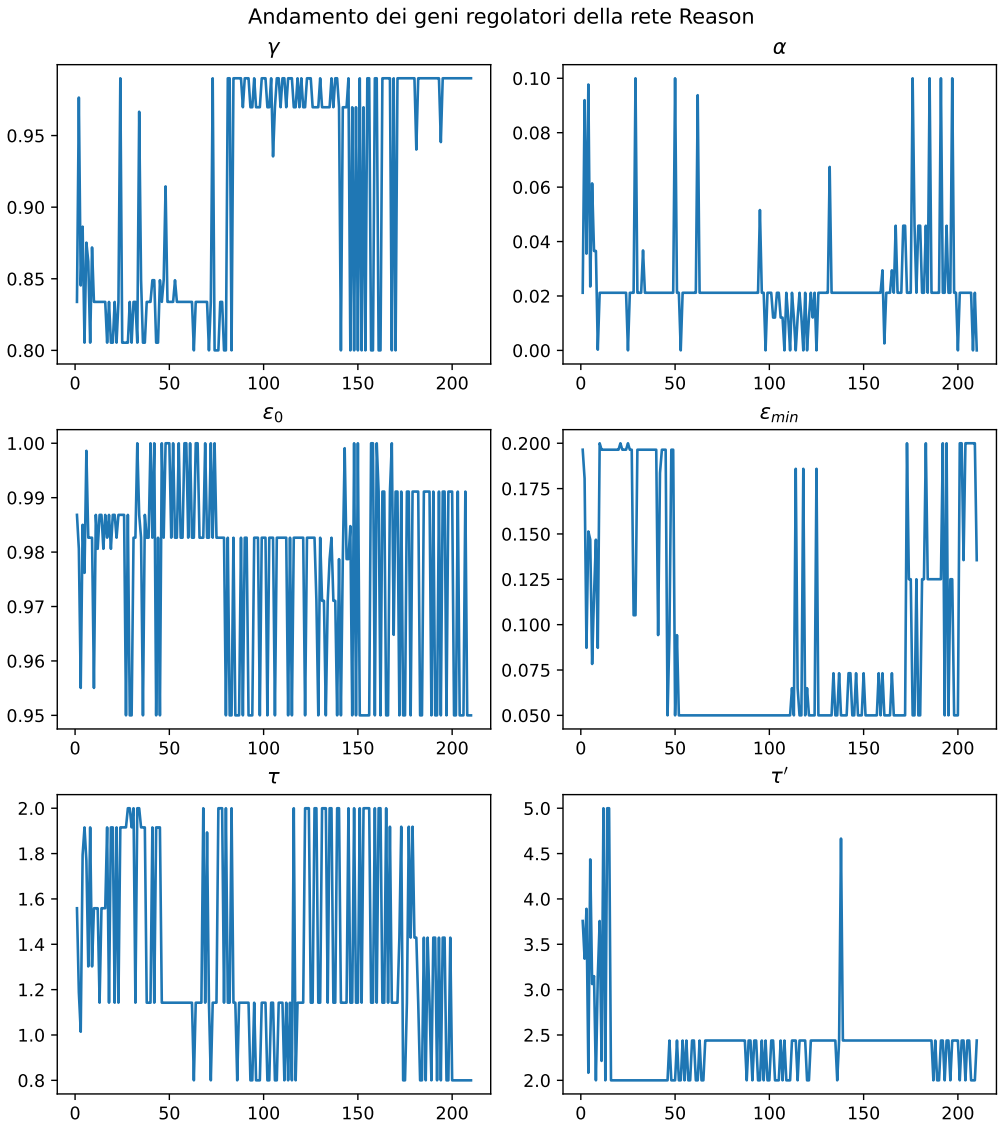

Figura 4.6: Grafici di tendenza per i geni regolatori del discount factor γ, del learning rate α, del tasso di esplorazione iniziale ε_0, del tasso di esplorazione minimo ε_{min}, e dei periodi di aggiornamento τ della rete principale e τ' della rete Target ottimizzati tramite Algoritmo Genetico per il lobo Reason. Sull'asse x sono rappresentati gli ID degli agenti—rappresentando quindi le generazioni in ordine cronologico—a cui viene fatto corrispondere sull'asse y il valore registrato dall'agente per la misura osservata.

quello applicato ai geni regolatori dei parametri vitali.

4.4 Prestazioni e giocabilità

Modalità di valutazione. L'obiettivo primario della valutazione delle prestazioni del sistema è analizzare il funzionamento del motore di gioco per valutarne la fruibilità da parte degli utenti. È quindi necessario replicare, ai fini della valutazione, le stesse condizioni di esecuzione previste per l'utente finale.

I dati sono stati quindi raccolti su un solo mondo attivando i contributi di ottimizzazione sia del DRL che del GA. La dimensione di base della popolazione, per questioni di consistenza con le simulazioni precedenti, è stata fissata anche in questo caso a 7 individui. Le differenze principali sono l'attivazione dell'interfaccia grafica—precedentemente disattivata per questioni appunto di efficienza, in questo caso deve essere considerata perché parte fondamentale della fruizione del gioco—e la durata, fissata in questo caso a un'ora senza vincoli sulla dimensione del campione di popolazione osservato. L'esecuzione è stata svolta anche in questo caso su un laptop equipaggiato con CPU Intel Core i3-1500G1, GPU integrata e 8GB di RAM.

Metriche di valutazione. La metrica scelta per quantificare la prestazione del sistema è il *framerate*, ovvero il numero di aggiornamenti del mondo di gioco presentati all'utente ogni secondo. Si è scelto di misurare l'efficienza con tale metrica perché comunemente usata per misurare le prestazioni delle macchine in ambienti di gioco, rendendo quindi facile la valutazione della giocabilità del sistema proposto confrontando il framerate ottenuto con quello osservato in altri giochi.

In fase di valutazione delle prestazioni del sistema si deve considerare che il framerate, che in senso stretto rappresenterebbe il numero di schermate (o *frame*) mostrate all'utente indipendentemente dai passi di aggiornamento del motore di gioco, viene invece nella valutazione qui proposta inteso come il numero di aggiornamenti del mondo di gioco effettuati ogni secondo, che nell'implementazione proposta corrisponde al numero di schermate mostrate all'utente. Si sottolinea inoltre che il ritmo di aggiornamento del motore di gioco è limitato a 30 Frame al Secondo (FPS).

	Framerate
Media	4.18
Minimo	0.49
Primo quartile	2.01
Mediana	2.58
Terzo quartile	4.29
Massimo	14.93

Tabella 4.6: Media, minimo e quartili della distribuzione dei valori di framerate osservati attivando il contributo del Deep Reinforcement Learning ottimizzato tramite Algoritmo Genetico.

Il sistema di valutazione prevede infine di calcolare un valore di framerate atteso ad ogni aggiornamento del mondo, basandosi sul reciproco del tempo richiesto per il singolo passo di aggiornamento.

Prestazioni complessive. La Tabella 4.6 contiene una rappresentazione sintetica della distribuzione dei valori di framerate osservati in fase di analisi delle prestazioni. Il primo dato da osservare è il framerate medio, corrispondente a circa 4.18 FPS. Tale misura lascia intuire in maniera abbastanza palese il fallimento dell'integrazione del modello di DRL all'interno del motore di gioco, dal momento che un framerate così basso porta all'osservazione di un comportamento "scattoso" che risulta poco gradevole agli occhi dell'utente e rischia di complicare le dinamiche di interazione e di inserimento dell'input, riducendo ulteriormente la fruibilità del prodotto.

Anche i valori di distribuzione osservati sembrano confermare l'ipotesi avanzata leggendo il valore medio: il 75% dei frame ha infatti avuto una durata tale da produrre un framerate di circa 4.29 FPS, rendendo di fatto inutilizzabile il gioco per la gran parte del tempo di esecuzione. Il valore massimo osservato è poi di circa 14.93 FPS, valore indicativo del fatto che nemmeno nei frame di minima attività si è raggiunto il livello massimo di efficienza permesso dal motore di gioco.

Ottimizzazione dell'interfaccia. A gravare però sulle prestazioni del sistema non è in questo caso il solo modello di DRL, ma anche il motore di rendering. L'implementazione proposta per il sistema di gioco non prevede infatti la separazione

	Framerate
Media	9.60
Minimo	0.31
Primo quartile	2.54
Mediana	5.35
Terzo quartile	20.41
Massimo	30.30

Tabella 4.7: Media, minimo e quartili della distribuzione dei valori di framerate osservati attivando il contributo del Deep Reinforcement Learning ottimizzato tramite Algoritmo Genetico senza renderizzare l'interfaccia grafica.

dei flussi di esecuzione responsabili dell'aggiornamento del mondo e delle schermate, rallentando inevitabilmente l'esecuzione del gioco stesso. Nella Tabella 4.7 sono sintetizzati i dati di distribuzione per il framerate ottenuti nella simulazione analizzata nella Sezione 4.3, che non prevedeva alcuna resa grafica.

Già solo rimuovendo il peso computazionale del processo di rendering del mondo di gioco il valore medio del framerate cresce fino a circa 9.60, più che raddoppiando il ritmo di esecuzione rispetto alla simulazione con interfaccia grafica. Anche i valori di distribuzione ricavati dallo studio dei quartili indicano un netto miglioramento rispetto al risultato precedente: sebbene la mediana del framerate sia pari a solo 5.35 FPS—già comunque maggiore rispetto al valore medio complessivamente osservato in caso di attivazione dell'interfaccia grafica—il terzo quartile misura 20.41 FPS, valore in corrispondenza di cui il ritmo di scorrimento del gioco può essere reputato accettabile. Un primo avvicinamento al livello di efficienza desiderato si potrebbe quindi ottenere separando i passi di aggiornamento del mondo e dell'interfaccia grafica e assegnando tali compiti a flussi di controllo separati da eseguire parallelamente, come accade nella maggior parte dei sistemi di gioco moderni.

Ottimizzazione dell'apprendimento. Infine si sottolinea che le librerie usate per l'implementazione delle DNNs di controllo supportano pienamente il parallelismo su GPU, che renderebbe notevolmente più rapidi i passi decisionali e di apprendimento del DRL. Eseguire il motore di gioco su una macchina equipaggiata con una GPU adatta all'esecuzione dei calcoli richiesti dai modelli di apprendimento—

	Framerate
Media	13.25
Minimo	1.08
Primo quartile	6.99
Mediana	10.53
Terzo quartile	20.00
Massimo	22.22

Tabella 4.8: Media, minimo e quartili della distribuzione dei valori di framerate osservati attivando il contributo del Deep Reinforcement Learning ottimizzato tramite Algoritmo Genetico, eseguendo il test su laptop da gioco con GPU esterna.

requisito spesso soddisfatto, se si considerano i sistemi da gioco attualmente disponibili sul mercato—potrebbe quindi essere l'ultimo passo verso l'ottenimento di un livello di efficienza accettabile per considerare fattibile l'integrazione del modello di DRL all'interno del motore di gioco.

Valutazione su laptop da gioco. Per valutare effettivamente l'efficacia del sistema di gioco proposto sfruttando al meglio le ottimizzazioni offerte dalle tecnologie implementative scelte, si ripete la valutazione su un laptop da gioco con CPU Intel Core i5-12500H, 16GB di RAM e GPU esterna Nvidia GeForce RTX 3050 Laptop con 4GB di memoria dedicata. I parametri della simulazione su cui si raccolgono i dati di efficienza restano gli stessi: un'ora di esecuzione su un singolo mondo con 7 individui, attivando i contributi di ottimizzazione sia del DRL che del GA e mantenendo attiva l'interfaccia grafica.

La Tabella 4.8 contiene quindi una sintesi della media e della distribuzione osservata in tale scenario per il valore del framerate. Già osservando il solo valore medio del framerate, paragonandolo a quello osservato sul laptop con GPU integrata, si nota un netto miglioramento: rispetto ai 4.18 FPS precedentemente osservati si passa ora a una media di circa 13.25, più che triplicando la prestazione ottenuta.

I dati dei quartili della distribuzione confermano il risultato osservato. Nonostante il dato massimo non raggiunga il limite per il ritmo di esecuzione posto al motore di gioco—che veniva invece raggiunto anche su laptop con GPU integrata disattivando il rendering, che si può quindi considerare almeno in parte respon-

sabile del risultato—già il valore del primo quartile (circa 6.99 FPS) è quasi 1.63 volte più grande rispetto al valore del terzo quartile osservato in precedenza con GPU integrata. Ciò significa che un frame considerato in questa osservazione lento, appartenente al peggiore quartile con GPU esterna, sarebbe stato invece reputato rapido e sarebbe potuto appartenere al migliore quartile con GPU integrata.

Inoltre si sottolinea che circa il 44.36% di tutti i frame osservati con GPU esterna è stato eseguito più rapidamente rispetto al migliore frame osservato con GPU integrata. Ciò significa che quasi nella metà dei frame eseguiti il laptop da gioco è riuscito a superare la massima velocità di esecuzione ottenuta nell'osservazione precedente.

Sebbene quindi non sia ancora un risultato ottimale—la mediana a 10.53 FPS denuncia comunque una lentezza generale del sistema per la maggior parte del tempo di esecuzione—il valore di framerate osservato eseguendo il sistema su un laptop da gioco è rassicurante nei confronti delle prestazioni dello stesso. Non è infatti da escludere che, semplicemente separando il flusso di esecuzione del processo di rendering dell'interfaccia o introducendo delle ottimizzazioni ulteriori nella gestione dei passi di aggiornamento del mondo di gioco, si possa ulteriormente incrementare il valore osservato fino a raggiungere un ritmo di esecuzione accettabile per ritenere il sistema integrabile in un motore di gioco completo.

Capitolo 5

Conclusione e lavori futuri

La ricerca svolta ha quindi come obiettivo quello di affrontare diversi problemi legati all'efficacia dei modelli di DRL per il controllo di agenti di gioco autonomi e all'efficienza del processo di apprendimento.

Ottimizzazione dei parametri di apprendimento. In primo luogo si propone un metodo computazionalmente sostenibile per la ricerca educata dei parametri degli algoritmi di DRL. Risolvere questo problema—noto in letteratura ma spesso non affrontato, ricorrendo piuttosto a valori determinati empiricamente tramite esplorazioni parziali e valutazioni qualitative [MKS+15]—permetterebbe infatti di ottimizzare i modelli di DRL tramite iterazioni successive del processo di apprendimento, senza dover determinare a priori un set di parametri reputati ottimali ma piuttosto sfruttando la convergenza verso valori ideali garantita dalla ricerca educata.

L'uso di un GA permette in tal senso di abbattere sensibilmente il costo della ricerca—che per essere affidabile dovrebbe altrimenti essere estensiva e considerare tutte le configurazioni possibili—guidando l'esplorazione di nuove combinazioni di parametri o il mantenimento di quelle osservate tramite l'analisi e la valutazione continua dei risultati ottenuti.

Modello di comportamento attenzione-azione. Si realizza inoltre un sistema di controllo basato sul binomio attenzione-azione piuttosto che sul solo concetto di azione. Modellare esplicitamente la pre-processazione delle letture dell'am-

biente svolte dagli agenti per produrre un valore di attenzione semanticamente significativo—nel caso dello studio svolto il tipo di oggetto fra le varie entità di gioco con cui è possibile interagire—permette infatti di articolare strategie più complesse e redditizie.

Rappresentare separatamente i concetti di attenzione e azione rende infatti possibile la modellazione di un "pensiero" bidimensionale basato su verbo e oggetto, che determina le azioni dell'agente a seconda dell'oggetto verso cui sono rivolte. Al contrario i modelli basati sul solo valore di azione appiattiscono il processo decisionale a una valutazione unidimensionale, rendendo quindi impossibile la modellazione di strategie altrettanto complesse.

Efficienza del sistema integrato. Infine si affronta esplicitamente il problema dell'efficienza dei modelli di apprendimento e di evoluzione. Spesso infatti gli agenti autonomi basati su DRL sono implementati come sistemi di controllo indipendenti dall'ambiente con cui interagiscono, e sono talvolta eseguiti su macchine dedicate che permettano l'allocazione delle risorse necessarie per affrontare gli elevati costi computazionali richiesti.

Al contrario lo studio svolto propone di integrare il modello di DRL e il GA responsabile dell'evoluzione della popolazione direttamente all'interno del motore di gioco in cui agiscono, con l'obiettivo di realizzare un sistema eseguibile localmente su una macchina da gioco mantenendo una prestazione sufficiente a garantire un'esperienza utente positiva. A tale fine è fondamentale trovare un equilibrio fra l'alleggerimento del modello di DRL—che è sicuramente la componente più gravosa del sistema dal punto di vista computazionale—e l'inevitabile deterioramento di efficacia che ne consegue.

Risultati ottenuti. L'analisi svolta si è quindi concentrata su 4 metriche principali: l'efficacia del solo GA, l'efficacia del solo modello di DRL, l'efficacia del modello integrato di ottimizzazione e addestramento e infine l'efficienza del sistema di gioco nel suo complesso.

Dall'analisi dei risultati ottenuti applicando il solo GA è apparsa da subito evidente la robustezza complessiva dell'algoritmo proposto, che nonostante le instabilità ha dimostrato convergenza verso valori ottimali. I dati osservati per il

tempo di vita atteso durante lo studio del modello combinato di ottimizzazione confermano tale osservazione: nonostante una popolazione iniziale con un genoma particolarmente sfavorevole in tal senso, l'evoluzione ha portato a generare individui con un tempo di vita atteso pari al valore massimo ammissibile—poi rapidamente scartato per prediligere una configurazione di parametri più favorevole al processo di apprendimento, dimostrando la capacità dell'algoritmo di mediare fra le due diverse direzioni di ottimizzazione.

La valutazione del rendimento del DRL ha poi fatto emergere l'importanza dell'ottimizzazione dei parametri degli agenti, sia vitali che di apprendimento. La differenza di rendimento osservata tra il modello di ottimizzazione combinato e il modello di solo DRL con generazione casuale dei genomi ha infatti reso palese l'impatto che i genomi sfavorevoli possono avere sulla qualità del processo di apprendimento, in primo luogo limitando le capacità di esplorazione del mondo di gioco e in secondo luogo rendendo più complessa la convergenza del modello di DRL verso la strategia ottimale. Ciò è dimostrato dal fatto che numerosi agenti—quasi la metà—non sono riusciti ad apprendere una strategia efficace in caso di applicazione del solo DRL, problema quasi completamente superato nello scenario di applicazione del modello di ottimizzazione combinato.

Criticità osservate. Dall'altra parte in fase di analisi sono emerse anche alcune criticità, soprattutto in merito all'efficienza del modello proposto. Si è infatti visto come in assenza di ottimizzazioni mirate—quali possono essere l'esecuzione del rendering in un flusso di controllo dedicato, o una modifica delle modalità di gestione degli aggiornamenti del mondo di gioco—le prestazioni ottenute dall'esecuzione del motore di gioco completo di modello combinato di ottimizzazione siano deludenti. Anche su un laptop da gioco con GPU dedicata, in grado quindi di sfruttare le ottimizzazioni offerte dalle librerie usate in fase di implementazione, le prestazioni registrate in termini di framerate non sono sufficienti a poter offrire un'esperienza piacevole all'utente, risultando invece in un flusso di gioco "scattoso" e poco fruibile.

Per quanto riguarda l'efficacia del modello combinato, invece, si vedono superate le principali criticità degli algoritmi emerse dalle analisi indipendenti degli stessi. L'instabilità del GA è infatti tollerabile se si considera l'effetto comples-

sivamente positivo che l'ottimizzazione dei genomi ha sulla qualità dei risultati ottenuti dal modello di DRL, che dimostra come il processo evolutivo sia nel complesso efficace indipendentemente dalle perturbazioni istantanee. Dall'altra parte il numero di agenti che non sono stati in grado di apprendere una strategia efficace cala sensibilmente in caso di attivazione del processo evolutivo, dimostrando come la presenza di tali individui sia probabilmente da imputare alla generazione casuale di genomi sfavorevoli piuttosto che all'inefficacia del processo di apprendimento.

5.1 Lavori futuri

La ricerca proposta non è comunque da considerarsi esaustiva rispetto agli ambiti esplorati, ma piuttosto può rappresentare un promettente punto di partenza da cui prendere spunto per approfondire varie possibili direzioni di ricerca.

Parametrizzazione delle interazioni. La valutazione del rendimento del modello di ottimizzazione combinato ha dimostrato in primo luogo l'efficacia del processo evolutivo di ottimizzazione dei parametri guidato da un GA. Con una popolazione base di 7 individui si sono infatti raggiunte nell'arco di poche generazioni configurazioni genetiche ottimali, facendo emergere quindi inequivocabilmente anche la convenienza computazionale del modello di ricerca educata dei parametri se paragonato ad altri metodi basati su ricerca estensiva degli stessi.

Da un'analisi più approfondita dei dati osservati è poi emersa una correlazione fra il tempo di vita atteso e la qualità delle strategie apprese. Tale osservazione suggerisce quindi come la parametrizzazione e l'ottimizzazione delle modalità di interazione fra agente e ambiente—oltre alla più comune ottimizzazione dei parametri dell'algoritmo di apprendimento [SLLN19, BPK25]—possa rappresentare una direzione di ricerca verso l'incremento delle prestazioni dei modelli di DRL.

Approccio neuroevolutivo. Una diversa modalità di estensione dell'ambito di applicazione dei GAs al problema di ottimizzazione delle prestazioni del DRL è poi l'implementazione di un processo neuroevolutivo [RT17]. La neuroevoluzione è infatti un metodo di ottimizzazione che prevede di evolvere—eventualmente

tramite l'applicazione di un GA, anche se esistono approcci differenti—i pesi e le topologie delle reti neurali di controllo per guidare il processo di apprendimento.

Riprendendo quindi lo studio svolto, volendo mantenere il processo di ottimizzazione dei pesi delle DNNs tramite DRL, sarebbe possibile integrare un approccio neuroevolutivo per la modifica delle topologie delle DNNs coinvolte piuttosto che dei loro pesi, di fatto estendendo la modellazione genetica degli individui alla struttura dell'organo di controllo del loro comportamento. Così facendo, si solleverebbe lo sviluppatore dalla responsabilità di definizione della topologia delle DNNs, in gran parte determinante per la prestazione delle stesse.

Sebbene rappresenti una possibilità di ulteriore raffinamento della qualità del modello di apprendimento, l'approccio neuroevolutivo potrebbe sollevare problemi se applicato ciecamente in combinazione con il metodo evolutivo di ottimizzazione dei parametri. L'efficacia della parametrizzazione proposta per l'algoritmo di apprendimento dipende infatti strettamente dalla formulazione dello stesso e dalla topologia della DNN di controllo, di cui regola il processo di addestramento. Risulta perciò particolarmente interessante lo studio dell'integrazione di un approccio neuroevolutivo al sistema proposto per indagare le modalità di interazione—costruttiva o distruttiva—dei due processi evolutivi di ottimizzazione.

Estensione delle Reti Neurali Profonde. La formulazione proposta per il modello di DRL poi, sebbene generalmente efficace, si basa su DNNs con una topologia estremamente semplice e sulla processazione dei soli dati di input rilevanti alla determinazione delle azioni da svolgere nel contesto di gioco affrontato. Tale approccio, intrapreso principalmente con l'obiettivo di limitare il costo computazionale richiesto per il processo decisionale, esclude però la possibilità di adattare il modello proposto ad altri ambienti di gioco.

Superare i vincoli imposti dal tentativo di riduzione del costo computazionale complessivo permetterebbe quindi di estendere la topologia delle DNNs di controllo ad architetture arbitrariamente complesse, avvicinandosi all'obiettivo ideale di creare agenti di gioco generalmente in grado di affrontare qualsiasi sfida basandosi su un input comparabile a quello fornito ai giocatori umani [RT17].

In particolare l'implementazione di modelli di DRL basati su meccanismi di visione artificiale, e quindi sull'uso delle schermate come unico valore grezzo di in-

put, sembra essere la direzione per il raggiungimento di tale obiettivo [MKS+15]. Realizzare quindi DNNs più complesse rappresenta una possibile direzione di estensione del lavoro proposto, con l'obiettivo di studiare la possibilità di adattamento del modello di apprendimento così addestrato al controllo di agenti di gioco in ambienti diversi.

Studio delle interazioni utente. L'estensione in tal senso del modello di DRL, con l'obiettivo di permettere l'interazione agente-ambiente tramite osservazioni comparabili a quelle dei giocatori umani, potrebbe poi se implementata efficacemente aprire direzioni di ricerca più strettamente legate allo studio delle interazioni uomo-macchina nell'ambito dei videogiochi.

La modellazione esplicita del concetto di Attention come valore pre-processato e semanticamente significativo dell'osservazione dell'ambiente ricopre in tal senso un ruolo fondamentale. Rappresentare il valore intermedio prodotto da Attention in maniera rapportabile alle comuni metriche di valutazione delle interazioni umane con i mondi di gioco permetterebbe infatti di approfondire la comprensione del processo decisionale che guida gli agenti di DRL paragonandolo—per differenze e analogie—al processo decisionale dei giocatori umani.

Ricalcando quindi la direzione principale di sviluppo della visione artificiale, fortemente basata sulle conoscenze neuroscientifiche del processo di visione umana, si potrebbero poi proporre modelli di apprendimento e valutazione degli stimoli basati su processi computazionali quanto più simili a quelli che guidano il pensiero umano, di fatto simulando il funzionamento di un cervello biologico. Tale approccio, primariamente legato quindi al campo delle neuroscienze, può essere adottato per introdurre negli agenti autonomi disturbi della percezione, quali possono essere daltonismo, deficit dell'attenzione o iperfissazioni.

L'obiettivo della ricerca in tale direzione diventerebbe quindi quello di sfruttare il meccanismo di Attention per misurare con metriche note i risultati ottenuti dalla popolazione di agenti autonomi con disturbi indotti, con l'obiettivo finale di valutare l'accessibilità dei mondi di gioco studiati e proporre metodi di modellazione in grado di garantire la produzione di ambienti e dinamiche di gioco accessibili.

Bibliografia

[AHSB18] Forest Agostinelli, Guillaume Hocquet, Sameer Singh, and Pierre Baldi. From reinforcement learning to deep reinforcement learning: An overview. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11100 LNAI:298 – 328, 2018.

[BPK25] Bartłomiej Brzęk, Barbara Probierz, and Jan Kozak. Exploration-driven genetic algorithms for hyperparameter optimisation in deep reinforcement learning. *Applied Sciences (Switzerland)*, 15(4), 2025.

[GC98] Stephen Grand and Dave Cliff. Creatures: Entertainment software agents with artificial life. *Autonomous Agents and Multi-Agent Systems*, 1(1):39 – 57, 1998.

[GCM97] Stephen Grand, Dave Cliff, and Anil Malhotra. Creatures: Artificial life autonomous software agents for home entertainment. page 22 – 29, 1997.

[Hub64] Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73 – 101, 1964.

[KB15] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. 2015.

[KLM96] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237 – 285, 1996.

[LL12] Adam Lipowski and Dorota Lipowska. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications*, 391(6):2193 – 2196, 2012.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529 – 533, 2015.

[RT17] Sebastian Risi and Julian Togelius. Neuroevolution in games: State of the art and open challenges. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(1):25–41, 2017.

[SHM+16] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484 – 489, 2016.

[SLLN19] Adarsh Sehgal, Hung La, Sushil Louis, and Hai Nguyen. Deep reinforcement learning using genetic algorithm for parameter optimization. page 596 – 601, 2019.

[SSS+17] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George Van Den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354 – 359, 2017.

[Tes90] Gerald Tesauro. Neurogammon: A neural-network backgammon program. page 33 – 39, 1990.

[TT95] Covid Tesau and Gerald Tesau. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58 – 68, 1995.

[VDN26] Khanh-Linh Vuong, Huy Truong Dinh, and Cuong Tuan Nguyen. Integrating visual attention into deep reinforcement learning for enhanced control in racing games. *Communications in Computer and Information Science*, 2587 CCIS:256 – 270, 2026.

[WWL+24] Xu Wang, Sen Wang, Xingxing Liang, Dawei Zhao, Jincai Huang, Xin Xu, Bin Dai, and Qiguang Miao. Deep reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4):5064 – 5078, 2024.