GottaBattleEmAllGA

Angelo Alberico M: 0512113929
Gianvincenzo Landolfi M: 0512114922
Luigi Rocchino M: 0512114301

Data di consegna 2024

Link alla repository Github

Sommario

1 Introduzione

I Pokémon rappresentano un vasto universo affascinante, dove l'impiego di strumenti di intelligenza artificiale apre nuove prospettive nell'analisi delle interazioni tra queste creature. Questo progetto si concentra sull'utilizzo di algoritmi di intelligenza artificiale per prevedere l'esito degli scontri tra due squadre Pokémon. Attraverso l'analisi delle caratteristiche e delle abilità dei Pokémon, cerchiamo di sviluppare un modello predittivo per determinare il vincitore di un confronto.

Per cominciare, verranno introdotti i Pokémon, seguendo poi con l'analisi della specifica PEAS (Performance, Environment, Actuators, Sensors) e delle proprietà dell'ambiente in cui si svolgono gli scontri.

1.1 Introduzione ai Pokémon

I Pokémon sono creature immaginarie popolari nell'universo dei videogiochi, dei fumetti, dell'animazione e dei giocattoli. Creati nel 1996 da Satoshi Tajiri e Ken Sugimori e sviluppati dalla società giapponese Nintendo, i Pokémon sono diventati un fenomeno culturale globale.

Queste creature abitano un mondo fantastico popolato da allenatori, ciascuno dei quali cerca di catturare, addestrare e combattere con i Pokémon. Ogni Pokémon ha abilità uniche, caratteristiche fisiche e poteri speciali, rendendoli affascinanti e diversificati.

I Pokémon sono comunemente classificati in diverse specie e tipologie, che variano da creature simili a animali reali a forme bizzarre e mistiche. I giochi Pokémon si concentrano spesso sulla formazione di squadre di Pokémon per competere in lotte contro altri allenatori, in tornei o in avventure per diventare il migliore allenatore di Pokémon.

Con l'introduzione di nuove generazioni di videogiochi, serie animate,

film e giocattoli, l'universo dei Pokémon continua a evolversi, attirando un vasto pubblico di appassionati di tutte le età.

1.2 Specifica PEAS

Un ambiente viene generalmente descritto tramite la formulazione PEAS:

- **Performance**: L'operato dell'agente è valutato in base a diverse caratteristiche e statistiche dei Pokémon.
- **Enviroment**: L'ambiente in cui opera l'agente è costituito dall'insieme di tutte le possibili squadre che si possono comporre con i Pokémon.
- **Actuators**: L'agente interagisce con l'ambiente selezionando i Pokémon disponibili per formare le squadre.
- **Sensors**: L'agente raccoglie dall'ambiente tutte le informazioni riguardanti i Pokémon che formano le squadre.

1.3 Proprietà dell'ambiente

Le proprietà relative all'ambiente sono:

- **Completamente Osservabile**: I sensori di un ambiente danno accesso allo stato completo dell'ambiente in ogni momento. In quanto si ha accesso a tutte le informazioni relative ai Pokémon.
- **Stocastico**: Siccome è presente una componente randomica che determina lo stato successivo.

- **Sequenziale**: Le decisioni dell'agente (come selezionare o cambiare un Pokémon) influenzano le decisioni successive e le prestazioni complessive del team.
- **Statico**: L'ambiente è invariato mentre un agente sta deliberando.
- **Discreto**: L'ambiente fornisce un numero limitato di percezioni e azioni distinte, chiaramente definite.
- **Singolo Agente**: L'ambiente consente la presenza di un unico agente.

2 Analisi del problema

Il problema che ci proponiamo di affrontare riguarda la determinazione del vincitore in uno scontro tra due squadre Pokémon. Questo compito presenta diverse sfide, tra cui la vastità dello spazio di ricerca, la complessità delle interazioni tra i Pokémon e le diverse strategie adottate dagli allenatori.

Una delle principali sfide è decidere quale approccio adottare per risolvere il problema. Due approcci comuni sono la ricerca locale e l'apprendimento. Nella ricerca locale, l'obiettivo è trovare una soluzione ottima esplorando in profondità le possibili mosse e adattando la strategia in base alle condizioni del momento. D'altro canto, l'apprendimento coinvolge l'uso di algoritmi di intelligenza artificiale per addestrare un modello predittivo in grado di stimare il risultato di uno scontro basandosi su dati storici e pattern emergenti.

Il nostro obiettivo è identificare l'approccio più adatto per affrontare il problema della determinazione del vincitore in uno scontro tra squadre Pokémon, tenendo conto delle specifiche esigenze e delle caratteristiche uniche del dominio del problema.

2.1 NO all'Apprendimento

Nonostante l'appeal e la promessa di precisione offerta dagli algoritmi di apprendimento, la loro adozione potrebbe non essere la scelta più sensata nel contesto della determinazione del vincitore in uno scontro tra squadre Pokémon. La principale ragione dietro questa decisione risiede nell'assenza di dati storici di partite. Nell'ambito dei combattimenti Pokémon, i dati storici dettagliati e affidabili sulle partite passate sono spesso difficili da ottenere o semplicemente non esistono. Senza un ampio set di dati su partite precedenti, un algoritmo di apprendimento potrebbe non essere in grado di addestrarsi efficacemente o di generare previsioni accurate. Inoltre, anche se disponibili, i dati storici potrebbero non riflettere appieno la varietà delle situazioni di gioco, con molte partite che si svolgono in modi unici e imprevedibili. Pertanto, senza una base solida di dati storici su cui basare il processo di apprendimento, l'uso di algoritmi di apprendimento potrebbe non essere pratico o produttivo per affrontare il problema della determinazione del vincitore in uno scontro Pokémon.

2.2 Scomposizione del problema

Nel tentativo di risolvere il complesso problema della determinazione del vincitore in uno scontro tra due squadre Pokémon, possiamo suddividerlo in due sottoproblemi distinti.

Il primo sottoproblema riguarda l'individuazione delle migliori squadre possibili attraverso l'uso di algoritmi di ottimizzazione di Ricerca Locale, come l'hill climbing, il simulated annealing e gli algoritmi genetici. Questi algoritmi ci consentono di esplorare una varietà di soluzioni valide che eccellono in diversi aspetti legati alle statistiche dei Pokémon che formeranno la squadra.

Il secondo sottoproblema si concentra sulla previsione del vincitore dello

scontro. Utilizzando le informazioni fornite dal primo sottoproblema, possiamo progettare un algoritmo che, prendendo in input il fronte di Pareto e le due squadre in competizione, calcola la distanza di ciascuna squadra rispetto alle squadre forti del fronte. Utilizzando queste informazioni, siamo in grado di determinare quale delle due squadre disti di meno dalle rispettive squadre forti e quindi prevedere il vincitore dello scontro.

2.2.1 Ricerca locale

Nel contesto del primo sottoproblema, possiamo considerare l'utilizzo di diversi algoritmi di ricerca locale per individuare le migliori squadre possibili. Tra questi, l'Hill Climbing, il Simulated Annealing e l'Algoritmo Genetico sono tra i più comuni.

L'Hill Climbing è un algoritmo di ottimizzazione locale che esplora lo spazio delle soluzioni cercando iterativamente una soluzione migliore adiacente alla soluzione corrente. Sebbene sia efficiente e facile da implementare, l'Hill Climbing può rimanere bloccato in minimi locali e non garantisce la ricerca della soluzione ottimale.

Simulated Annealing è una tecnica di ottimizzazione probabilistica ispirata al processo di ricottura dei metalli, che permette di esplorare lo spazio delle soluzioni in modo più ampio, accettando occasionalmente soluzioni peggiori per evitare minimi locali. Tuttavia, richiede la messa a punto di parametri sensibili, come la temperatura di annealing.

L'Algoritmo Genetico è una tecnica di ottimizzazione basata su popolazioni ispirata alla teoria dell'evoluzione biologica, che utilizza concetti come la selezione naturale, la mutazione e l'incrocio per esplorare lo spazio delle soluzioni. È particolarmente adatto a problemi con molteplici obiettivi e vincoli, come la ricerca delle migliori squadre Pokémon.

Nel nostro contesto, l'utilizzo di un Algoritmo Genetico potrebbe essere preferibile, poiché permette di esplorare efficacemente lo spazio delle soluzioni multi-obiettivo e di identificare un insieme di squadre competitive rappresentate dal fronte di Pareto.

2.2.2 Distanza dal Fronte

Per quanto riguarda il secondo sottoproblema, possiamo utilizzare un approccio basato sulla distanza dal fronte di Pareto per valutare la diversità tra le squadre e determinare il vincitore dello scontro. Progettiamo un algoritmo che, prendendo in input il fronte di Pareto e le due squadre in competizione, calcola la distanza di ciascuna squadra rispetto alle squadre forti del fronte.

Successivamente, utilizziamo queste informazioni per determinare quale delle due squadre disti di meno dalla rispettiva squadra forte del fronte di Pareto. In altre parole, confrontiamo le squadre in competizione con le squadre di riferimento del fronte di Pareto e calcoliamo la loro distanza. La squadra che si avvicina di più alla rispettiva squadra forte del fronte di Pareto è considerata quella con maggiori probabilità di vincere lo scontro.

Questo approccio ci consente di prevedere il vincitore dello scontro in modo efficace, basandoci sulle caratteristiche delle squadre presenti nel fronte di Pareto.

3 Struttura del Progetto

La struttura del nostro progetto è stata progettata per massimizzare l'efficienza e la flessibilità nell'implementazione del nostro algoritmo genetico per la determinazione del vincitore in uno scontro tra due squadre Pokémon. Abbiamo adottato un approccio modulare e organiz-

zato, che comprende l'utilizzo della libreria JMetal per implementare il nostro algoritmo genetico.

Abbiamo scelto JMetal per la sua flessibilità e la vasta gamma di funzionalità offerte per la progettazione e l'implementazione di algoritmi evolutivi. La sua architettura modulare ci ha consentito di adattare facilmente l'algoritmo genetico alle nostre specifiche esigenze, mentre le sue funzionalità di ottimizzazione multiobiettivo si sono rivelate particolarmente utili per gestire la complessità delle nostre analisi.

Inoltre, abbiamo integrato nel nostro progetto due dataset completi contenenti tutti i dati relativi ai Pokémon, strutturato in formato CSV. Questi dataset sono stati fondamentali per alimentare il processo decisionale del nostro algoritmo genetico, consentendo una valutazione accurata delle squadre Pokémon e delle loro prestazioni.

Abbiamo decomposto il nostro progetto in due sottosistemi distinti per affrontare i due sottoproblemi precedentemente identificati. Il primo sottosistema si occupa dell'individuazione delle migliori squadre possibili, mentre il secondo affronta l'analisi delle prestazioni delle squadre in competizione e la previsione del vincitore dello scontro. Questa suddivisione ci ha permesso di gestire in modo efficiente la complessità del problema e di implementare soluzioni mirate per ciascun aspetto del processo decisionale.

4 Progettazione degli Algoritmi

4.1 Algoritmo Genetico

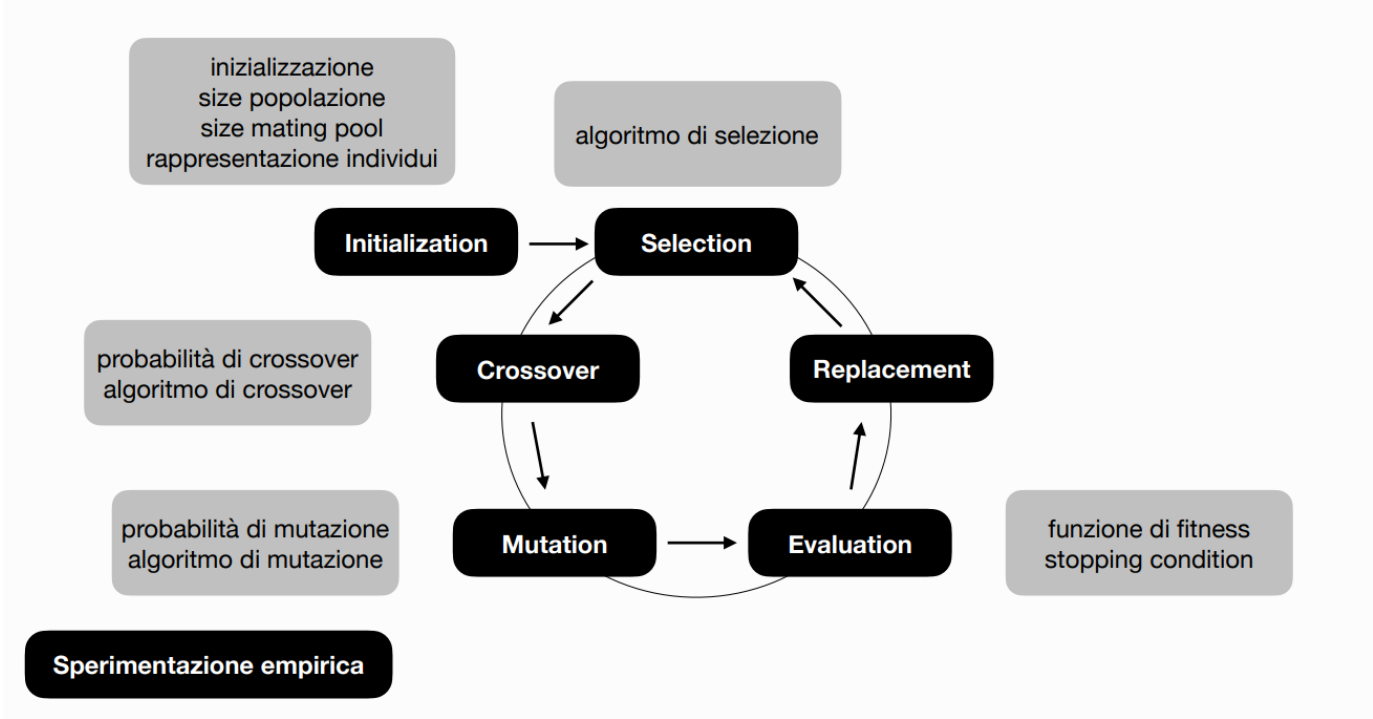

Figura 1 - Step per la progettazione del GA.

4.1.1 Codifica degli Individui

Per rappresentare gli individui nel nostro algoritmo genetico, adotteremo un approccio basato su un array di interi. Ogni elemento di questo array corrisponderà alla chiave di un Pokémon all'interno di un'HashMap pre-caricata che contiene tutti i Pokémon del dataset.

Attraverso questa scelta, miriamo a garantire una gestione efficiente degli individui e un accesso rapido alle informazioni specifiche di ciascun Pokémon. Questa rappresentazione agevolerà le operazioni di selezione,

crossover e mutazione, fornendo nel contempo una buona efficienza computazionale.

4.1.2 Inizializzazione della Popolazione

Una caratteristica fondamentale degli algoritmi genetici è la presenza di una popolazione iniziale di individui, che costituisce il punto di partenza per il processo evolutivo. Questa popolazione iniziale è essenziale perché fornisce diversità genetica all'algoritmo, consentendo di esplorare una vasta gamma di soluzioni potenziali nello spazio del problema. Senza una popolazione iniziale diversificata, l'algoritmo genetico potrebbe convergere prematuramente su soluzioni subottimali.

Per garantire una valutazione accurata delle performance del nostro algoritmo genetico, sperimenteremo empiricamente due approcci distinti all'inizializzazione della popolazione. Il primo approccio prevede l'inizializzazione casuale, in cui gli individui vengono generati casualmente all'interno del dominio del problema.

Il secondo approccio impiega un metodo di inizializzazione basato su euristiche. Qui, sfrutteremo conoscenze specifiche del dominio del problema per generare una popolazione iniziale che rifletta le caratteristiche desiderate delle soluzioni.

4.1.3 Funzione di Fitness

Le funzioni di fitness sono fondamentali negli algoritmi genetici poiché servono a valutare la qualità di ciascun individuo all'interno della popolazione. Determinano quale individuo è più adatto a sopravvivere e a essere selezionato per generare la prossima generazione di individui.

Adotteremo una funzione di fitness multiobiettivo, che considererà simultaneamente più criteri di valutazione.Questo ci consentirà di val-

utare gli individui in base a più obiettivi(funzioni) e di specificare dei pesi per ognuno di essi.

Le nostre quattro funzioni di fitness saranno:

1. **Media del Totale di un Team**: Questa funzione calcolerà la media delle statistiche totali dei Pokémon presenti nella squadra. Le statistiche totali includeranno i punti base di attacco, difesa, velocità e altri attributi rilevanti. Un punteggio più alto indicherà una squadra globalmente più potente.

2. **Copertura del tipo Pokemon**: Questa funzione valuterà la diversità dei tipi di Pokémon presenti nella squadra. Una copertura equilibrata dei diversi tipi di Pokémon potrà aumentare le possibilità di affrontare una varietà di avversari con efficacia.

3. **Copertura del tipo di Mossa**: Questa funzione valuterà la diversità dei tipi di mosse disponibili nella squadra. Una copertura equilibrata dei diversi tipi di mosse potrà consentire al team di gestire una gamma più ampia di situazioni di combattimento.

4. **Statistiche delle mosse**: Questa funzione valuterà le statistiche e la potenza delle mosse utilizzate dai Pokémon nella squadra. Le mosse con statistiche più alte e potenti potranno influenzare positivamente le prestazioni complessive del team durante gli scontri.

Per il nostro scopo, utilizzeremo una funzione di fitness multi-obiettivo espressa dalla forma:

$$\text{Fitness} = w_1 \cdot f_1 + w_2 \cdot f_2 + \ldots + w_i \cdot f_i$$

dove ogni f_i rappresenta uno degli obiettivi della nostra ottimizzazione, e w_i è il peso associato all'obiettivo corrispondente. È importante notare che i valori di ciascun obiettivo f_i saranno normalizzati prima di essere utilizzati nella funzione di fitness. La normalizzazione è necessaria per garantire che gli obiettivi con diverse scale o range di valori abbiano un impatto equilibrato sulla valutazione complessiva della fitness. Questo processo di normalizzazione aiuta a garantire che ogni obiettivo contribuisca in modo equo al processo di ottimizzazione, migliorando l'efficacia complessiva del nostro algoritmo genetico. In particolare utilizzeremo la min-max normalization:

$$\widehat{y} = a + \frac{(x - \min(x))(b-a)}{\max(x) - \min(x)}$$

4.1.4 Operatori Genetici

Gli operatori genetici includono la selezione, il crossover e la mutazione e sono fondamentali nell'evoluzione della popolazione all'interno di un algoritmo genetico. Questi operatori lavorano insieme per esplorare e sfruttare lo spazio delle soluzioni in modo efficace, influenzando il bilancio tra exploitation ed exploration.

Un adeguato equilibrio tra exploitation ed exploration è cruciale per il successo dell'algoritmo genetico. L'exploitation favorisce la convergenza verso soluzioni di alta qualità, mentre l'exploration permette di esplorare nuove possibilità e di evitare massimi locali. Pertanto, gli operatori genetici devono essere progettati e bilanciati attentamente per ottenere il miglior compromesso tra exploitation ed exploration.

4.1.5 Selezione

La fase di selezione in un algoritmo genetico svolge un ruolo fondamentale nel determinare quali individui della popolazione saranno inclusi

nel mating poll, i quali avranno la possibilità di riprodursi e generare la prossima generazione di individui. Per la nostra implementazione, esploreremo due strategie di selezione diverse: K-Tournament e RandomSelection.

- **K-Tournament**: La strategia K-Tournament seleziona casualmente K individui dalla popolazione e li fa competere tra loro. L'individuo con la fitness più alta nel torneo viene selezionato per la riproduzione.
- **RandomSelection**: La strategia di selezione casuale, o RandomSelection, seleziona casualmente gli individui dalla popolazione senza considerare la loro fitness. Anche se questa strategia può sembrare meno sofisticata rispetto al K-Tournament, può essere utile per mantenere una maggiore diversità nella popolazione e prevenire la convergenza prematura verso una soluzione subottimale.

4.1.6 Crossover

Il crossover è un operatore genetico fondamentale che permette di combinare le informazioni genetiche provenienti da due genitori selezionati per generare una prole con caratteristiche derivate da entrambi i genitori. Per la nostra implementazione, esploreremo due strategie di crossover diverse: K-Point e RandomK-Point.

- **K-Point**: La strategia K-point il punto di crossover è fissato e non è casuale. Questo punto divide i genitori in due segmenti, e i

segmenti vengono scambiati tra i genitori per generare i discendenti. Il punto di crossover verrà selezionato sistematicamente per effettuare sperimentazioni empiriche.

- **RandomK-Point**: La strategia RandomK-Point utilizza un approccio simile al K-Point, ma il punto di crossover è selezionato casualmente lungo la lunghezza dei cromosomi genitori.

4.1.7 Mutazione

La mutazione è un altro operatore genetico essenziale all'interno degli algoritmi genetici, il cui compito è introdurre variazioni casuali nei genotipi degli individui della popolazione.

Una delle strategie di mutazione comuni è il **Random Resetting**, in cui un gene casuale all'interno del cromosoma di un individuo viene selezionato casualmente e sostituito con un valore casuale. Verrà sperimentata la probabilità di mutazione ideale.

4.1.8 Stopping Condition

La stopping condition, o condizione di terminazione, determina quando l'esecuzione dell'algoritmo deve essere interrotta. Serve a garantire che l'algoritmo genetico abbia sufficiente tempo per esplorare e sfruttare efficacemente lo spazio delle soluzioni, evitando allo stesso tempo di continuare indefinitamente l'ottimizzazione senza ottenere miglioramenti significativi.

Nel nostro caso, la stopping condition verrà implementata impostando un numero massimo di iterazioni. Tale numero verrà definito empiricamente.