Simulazione di guida autonoma con NEAT (NeuroEvolution of Augmenting Topologies)

Andrea De Lorenzis[1]

[1]a.delorenzis@campus.uniurb.it

Riassunto

In questo progetto si è sviluppato un programma con lo scopo di permettere a un'automobile di navigare autonomamente in un ambiente simulato. L'ambiente di simulazione, in due dimensioni, è stato sviluppato utilizzando la libreria Pygame e, per l'apprendimento, è stato impiegato l'algoritmo NEAT (NeuroEvolution of Augmenting Topologies), una metodologia evolutiva che permette di sviluppare reti neurali che modificano la propria infrastruttura, oltre che i pesi delle connessioni, nel corso del tempo per raggiungere determinati obiettivi di apprendimento. Inoltre, si è utilizzato l'approccio di apprendimento basato su curriculum, esponendo l'auto progressivamente a sfide sempre maggiori, permettendo così all'algoritmo di affinare ed evolvere l'architettura della rete neurale superando potenziali minimi locali. Questa combinazione di tecniche ha portato allo sviluppo di un modello in grado di superare con successo il percorso di validazione senza incidenti.

1 Introduzione

Il problema centrale affrontato in questo progetto è quello di sviluppare un modello di intelligenza artificiale capace di guidare un'automobile in modo autonomo in un ambiente simulato. Questa sfida e la sua soluzione permettono di evidenziare il potenziale delle tecniche di apprendimento evolutivo nel campo del machine learning, e fornisce una solida base per future esplorazioni nell'ambito della guida autonoma simulata e degli algoritmi evolutivi.

La creazione dell'ambiente di simulazione è un aspetto cruciale di questo progetto, gettando le fondamenta per il resto del lavoro. Si è optato per un ambiente 2D con visuale dall'alto, permettendo di monitorare le macchine e loro sensori in qualsiasi punto del percorso. Gli input del sistema consistono in segnali provenienti da sei sensori virtuali posizionati sull'auto che rilevano l'ambiente circostante. Gli output sono le decisioni di sterzo dell'auto, gestite tramite due attuatori che controllano le manovre a sinistra e a destra.

Il cuore del progetto è l'utilizzo di **NEAT** (NeuroEvolution of Augmenting Topologies) [1], un algoritmo evolutivo per la generazione di reti neurali ottimizzate. NEAT si distingue per la sua capacità di evolvere sia la struttura che i pesi di una rete neurale, adattandosi in modo dinamico alle esigenze specifiche del problema affrontato. In NEAT il processo di apprendimento avviene attraverso una serie di generazioni e percorsi. Inizialmente, vengono create diverse reti neurali con topologie diverse, che vengono poi valutate in base alla loro capacità di guidare l'automobile lungo i percorsi simulati, a partire da quello iniziale. Le reti più performanti vengono selezionate e utilizzate per generare una nuova popolazione di reti, attraverso un processo di incrocio e mutazione genetica. L'evoluzione continua fino a quando non viene raggiunta una rete neurale che riesce a navigare il percorso di validazione in modo soddisfacente.

La strategia di presentare percorsi di allenamento in ordine di difficoltà crescente rientra nel paradigma del **curriculum-based learning** [2]. Questo approccio pedagogico, ispirato all'apprendimento umano, permette alla rete neurale di acquisire gradualmente le competenze necessarie per affrontare compiti sempre più complessi.

2 Metodi

NEAT inizia con una rete semplice e incrementa gradualmente la sua complessità. Questo viene fatto attraverso un processo di selezione naturale, incrocio e

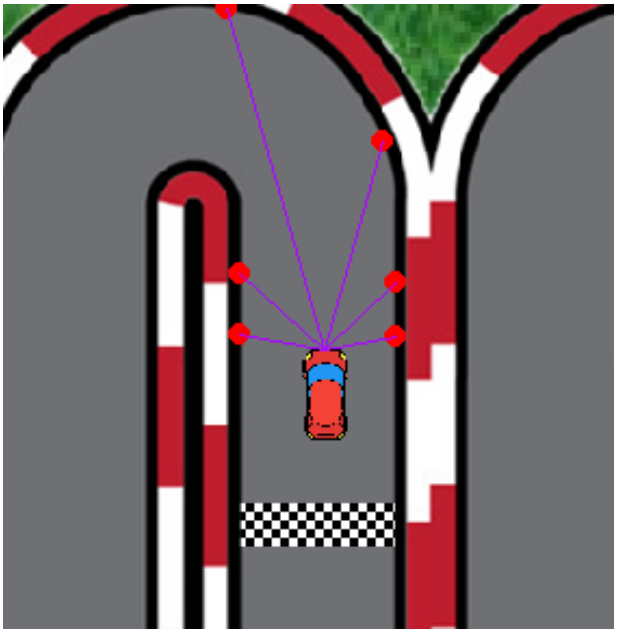


Figura 1: Disposizione dei sensori della macchina.

mutazione, simile a quello dell'evoluzione biologica. Il funzionamento di NEAT può essere schematicamente rappresentato come segue:

1. **Inizializzazione**: si parte da una popolazione di reti neurali semplici, con solo nodi di input e output senza alcuno strato nascosto.
2. **Valutazione**: ogni rete nella popolazione è valutata in base alla sua *fitness*, determinata da una funzione che valuta quanto bene viene svolto il compito assegnato.
3. **Selezione**: alla fine di ogni generazione, le reti con le prestazioni migliori in termini di fitness sono selezionate per la riproduzione.
4. **Riproduzione**: nuove reti sono generate attraverso incrocio e mutazione, introducendo ad esempio nuovi nodi e connessioni.

Il dataset utilizzato è generato dagli input dei sensori delle macchine. I sensori sono tutti proiettati in avanti dal muso della macchina, sono distanziati gli uni dagli altri di un certo grado, e si estendono per 200 unità di lunghezza, come è possibile vedere in figura 1.

Per calcolare la distanza di ogni raggio partendo dal sensore sulla macchina e arrivando alla collisione con il bordo si utilizza la formula di distanza euclidea:

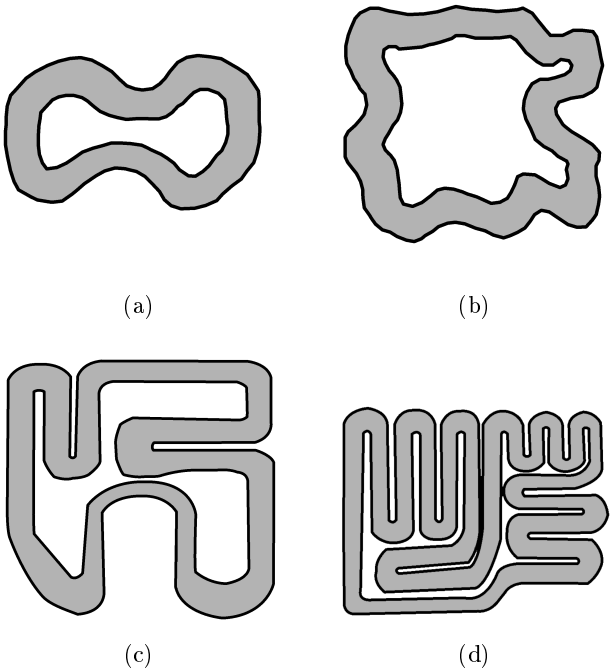
$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

dove x_1, y_1 e x_2, y_2 sono le coordinate di sensore e ostacolo, rispettivamente.

Per quanto riguarda i percorsi, sono stati creati quattro percorsi di allenamento con difficoltà crescente:

1. **Percorso A**: un circuito a forma di otto rovesciato, che introduce l'auto a delle semplici svolte.
2. **Percorso B**: un circuito più complesso con numerose curve a destra e sinistra, ma senza svolte troppo brusche.
3. **Percorso C**: un circuito con alcune curve ad U e passaggi molto stretti.
4. **Percorso D**: un circuito molto complesso con molte curve brusche ad U e diversi passaggi stretti.

Questi diversi percorsi sono visibili nella figura 2. Nella figura 3 invece, è mostrato il percorso di validazione usato per testare le capacità delle reti allenate, e il cui superamento ha rappresentato la sfida principale a cui si è puntato con questo lavoro.



(a) (b)

(c) (d)

Figura 2: I quattro percorsi di allenamento (a, b, c, d).

La fitness di ogni auto viene calcolata sulla base di diversi criteri:

- **Sopravvivenza**: maggiore è il tempo trascorso senza incidenti, maggiore è il suo punteggio. In

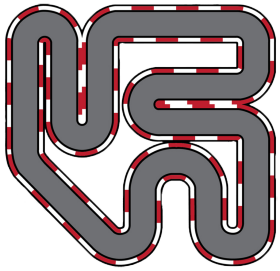


Figura 3: Il percorso di validazione.

particolare ad ogni tick del ciclo di gioco, viene assegnato un punto aggiuntivo alla fitness.

- **Distanza percorsa**: vengono utilizzati dei checkpoint per misurare la distanza percorsa lungo il circuito, con un punteggio maggiore assegnato per una distanza maggiore. Ad ogni tick di gioco viene ricalcolato il checkpoint più lontano raggiunto da ogni auto.
- **Superamento**: ogni volta che un auto supera uno dei percorsi di allenamento, gli viene dato un grande incremento nella fitness.
- **Guida sicura**: premia le auto che rimangono più lontane dai bordi durante la guida. Per farlo, viene calcolata la media delle distanze rilevate dai sei sensori rispetto ai bordi della strada. Un valore maggiore indica un'auto più al centro della strada, e meno vicina ai bordi, cosa che le rende più difficile andare a sbattere e fornisce alla rete più tempo per prendere una decisione di svolta.

La fase di riproduzione in NEAT è cruciale poiché è qui che avviene la generazione di nuove reti neurali, che potenzialmente porteranno a prestazioni migliorate. Questa fase si suddivide essenzialmente in due sotto-fasi:

- **Incrocio**: due reti neurali (genitori) vengono scelte dalla popolazione esistente, tipicamente in base alla loro fitness. Reti con fitness più alta hanno maggiori probabilità di essere selezionate per la riproduzione. Le strutture neurali dei genitori vengono combinate per creare una nuova rete (figlio). In NEAT, ogni gene (che rappresenta una connessione o un nodo nella rete neurale) ha un identificatore unico. Durante l'incrocio, i geni con lo stesso identificatore nei genitori vengono accoppiati, mentre i geni unici (presenti solo in uno dei genitori) possono essere ereditati a seconda di specifiche regole, come l'ereditarietà dal genitore più "fit".
- **Mutazione**: dopo l'incrocio, la nuova rete può subire mutazioni. Queste possono includere variazioni casuali dei pesi delle connessioni, l'aggiunta di nuove connessioni tra nodi precedentemente non connessi, o l'aggiunta di nuovi nodi (spesso suddividendo una connessione esistente). La mutazione introduce variazioni che possono portare a nuove soluzioni e strategie nel comportamento della rete neurale. È un meccanismo chiave per esplorare lo spazio delle soluzioni e prevenire l'omogeneità della popolazione.

Un altro aspetto importante in NEAT è la **speciazione**, un processo che raggruppa reti neurali simili in *specie*. Questo raggruppamento si basa sulla somiglianza topologica delle reti, cioè su quanto le loro strutture sono simili tra loro. Il principale obiettivo della speciazione è proteggere le innovazioni genetiche. Infatti, quando una rete neurale subisce mutazioni significative, potrebbe non essere immediatamente competitiva con reti più mature e ottimizzate, con un'alta probabilità di rimanere indietro. Collocandola in una nuova specie, ha maggiori possibilità di sopravvivere e migliorare prima di dover competere con l'intera popolazione. In altre parole, la speciazione contribuisce a mantenere la diversità genetica all'interno della popolazione, prevenendo la convergenza prematura verso soluzioni sub-ottimali (raggiungimento di un minimo locale).

Per raggruppare le reti in specie, NEAT calcola la distanza genetica tra i genomi. Questa distanza è una misura di differenza basata su tre fattori principali:

- **Eccesso di geni**: geni presenti in una rete ma non nell'altra.
- **Geni disgiunti**: geni che non si sovrappongono nell'ordine di comparizione tra le due reti.
- **Differenze di peso**: variazioni nei pesi delle connessioni corrispondenti tra le due reti.

La *distanza di compatibilità* δ di diverse strutture viene calcolata tramite una combinazione lineare del numero di geni in eccesso E e di quelli disgiunti D, assieme alla differenza media di pesi $\overline{W}$ nei geni che coincidono:

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot \overline{W}$$

dove i coefficienti c_1, c_2 e c_3 consentono di regolare l'importanza dei tre fattori, mentre il fattore N (numero di geni nel genoma più grande), effettua una normalizzazione per dimensione del genoma. Le reti

neurali vengono assegnate ad una specie se la loro distanza genetica da almeno una rete "rappresentante" della specie è inferiore a una soglia predeterminata, detta *soglia di compatibilità* δ_t. Se una rete non rientra in nessuna specie esistente, ne viene creata una nuova. Ogni specie viene mantenuta attraverso più generazioni, permettendo alle reti in quella specie di competere principalmente tra loro, piuttosto che con l'intera popolazione.

Come meccanismo di riproduzione in NEAT si utilizza la *fitness esplicita condivisa*, secondo cui organismi nella stessa specie devono condividere la fitness degli altri organismi di quella specie. Questo viene fatto in modo che una specie non diventi troppo grande nel caso in cui molti dei sui organismi abbiano buone prestazioni, e quindi per evitare che una singola specie domini l'intera popolazione, che andrebbe contro il funzionamento corretto dell'evoluzione per speciazione. Per ogni organismo i viene calcolata una fitness adattata f'_i, calcolata sulla base della distanza δ da ogni altro organismo j nella popolazione:

$$f'_i = \frac{f_i}{\sum_{j=1}^{n} \mathrm{sh}(\delta(i,j))}$$

dove la funzione sh vale 0 quando la distanza $\delta(i,j)$ supera la soglia δ_t, mentre vale 1 altrimenti. Pertanto, il termine $\sum_{j=1}^{n} \mathrm{sh}(\delta(i,j))$ conta il numero di organismi nella stessa specie dell'organismo i. Ad ogni specie viene assegnato un numero potenzialmente diverso di membri in proporzione alla somma delle fitness adattate f'_i dei suoi organismi.

Oltre alla speciazione, un altro punto di forza di NEAT rispetto ad altri algoritmi evolutivi è che nuove aggiunte strutturali sono svolte in maniera incrementale a partire da una struttura minimale. La ricerca parte da una popolazione uniforme formata da reti con zero nodi nascosti (tutti gli input sono collegati a tutti gli output). In questo modo, le nuove modifiche strutturali che occorrono in NEAT sono sempre giustificate. Inoltre, la minor dimensionalità dello spazio lo porta ad essere più efficiente di altri metodi che invece partono da topologie non minimali.

3 Risultati sperimentali

In questa sezione, compariamo i risultati del modello con diversi parametri di configurazione, per osservare il loro effetto sulle prestazioni dell'algoritmo NEAT. Una volta trovata la migliore configurazione per l'algoritmo si proverà ad eseguire l'algoritmo con i parametri ottimali trovati, prima direttamente sul percorso più difficile (percorso D in figura 2d), e poi sulla serie di percorsi con difficoltà graduale vista in figura 2, per osservare se l'utilizzo dell'apprendimento basato su curriculum porti a delle prestazioni migliori o se, al contrario, basti usare da subito il percorso più complesso per arrivare ad ottenere buoni risultati. L'obiettivo non è solo trovare un set di parametri e percorsi che portino la macchina a superare il percorso di validazione, ma anche quelli che lo facciano nel minor numero possibile di generazioni.

Per il primo punto (trovare i parametri ottimali dell'algoritmo) si è valutato l'impatto che può avere il cambiare:

1. il tasso di mutazione dei pesi delle connessioni
2. il numero di macchine nella popolazione

Tabella 1: Parametri della simulazione

Parametro	**Valore**
Numero di simulazioni	3
Numero di generazioni	10
Numero di sensori	6
Tasso di mutazione dei pesi	10%, 30%, 50%, 70%, 90%
Numero di macchine	10, 20, 30, 50, 100

Nella tabella 1 sono mostrati i diversi parametri di simulazione usati negli esperimenti e nei due scenari. Per ciascuna variazione di algoritmo si è deciso di eseguire 3 simulazioni, prendendo poi la media generazione per generazione di queste 3. Questo è stato fatto per ridurre l'errore dovuto a componenti di casualità. Quando viene variato il numero di macchine, si mantiene il tasso di mutazione all'80%. Quando si varia il tasso di mutazione, viene mantenuto il numero di macchine a 30.

Nel primo scenario, si è valutato l'impatto che può avere il cambiare il tasso di mutazione di un peso della connessione, sostituendo quest'ultimo con un valore casuale durante il processo di mutazione. I risultati sono mostrati nella figura 4, che mostra l'andamento della fitness media della popolazione in funzione del numero di generazioni. Ogni linea rappresenta l'andamento per una certa probabilità di modifica del peso di connessione.

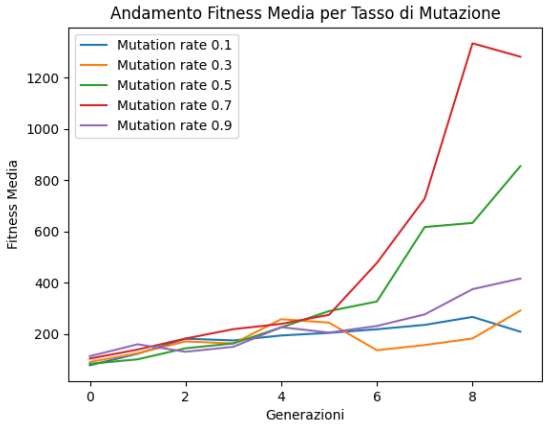


Figura 4: (Scenario 1) Prestazioni dell'algoritmo per diversi valori del tasso di mutazione dei pesi delle connessioni.

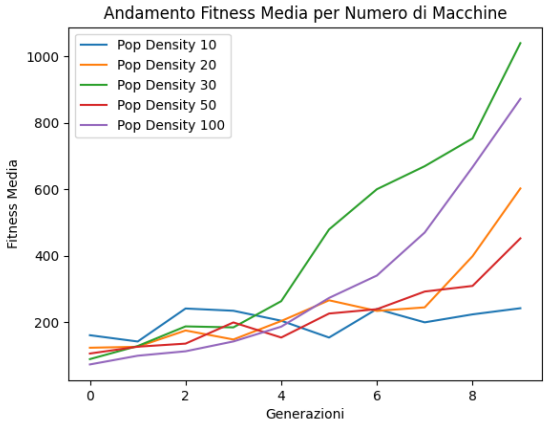


Figura 5: (Scenario 2) Prestazioni dell'algoritmo per diversi valori del numero di macchine.

Dagli esperimenti, si è arrivati alla conclusione che l'algoritmo ha risultati migliori quando i pesi delle connessioni della rete neurale hanno una probabilità del 70% di essere cambiati. Un valore maggiore porterebbe invece ad un peggioramento delle prestazioni, dovuto ad una quantità eccessiva di casualità.

Nel secondo scenario, si è voluta inoltre investigare la correlazione tra numero di membri della popolazione e prestazioni dell'algoritmo. I risultati di questi esperimenti sono mostrati nella figura 5, dove ogni linea mostra l'andamento della fitness media in funzione delle generazioni per diversi valori del numero totale di macchine.

Da questi esperimenti, si può concludere che il valore per il numero di macchine che porta al migliore risultato in termini di fitness media è 30, piuttosto che 100. Questo risultato è discordante dall'ipotesi fatta inizialmente, secondo qui aumentando in modo significativo il numero di macchine si raggiungono prestazioni migliori. Pertanto, non possiamo concludere che ci sia una correlazione significativa tra le due variabili.

Un'altra domanda a cui si voleva dare risposta in questo progetto è la seguente: può l'apprendimento basato su curriculum portare a delle prestazioni migliori per l'algoritmo? Per rispondere a questa domanda si è creato un terzo scenario, in cui viene eseguito l'algoritmo direttamente sul percorso più difficile, ossia il percorso D visto nella figura 2 (dopo il quale l'algoritmo solitamente sarebbe capace di superare il percorso di validazione).

Nello scenario presentato nella figura 4, dove era stato usato un approccio con apprendimento per curriculum, nell'arco di sole 10 generazioni si raggiunge una fitness media di 1500 con una scelta adeguata del tasso di mutazione. Invece, dalla figura 6a, che rappresenta l'andamento della fitness nello scenario senza apprendimento basato su curriculum, notiamo che anche dopo 50 generazioni, con stesso tasso di mutazione del 70% e stesso numero di macchine (e anche qui 3 simulazioni per ridurre le casualità), non si raggiungono le stesse prestazioni, e ciò lo si capisce anche dal fatto che, durante l'esecuzione, nessuna delle auto è in grado di avvicinarsi alla fine del percorso. Senza fornire degli esempi meno complessi all'algoritmo, il modello si blocca in un minimo locale e, raggiunto un certo punto del percorso, non riesce ad andare oltre. Il modello non ha cioè sufficiente esperienza per poter affrontare da subito un percorso come quello D in figura 2.

Un altro modo per capire in maniera visiva se l'algoritmo NEAT sta procedendo bene oppure no è tramite il grafico che mostra la dimensione delle specie per numero di generazioni, come in figura 6b, da cui si nota come la prima specie si estingua proprio in corrispondenza della generazione dove la fitness migliore raggiunge un punto piatto e non riesce ad aumentare ulteriormente.

Pertanto, si può concludere che per un problema di questo tipo che sfrutta NEAT come algoritmo evolutivo, l'apprendimento basato su curriculum (che consiste essenzialmente nel presentare sfide gradualmente più complesse all'algoritmo) porti a dei risultati migliori.

Questo esempio mette anche in luce una delle limitazioni di un algoritmo evolutivo come NEAT, cioè il fatto che non scala bene con l'aumentare della complessità. Non sempre si hanno a disposizione abbastanza dati da rendere possibile procedere tramite apprendimento per curriculum. In casi complessi in cui si ha poca esperienza pregressa, l'algoritmo può richiedere molte generazioni per convergere a una soluzione efficace. Ciò può renderlo meno efficiente per problemi complessi rispetto, ad esempio, all'apprendimento per rinforzo (RL).

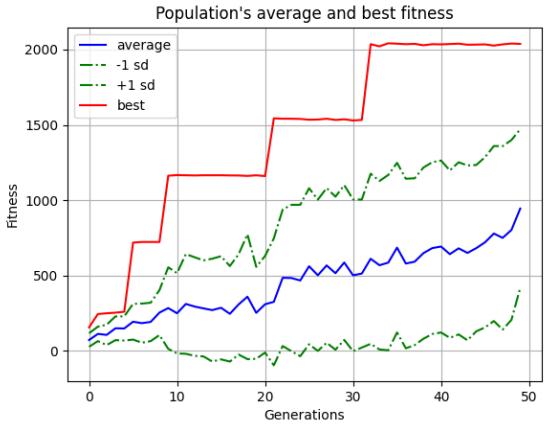
Tuttavia, NEAT è particolarmente semplice da implementare rispetto al RL, e può esplorare una varietà di strategie e comportamenti attraverso il suo processo evolutivo, il che può essere vantaggioso in certi scenari dove le strategie da adottare non siano immediatamente evidenti.

4 Conclusioni

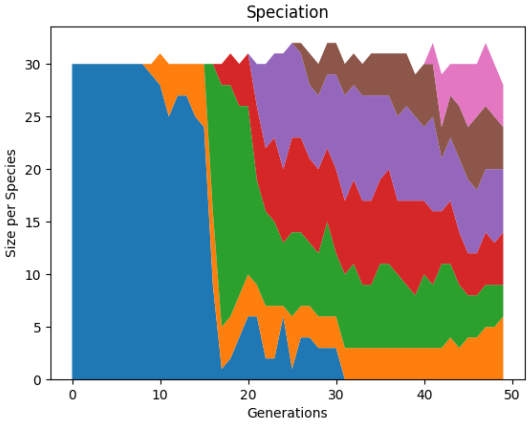
Questo progetto ha esplorato l'efficacia dell'algoritmo NEAT (NeuroEvolution of Augmenting Topologies) nella simulazione di guida autonoma, dimostrando come possa essere un'alternativa promettente agli approcci tradizionali di apprendimento automatico. Attraverso la realizzazione di un ambiente simulato 2D e l'impiego di diversi scenari di test, è stato dimostrato come NEAT sia capace di sviluppare reti neurali efficienti in grado di adattarsi dinamicamente a diverse sfide di navigazione.

Uno dei punti chiave emersi da questo studio è l'importanza dell'apprendimento basato su curriculum nel processo di addestramento. Si è osservato che, esponendo gradualmente l'algoritmo a compiti di crescente difficoltà, sono state migliorate significativamente le sue prestazioni, evitando le trappole dei minimi locali e promuovendo un apprendimento più robusto. Questo si è contrapposto all'approccio inizialmente adottato di immergere direttamente l'algoritmo in scenari complessi, che ha portato a prestazioni inferiori.

Da una prospettiva futura, ci sono diverse aree in cui questo lavoro potrebbe essere espanso. Uno sviluppo potrebbe essere l'integrazione di NEAT con tecniche di apprendimento per rinforzo per formare un approccio ibrido che potrebbe sfruttare i punti di forza di entrambi i metodi. Inoltre, l'impiego di risorse computazionali maggiori potrebbe consentire l'esplorazione di ambienti di simulazione più complessi e dettagliati, avvicinandosi maggiormente alle condizioni reali di guida autonoma (p.e. ambiente 3D, ostacoli, traffico, ecc.).



(a) Fitness media e migliore.



(b) Specie per generazione.

Figura 6: (Scenario 3) Prestazioni dell'algoritmo con 30 macchine, tasso di mutazione al 70%, senza apprendimento basato su curriculum.

References

[1] Kenneth O. Stanley e Risto Miikkulainen. "Evolving Neural Networks through Augmenting Topologies". In: *Evolutionary Computation* 10.2 (2002), pp. 99–127. DOI: 10.1162/106365602320169811 (cit. a p. 1).

[2] Y. Bengio et al. "Curriculum learning". In: vol. 60. Giu. 2009, p. 6. DOI: 10.1145/1553374.1553380 (cit. a p. 1).