UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA

Corso di Laurea Magistrale in Informatica

Frozen Lake

Relatore:
Ch.mo Prof.
Vincenzo DEUFEMIA

Candidata:
Rita CUCCARO
Mat. 0522501864

ANNO ACCADEMICO 2025/2026

CAPITOLO 1

INTRODUZIONE

Il progetto intitolato "**Applicazione del Reinforcement Learning e dei Large Language Models all'environment Frozen Lake**" ha come obiettivo lo sviluppo e l'analisi di diversi agenti intelligenti in grado di apprendere un comportamento ottimale all'interno di un ambiente simulato.

In particolare, il lavoro si concentra sull'utilizzo di tecniche di Reinforcement Learning (RL) applicate all'environment Frozen Lake, fornito dalla libreria `Gymnasium`, e sull'integrazione dei Large Language Models (LLM) come supporto al processo decisionale e di valutazione delle azioni.

L'idea centrale del progetto è studiare come un agente possa imparare, tramite esperienza diretta e interazione con l'ambiente, a raggiungere un obiettivo finale evitando situazioni sfavorevoli, e confrontare le prestazioni di:

- un agente basato esclusivamente su RL,
- un agente di RL guidato da un LLM nella scelta delle azioni,
- un agente di RL che utilizza un LLM come reward model.

CAPITOLO 2

STUDIO DELL'ENVIRONMENT

Frozen Lake è un ambiente appartenente alla categoria *Toy Text* della libreria `Gymnasium` ed è comunemente utilizzato per introdurre e sperimentare i concetti fondamentali del Reinforcement Learning (RL).

L'ambiente rappresenta un lago ghiacciato modellato come una griglia bidimensionale, all'interno della quale un agente deve muoversi partendo da una posizione iniziale con l'obiettivo di raggiungere una cella finale, evitando di cadere in celle pericolose che rappresentano dei buchi. La navigazione avviene su una mappa discreta, in cui ogni cella può assumere uno dei seguenti ruoli:

- **S (Start):** posizione iniziale dell'agente;
- **F (Frozen):** celle sicure sulle quali l'agente può muoversi senza penalità;
- **H (Hole):** celle pericolose che terminano l'episodio;
- **G (Goal):** cella obiettivo che conclude con successo l'episodio.

Il problema è formulato come una **sequenza di episodi finiti**, ciascuno dei quali termina al raggiungimento del goal o di uno stato terminale sfavorevole.

L'environment è caratterizzato da uno **spazio delle azioni discreto di dimensione 4**, che rappresenta le possibili direzioni di movimento dell'agente:

- **0:** sinistra;
- **1:** giù;
- **2:** destra;
- **3:** su;

Lo **spazio degli stati** è anch'esso discreto e rappresenta la posizione corrente dell'agente all'interno della griglia. Ogni stato è codificato come un valore intero secondo la relazione:

$$\text{stato} = \text{riga} \cdot n_{\text{colonne}} + \text{colonna}$$

Di conseguenza, lo stato iniziale è sempre lo stato 0, che corrisponde alla posizione [0,0] nella griglia.

Una caratteristica fondamentale dell'environment è la **natura scivolosa del terreno**, attivata impostando il parametro `is_slippery` a True. In questo caso, l'azione selezionata dall'agente non viene sempre eseguita in modo deterministico: con una certa probabilità l'agente può deviare verso una direzione perpendicolare rispetto a quella desiderata. La probabilità di eseguire correttamente l'azione scelta è regolata dal parametro `success_rate`. Questa componente introduce stocasticità nel problema, aumentando la complessità dell'apprendimento e richiedendo all'agente di sviluppare una politica robusta all'incertezza dell'ambiente.

Il sistema di **ricompense** predefinito dell'environment è semplice: una ricompensa pari a +1 viene assegnata quando l'agente raggiunge il goal mentre una ricompensa pari a 0 viene assegnata in tutti gli altri casi, sia quando l'agente si muove su una cella frozen sia quando cade in un buco.

L'episodio termina se l'agente raggiunge il goal, cade in un buco oppure supera il numero massimo di passi consentiti.

Nel progetto vengono considerate due configurazioni dell'ambiente:

1. **Mappa 4×4:** utilizzata come scenario iniziale per analizzare e validare il comportamento degli agenti in un contesto semplice;

2. **Mappa 8×8:** caratterizzata da uno spazio degli stati più ampio e da una maggiore complessità, impiegata per valutare la scalabilità e la robustezza degli approcci proposti.

Il confronto tra le prestazioni degli agenti sulle due mappe permette di capire come la dimensione dell'ambiente e la sua complessità influenzano l'apprendimento.

CAPITOLO 3

AGENTE BASATO SU RL

Il **Reinforcement Learning (RL)** è un paradigma di apprendimento automatico in cui un agente apprende come comportarsi all'interno di un ambiente attraverso l'interazione diretta con esso.

L'agente prende decisioni sotto forma di azioni, osserva le conseguenze delle proprie scelte e riceve un segnale di ricompensa, chiamato **reward**, che guida il processo di apprendimento.

L'obiettivo dell'agente è imparare una **policy ottimale**, cioè una strategia che massimizza la ricompensa cumulativa nel tempo. Nel contesto dell'environment Frozen Lake, ciò equivale a imparare un percorso che permetta di raggiungere il goal finale evitando i buchi e riducendo il numero di mosse effettuate.

Il problema può essere formalizzato attraverso i principali elementi del RL:

- **Ambiente:** Frozen Lake, modellato come una griglia con celle sicure, buchi e una posizione di arrivo.
- **Agente:** il giocatore che si muove all'interno della griglia.

- **Stati:** le possibili posizioni dell'agente nella griglia, rappresentate da valori interi.
- **Azioni:** i movimenti consentiti (su, giù, sinistra, destra).
- **Ricompense:** un segnale numerico che indica la qualità dell'azione eseguita.

L'interazione tra agente e ambiente avviene in modo iterativo: ad ogni passo l'agente osserva lo stato corrente, seleziona un'azione, riceve una reward e procede in un nuovo stato.

3.1 Algoritmo di apprendimento: Q-learning

Per l'implementazione dell'agente base è stato utilizzato l'algoritmo del **Q-learning**, una tecnica di RL model-free che consente all'agente di apprendere una policy ottimale senza conoscere a priori il modello di transizione dell'ambiente.

Il Q-learning si basa sulla stima della **funzione Q(s, a)**, che rappresenta il valore atteso dell'eseguire un'azione a in uno stato s e seguire successivamente la policy ottimale.

L'agente memorizza questi valori all'interno di una **Q-table**, che è una struttura dati di dimensione pari al numero di stati per il numero di azioni.

L'aggiornamento della Q-table avviene secondo la seguente regola:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

dove α è il learning rate, che controlla la velocità di aggiornamento, γ è il discount factor (fattore di sconto), che bilancia l'importanza delle ricompense future, r è la ricompensa ottenuta dopo l'esecuzione dell'azione e s' è lo stato successivo.

Durante l'addestramento, l'agente deve bilanciare due aspetti fondamentali quali:

- **Exploration:** la scelta di azioni casuali per scoprire nuovi percorsi;

- **Exploitation:** la scelta delle azioni che risultano migliori secondo la Q-table appresa.

A tal fine viene utilizzata una strategia **ϵ-greedy**, che seleziona con probabilità ϵ un'azione casuale e con probabilità **1-ϵ** l'azione con il valore Q più alto nello stato corrente.

Nel corso dell'addestramento, il valore di ϵ viene progressivamente ridotto, favorendo inizialmente l'esplorazione e successivamente lo sfruttamento delle conoscenze acquisite.

3.2 Implementazione

L'agente di RL apprende una policy di navigazione ottimale in Frozen Lake mediante interazioni successive con l'ambiente, sfruttando il feedback fornito dal sistema di ricompense per migliorare progressivamente le proprie decisioni.

Il problema viene modellato come un **Processo Decisionale di Markov (MDP) discreto**, in cui l'agente osserva lo stato corrente, seleziona un'azione tra quelle disponibili e riceve una ricompensa che guida il processo di apprendimento. L'implementazione supporta due algoritmi di apprendimento tabellare, Q-learning e SARSA, consentendo il confronto tra un approccio off-policy e uno on-policy all'interno dello stesso framework sperimentale.

L'environment, come abbiamo già detto, viene istanziato tramite la libreria `Gymnasium` e configurato selezionando la dimensione della mappa e la presenza di dinamiche stocastiche nei movimenti dell'agente.

Lo spazio degli stati è discreto e rappresenta tutte le possibili posizioni dell'agente sulla griglia, mentre lo spazio delle azioni comprende i movimenti consentiti all'interno dell'ambiente. Su questa base viene inizializzata una Q-table di dimensione pari al numero di stati per il numero di azioni, inizialmente impostata a zero per tutte le coppie stato–azione.

La selezione delle azioni avviene mediante una strategia ϵ-greedy, che consente di bilanciare exploration ed exploitation. Nelle fasi iniziali dell'adde-

stramento l'agente privilegia l'exploration, selezionando frequentemente azioni casuali, mentre con il progredire degli episodi il valore di ϵ viene progressivamente ridotto fino a un valore minimo prefissato. Questo decadimento controllato dell'esplorazione permette all'agente di acquisire una conoscenza iniziale dell'ambiente e successivamente di consolidare le azioni più vantaggiose apprese.

L'addestramento è organizzato in episodi indipendenti, il cui numero viene scelto in base alla complessità dell'ambiente. Per la mappa 4×4 vengono utilizzati 8.000 episodi di addestramento mentre per la mappa 8×8 il numero di episodi viene aumentato a 50.000, al fine di consentire un'esplorazione adeguata di uno spazio di stati molto più ampio e di gestire l'elevata probabilità di fallimento tipica di questo ambiente. In entrambi i casi, il fattore di decadimento di ϵ è scelto in modo coerente con il numero di episodi, così da mantenere un equilibrio tra exploration iniziale ed exploitation finale.

Durante ogni episodio, l'agente interagisce con l'ambiente fino al raggiungimento di uno stato terminale, che può corrispondere al successo del compito o a una condizione di fallimento. La Q-table viene aggiornata a ogni passo utilizzando una regola di apprendimento incrementale che combina la ricompensa immediata con la stima del valore futuro, pesate rispettivamente dal tasso di apprendimento e dal fattore di sconto. Nel caso del **Q-learning**, l'aggiornamento è di tipo **off-policy** e utilizza la migliore azione possibile nello stato successivo, mentre nel caso del **SARSA** l'aggiornamento è **on-policy** e si basa sull'azione effettivamente selezionata secondo la policy corrente. Questa differenza consente di osservare comportamenti di apprendimento differenti, soprattutto in presenza di transizioni stocastiche.

Al termine di ciascun episodio, il valore di ϵ viene aggiornato secondo il fattore di decadimento stabilito, mantenendo comunque un valore minimo per evitare una completa eliminazione dell'esplorazione. Contemporaneamente, viene registrata la ricompensa totale ottenuta nell'episodio, che viene utilizzata per analizzare l'andamento dell'apprendimento nel tempo.

Una volta completata la fase di addestramento, la policy appresa viene

valutata in una fase di test separata. Durante il test, l'agente opera in modalità completamente greedy, selezionando sempre l'azione con il valore Q più elevato. Il numero di esecuzioni di test viene adattato alla complessità della mappa: per la configurazione 4×4 vengono effettuate 100 esecuzioni indipendenti, mentre per la mappa 8×8 il test viene ripetuto 2.000 volte, così da ottenere una stima più robusta delle prestazioni. La metrica principale utilizzata è il `success_rate`, definito come la percentuale di episodi conclusi con il raggiungimento dello stato finale.

A supporto dell'analisi sperimentale vengono presentati alcuni grafici, per comprendere i risultati ottenuti.

3.2.1 Analisi dei risultati

In questa sezione vengono analizzati i risultati ottenuti dagli agenti basati su Q-learning e SARSA nell'environment, considerando sia la mappa 4×4 sia la 8×8. L'analisi si basa sull'osservazione dei grafici relativi alla ricompensa, al success rate, al numero di passi necessari per completare l'episodio e alla distribuzione finale dei valori nella Q-table.

Iniziamo, quindi, considerando la mappa 4x4.

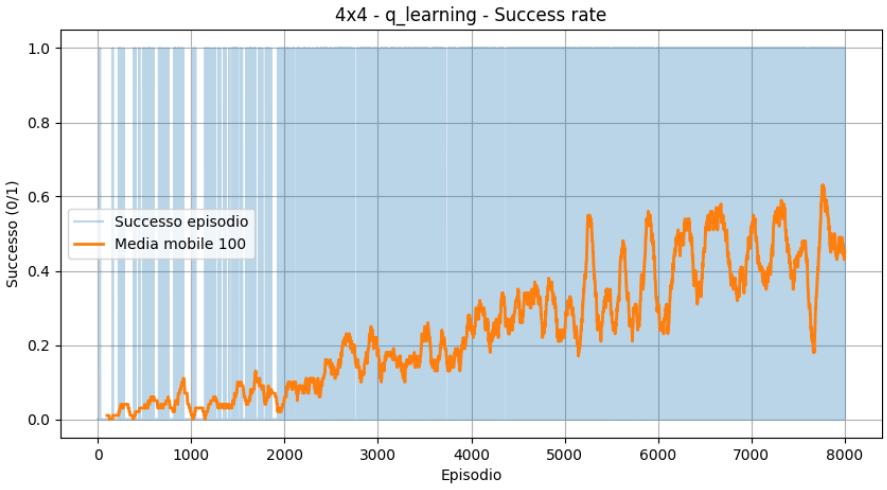

Figura 3.1: Success rate Q_learning mappa 4x4

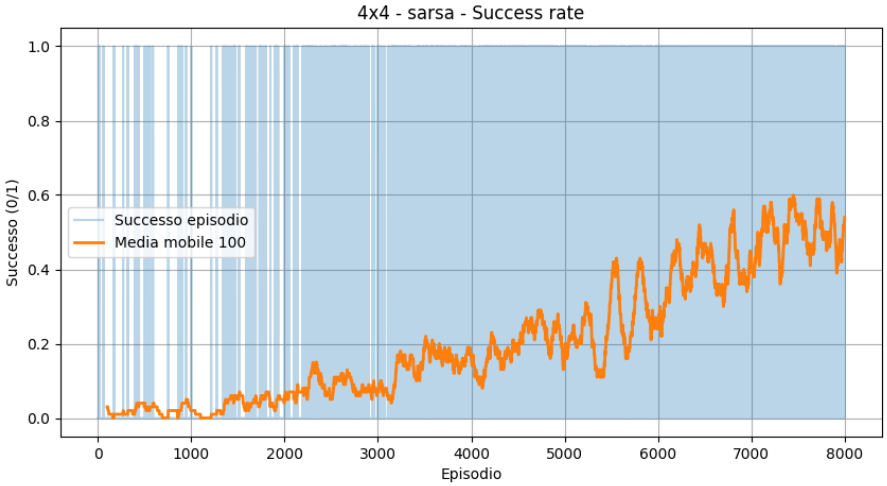

Figura 3.2: Success rate SARSA mappa 4x4

Le Figure 3.1 e 3.2 mostrano l'andamento del success rate durante l'addestramento degli algoritmi Q-learning e SARSA. In entrambi i casi si osserva una fase iniziale, che si estende per circa 1.500 episodi, in cui il success rate rimane prossimo allo zero. Questa fase è dominata dall'esplorazione: l'agente seleziona prevalentemente azioni casuali e riesce a raggiungere il goal solo occasionalmente.

A partire da circa 2.000 episodi, la curva della media mobile del success rate inizia a crescere in maniera evidente, segnalando che l'agente ha cominciato a individuare una strategia di navigazione più efficace. In questa fase emergono differenze tra i due algoritmi: il Q-learning mostra un andamento più stabile e raggiunge valori di successo leggermente superiori al 60%, mentre SARSA presenta una crescita più irregolare e oscillante. Questo comportamento è coerente con la natura on-policy di SARSA, che tende a mantenere policy più conservative e risente maggiormente della stocasticità dell'ambiente.

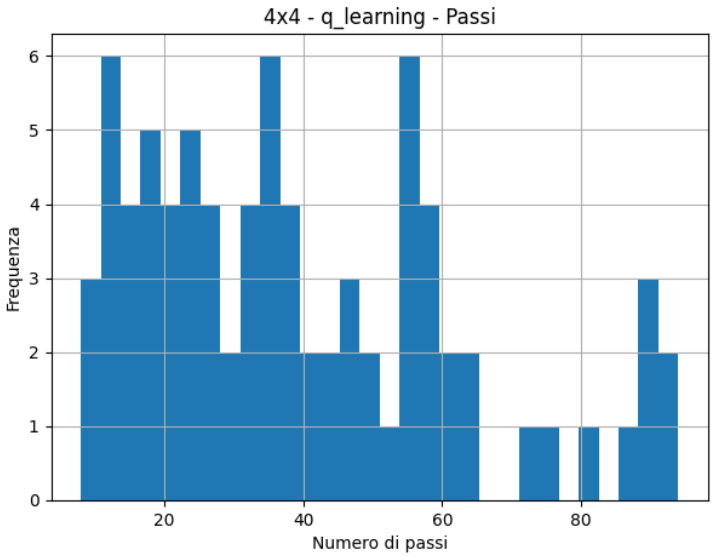

Figura 3.3: Distribuzione numero passi Q_learning mappa 4x4

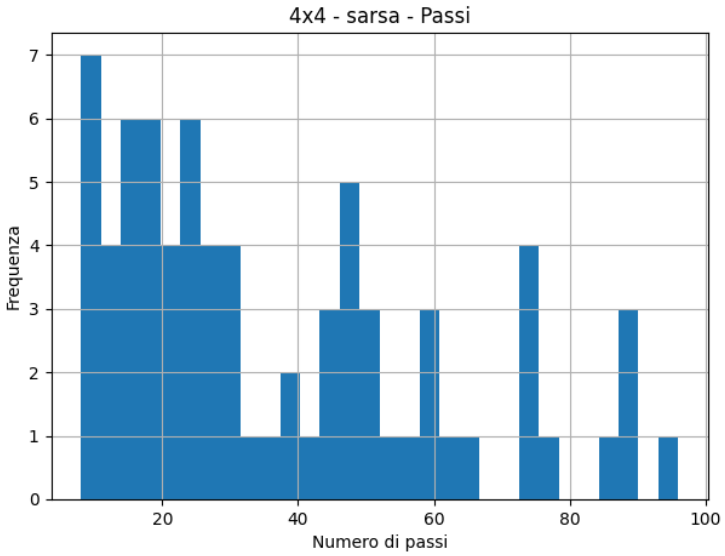

Figura 3.4: Distribuzione numero passi SARSA mappa 4x4

Le Figure 3.3 e 3.4 mostrano gli istogrammi del numero di passi necessari per raggiungere lo stato terminale, offrendo una visione più approfondita sull'efficienza delle policy apprese dai due algoritmi. Nel caso del Q-learning, la distribuzione risulta più dispersiva e presenta una struttura multimodale: sebbene una parte consistente degli episodi si concentri tra i 20 e i 40 passi, è

presente una coda lunga che si estende verso valori più elevati, fino a episodi con un numero di passi molto maggiore. Questo comportamento evidenzia una strategia maggiormente esplorativa, che può condurre l'agente a percorsi più lunghi o a cicli prima di convergere verso il goal.

Al contrario, SARSA mostra una distribuzione nettamente più concentrata, con un picco principale tra i 10 e i 20 passi. La ridotta dispersione indica che l'agente tende a seguire percorsi più stabili e consolidati, limitando l'esplorazione di traiettorie rischiose. Questo risultato è coerente con la natura on-policy dell'algoritmo, che privilegia comportamenti più conservativi e prevedibili una volta individuata una strategia efficace.

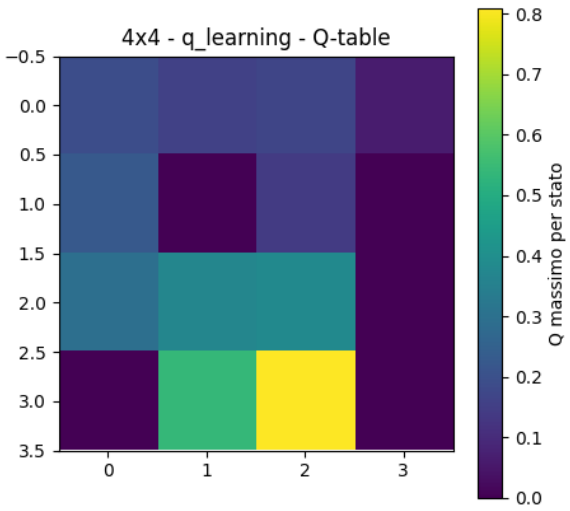

Figura 3.5: Q-table Q_learning mappa 4x4

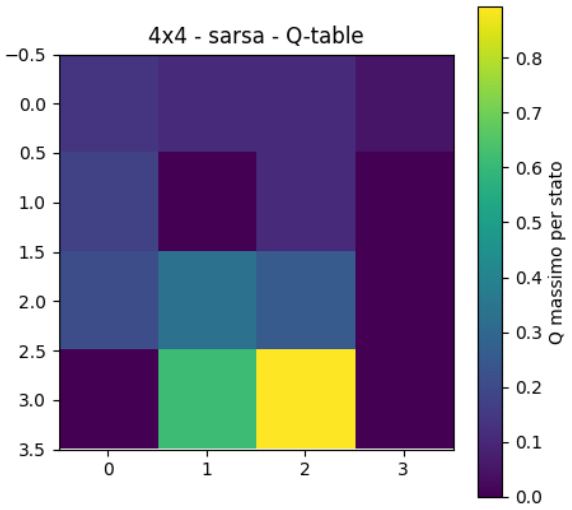

Figura 3.6: Q-table SARSA mappa 4x4

Le mappe di calore delle Q-table (Figure 3.5 e 3.6) rappresentano la distribuzione spaziale dei valori di Q appresi dagli agenti. I colori più chiari, come il giallo, indicano stati con valori elevati, tipicamente situati in prossimità del goal, mentre i colori scuri (viola) corrispondono a stati con valori bassi, associati a celle pericolose o poco vantaggiose.

In entrambe le mappe è visibile un percorso preferenziale che attraversa la parte centrale della griglia e conduce all'angolo in basso a destra, segno che sia Q-learning sia SARSA hanno identificato una strategia efficace per raggiungere l'obiettivo. Le differenze tra le due Q-table sono minime e riguardano principalmente l'intensità dei valori nelle zone adiacenti ai pericoli.

Adesso passiamo a considerare la mappa 8x8.

Figura 3.7: Success rate Q_learning mappa 8x8

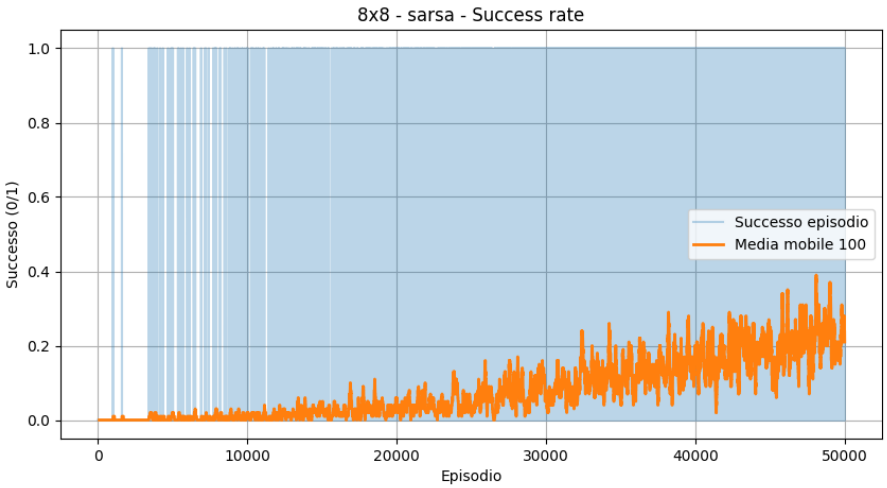

Figura 3.8: Success rate SARSA mappa 8x8

Le Figure 3.7 e 3.8 mostrano l'andamento del success rate nella mappa 8×8, evidenziando un processo di apprendimento significativamente più lento rispetto al caso 4×4. Nelle fasi iniziali dell'addestramento, fino a circa 10.500 episodi, le curve delle medie mobili rimangono vicine allo zero. In questa fase entrambi gli algoritmi sono dominati dall'esplorazione casuale e i rari episodi di successo osservabili sono da attribuire a eventi casuali.

A partire da circa 10.500 episodi, si osserva una graduale crescita del success rate, segnale che gli agenti iniziano a individuare percorsi più efficaci

verso il goal. In questa fase il Q-learning mostra una crescita leggermente più rapida e decisa, mentre SARSA presenta un andamento più cauto e progressivo, coerente con la sua natura.

Nelle fasi avanzate dell'addestramento, avvicinandosi ai 50.000 episodi, le prestazioni di entrambi gli algoritmi tendono a stabilizzarsi. Il Q-learning raggiunge un success rate compreso tra il 40% mentre SARSA si mantiene su valori inferiori, generalmente intorno al 30–35%. Le oscillazioni residue delle curve sono dovute alla persistenza di una componente di esplorazione e alla natura stocastica dell'ambiente, che rendono il raggiungimento del goal non sempre garantito anche in presenza di una policy consolidata.

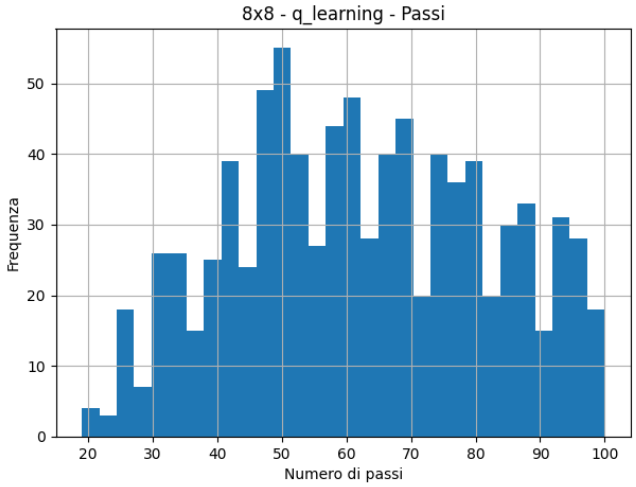

Figura 3.9: Distribuzione numero passi Q_learning mappa 8x8

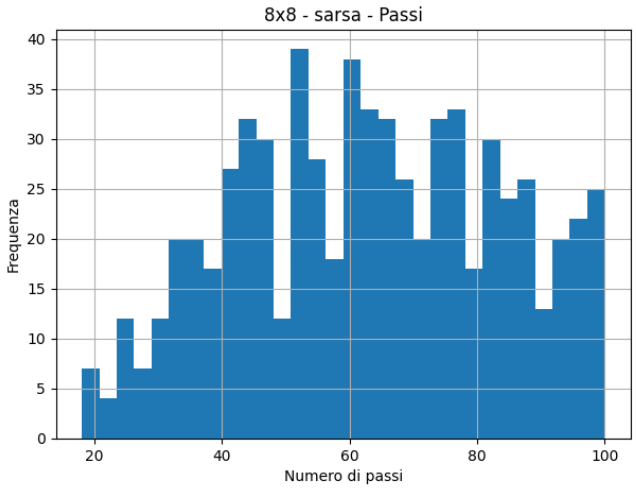

Figura 3.10: Distribuzione numero passi SARSA mappa 8x8

Gli istogrammi del numero passi (Figure 3.9 e 3.10) misurano l'efficienza degli algoritmi contando quanti passi servono per arrivare al traguardo. Nel caso del Q-learning, la distribuzione ha un picco centrato tra i 45 e i 55 passi. Questo ci indica che l'algoritmo tende a privilegiare prestazioni medie superiori, puntando dritto alla soluzione ottimale. Essendo un approccio più "aggressivo", riesce a stabilizzarsi su percorsi molto brevi, concentrando la maggior parte dei risultati sui valori più bassi della scala.

Al contrario, SARSA mostra una distribuzione spostata verso destra. La maggior parte delle prove si attesta in un intervallo più ampio, tra i 50 e i 70 passi, presentando inoltre una coda molto più pronunciata che si allunga fino ai 100 passi. Questa forma del grafico ci suggerisce che SARSA compie percorsi mediamente più lunghi e meno efficienti sotto il profilo del tempo. Tuttavia, questa lentezza è il prezzo da pagare per una maggiore cautela: l'algoritmo preferisce traiettorie più conservative, garantendo una maggiore stabilità e sicurezza durante l'intero processo di apprendimento.

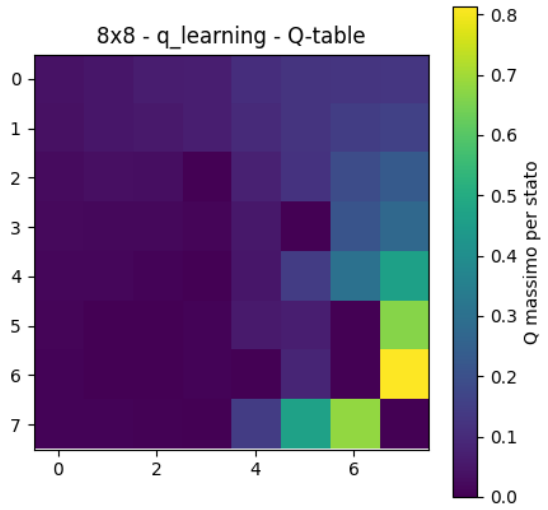

Figura 3.11: Q-table Q_learning mappa 8x8

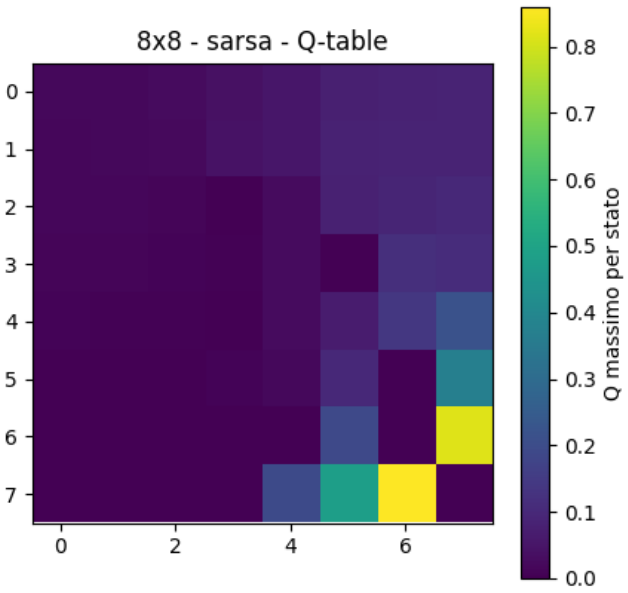

Figura 3.12: Q-table SARSA mappa 8x8

Le mappe di calore delle Q-table (Figure 3.11 e 3.12) confermano ulteriormente questa difficoltà: gran parte delle celle presenta valori vicini allo zero,

segno che ampie porzioni dell'ambiente non sono state esplorate o non hanno contribuito all'apprendimento. Solo le regioni prossime al goal mostrano valori elevati, suggerendo che gli agenti riescono a comportarsi in modo efficace solo quando partono da stati già vicini alla meta.

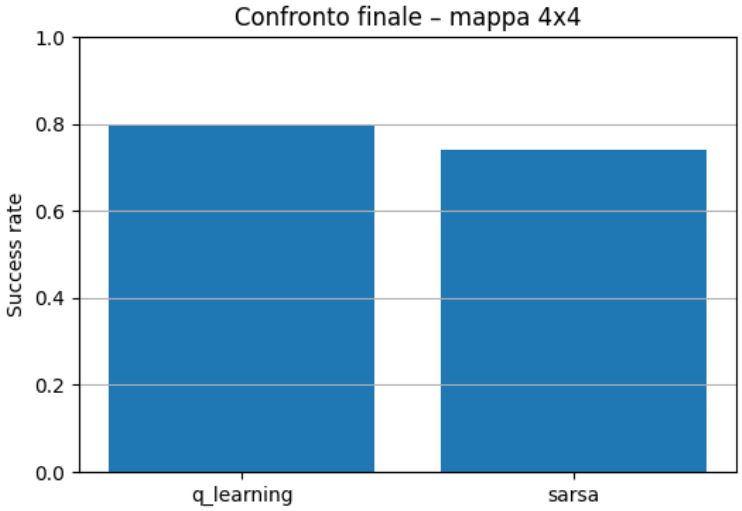

Figura 3.13: Confronto finale mappa 4x4

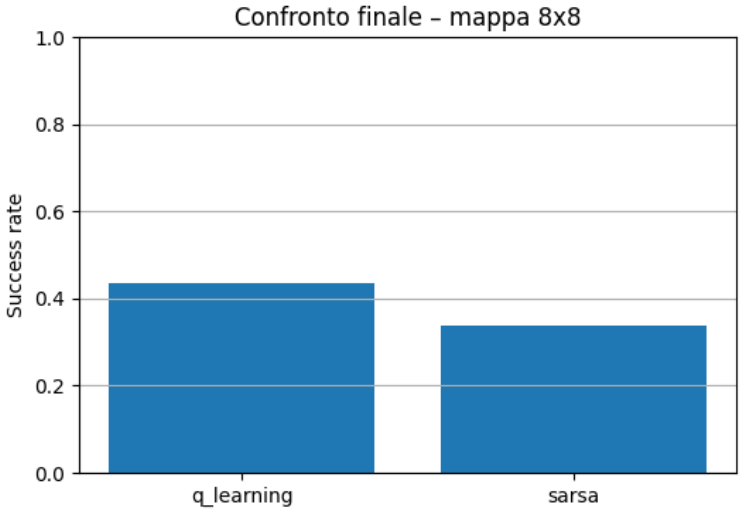

Figura 3.14: Confronto finale mappa 8x8

Infine, le Figure 3.13 e 3.14 mostrano il confronto finale tra i due algoritmi in termini di success rate, mettendo in evidenza l'impatto della complessità

dell'ambiente sulle prestazioni. Nella mappa 4×4, entrambi gli algoritmi ottengono risultati elevati. In questo scenario, il Q-learning raggiunge un success rate leggermente superiore rispetto a SARSA.

Il passaggio alla mappa 8×8 comporta invece una riduzione delle prestazioni per entrambi gli approcci. L'aumento della dimensione della griglia e la crescita dello spazio degli stati rendono l'esplorazione più complessa e aumentano la probabilità di fallimento, con success rate che scendono al di sotto del 50%. In questo contesto più sfidante emergono i limiti degli algoritmi tabellari, che faticano a generalizzare in ambienti più ampi e con ricompense sparse. Anche in questo caso, tuttavia, il Q-learning mantiene un vantaggio rispetto a SARSA, ottenendo un success rate leggermente più elevato e confermandosi l'algoritmo più efficace tra i due nel massimizzare il risultato finale.

CAPITOLO 4

AGENTE BASATO SU LLM

Oltre all'approccio basato esclusivamente sul RL, il progetto esplora l'integrazione dei **Large Language Models (LLM)** come supporto al processo decisionale dell'agente.

L'obiettivo è valutare se e in che misura un modello linguistico possa contribuire a migliorare l'apprendimento o il comportamento dell'agente all'interno dell'environment.

In particolare, sono state implementate due varianti dell'agente che incorporano l'uso di un LLM:

1. un agente in cui l'**LLM** genera direttamente l'**azione** da eseguire;
2. un agente in cui l'**LLM** agisce come **reward model**, valutando l'azione eseguita dall'agente di RL.

Questi approcci permettono di confrontare un apprendimento guidato esclusivamente dai segnali numerici con uno supportato da una forma di ragionamento semantico.

Analizziamo nel dettaglio le due varianti dell'agente con LLM.

4.1 LLM come generatore di azioni

Nella prima variante, il **LLM** viene utilizzato **come policy decisionale**. Ad ogni passo dell'episodio, lo stato corrente dell'ambiente viene convertito in una descrizione testuale che viene fornita in input al modello linguistico.

Il prompt descrive la posizione corrente dell'agente, la dimensione della mappa, le possibili azioni disponibili e l'obiettivo del gioco.

Sulla base di queste informazioni, l'LLM restituisce un'azione da eseguire, scelta tra quelle consentite dall'environment.

In questo scenario, l'agente non utilizza direttamente una Q-table per la selezione dell'azione, ma si affida al modello linguistico per decidere la mossa successiva, mantenendo comunque le regole e le dinamiche dell'ambiente.

Questo approccio consente di valutare la capacità dell'LLM di interpretare lo stato del gioco, pianificare una strategia locale e prendere decisioni coerenti con l'obiettivo finale.

4.1.1 Implementazione

In questa configurazione l'agente non apprende una policy tramite RL, ma delega direttamente la scelta delle azioni a un LLM, utilizzato come decisore esterno basato su conoscenza semantica e ragionamento simbolico. L'obiettivo di questa sezione è valutare quanto un LLM, pur non apprendendo direttamente dall'interazione con l'ambiente, sia in grado di guidare l'agente verso il goal in modo efficace, basandosi solo sulla descrizione dello stato corrente.

L'environment viene inizializzato in modalità stocastica, mantenendo la dinamica "slippery". Inoltre, vengono considerate sempre sia la mappa 4×4 che la 8×8, al fine di analizzare il comportamento del modello linguistico al crescere della complessità dell'ambiente. A differenza dell'agente RL classico, in questo caso non è presente alcuna Q-table né una fase di addestramento: ogni episodio è indipendente e l'agente non conserva l'esperienza passata sotto forma di valori appresi.

Per consentire al modello linguistico di prendere decisioni basate su informazioni comprensibili, lo stato numerico dell'ambiente viene trasformato in una rappresentazione più interpretabile. In particolare, la posizione dell'agente viene convertita in coordinate bidimensionali sulla griglia e vengono identificate le celle adiacenti nelle quattro direzioni cardinali. Per ciascuna direzione viene fornita al modello un'informazione semantica sulla tipologia della cella adiacente, distinguendo tra celle sicure, buchi, goal e posizioni fuori mappa. Questa trasformazione permette di fornire al LLM una visione locale dello stato, coerente con il modo in cui un essere umano affronterebbe il problema.

Sulla base di queste informazioni viene costruito un prompt testuale che descrive esplicitamente la dimensione della mappa, la posizione corrente dell'agente e il contenuto delle celle adiacenti. Il prompt include inoltre istruzioni vincolanti che richiedono al modello di selezionare una sola azione tra quelle ammesse, evitando celle pericolose e privilegiando percorsi sicuri o diretti verso l'obiettivo finale. Il modello linguistico viene interrogato in modalità deterministica, così da garantire risposte riproducibili a parità di stato.

Per ridurre il costo computazionale e il numero di interrogazioni al modello, viene introdotto un **meccanismo di caching**. Le azioni suggerite dal LLM vengono memorizzate per ciascuno stato visitato, in modo tale che, se lo stesso stato viene incontrato nuovamente in episodi successivi, l'azione viene recuperata direttamente dalla cache senza una nuova chiamata al modello. Questo consente di migliorare l'efficienza dell'approccio, specialmente negli ambienti più grandi.

Nonostante l'uso del modello linguistico come decisore principale, viene comunque mantenuto un minimo livello di esplorazione tramite una strategia ϵ-greedy. Con una probabilità ridotta, l'azione viene scelta in modo casuale, al fine di evitare un comportamento completamente deterministico e mitigare eventuali errori sistematici del modello. Questo aspetto è rilevante in un ambiente stocastico come Frozen Lake, in cui la stessa azione può produrre esiti differenti.

La valutazione delle prestazioni avviene eseguendo un numero prefissato

di episodi indipendenti. Per la mappa 4×4 vengono considerati 500 episodi di valutazione, mentre per la mappa 8×8 il numero viene aumentato a 1.000, così da ottenere una stima più stabile delle prestazioni in un ambiente più complesso. Durante ogni episodio vengono registrati sia il successo finale, definito come il raggiungimento del goal, sia il numero di passi necessari per terminare l'episodio.

Inoltre, a supporto dell'analisi vengono presentati alcuni grafici per comprendere i risultati ottenuti.

4.1.1.1 Analisi dei risultati

Anche in questo caso l'analisi dei risultati viene divisa considerando prima la mappa 4x4 e poi la 8x8.

I risultati ottenuti sulla mappa 4×4 evidenziano come l'approccio basato sull'uso di un LLM come generatore diretto di azioni presenti forti limitazioni, anche in un ambiente semplice e di dimensioni ridotte.

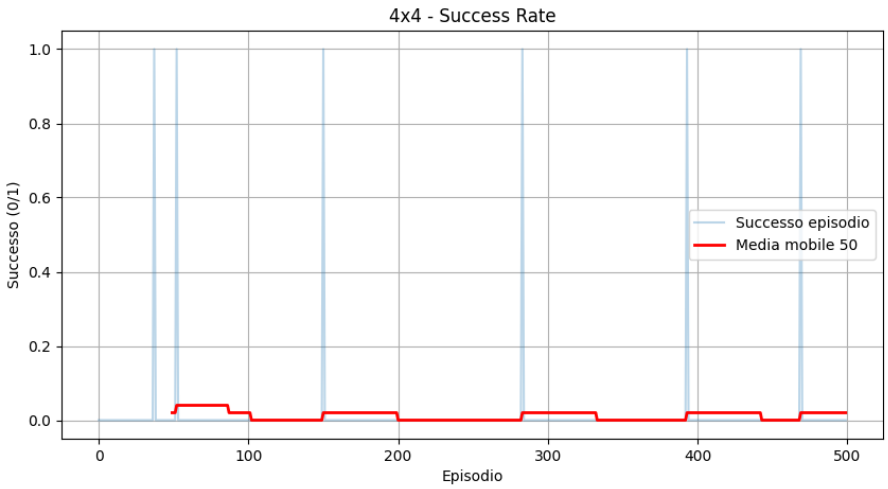

Figura 4.1: Success rate LLM action mappa 4x4

La Figura 4.1 mostra l'andamento del success rate nei 500 episodi di valutazione. I rari episodi di successo sono rappresentati da singoli picchi isolati, mentre la curva della media mobile rimane vicina allo zero per tutta la durata dell'esperimenti. Questo andamento indica l'assenza di qualsiasi

miglioramento delle prestazioni. Tale comportamento è coerente con la natura dell'approccio perchè il modello linguistico non dispone di un meccanismo di apprendimento dall'esperienza e non aggiorna una policy nel tempo, per cui non emerge nessuna dinamica assimilabile a un processo di apprendimento iterativo.

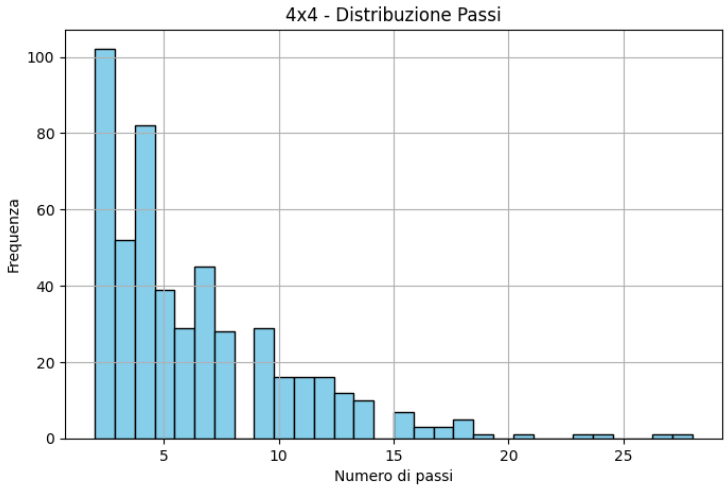

Figura 4.2: Distribuzione numero passi LLM action mappa 4x4

La Figura 4.2 riporta la distribuzione del numero di passi per episodio. L'istogramma evidenzia una forte concentrazione di episodi tra i 2 e 5 passi, questo indica che l'agente termina molto rapidamente la maggior parte delle esecuzioni a causa di una caduta nella cella di tipo hole. Questo risultato suggerisce una difficoltà del LLM nel pianificare anche sequenze di azioni brevi e sicure, nonostante la semplicità della mappa.

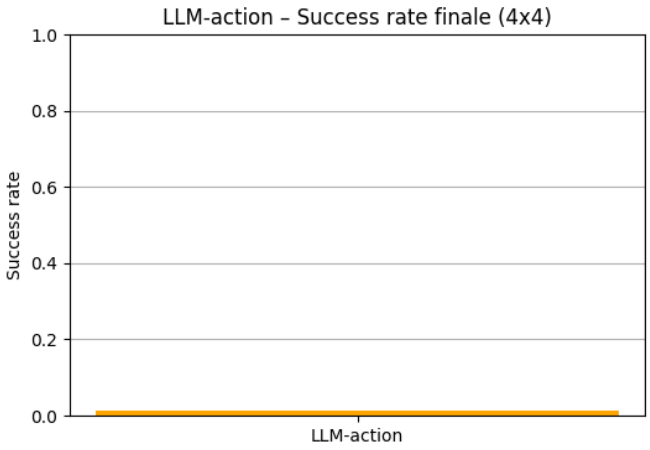

Figura 4.3: Success rate LLM action mappa 4x4

Il grafico di sintesi (Figura 4.3) mostra che il success rate medio è molto basso, con pochissimi episodi completati con successo. I rari successi sono quindi da considerarsi come eventi casuali.

Adesso passiamo ad analizzare i risultati della mappa 8x8.

L'aumento della dimensione della mappa introduce una complessità maggiore, e i risultati ottenuti evidenziano ulteriormente i limiti dell'approccio basato su LLM come policy decisionale.

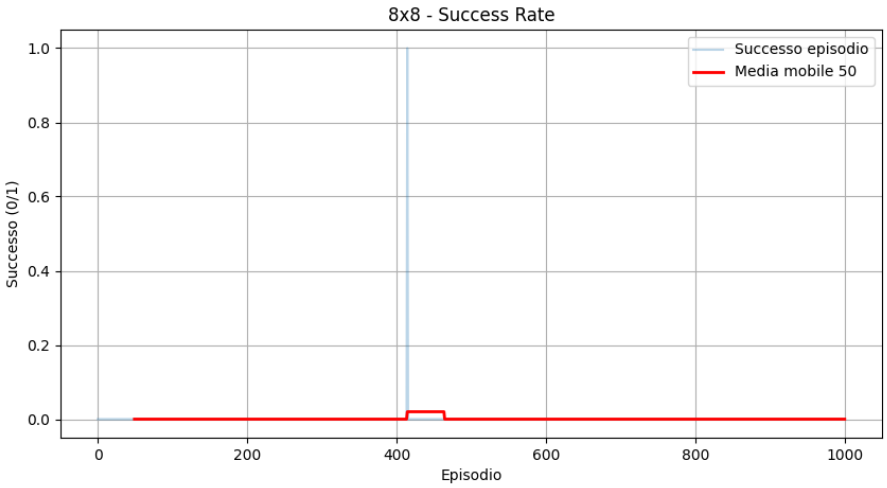

Figura 4.4: Success rate LLM action mappa 8x8

Nella Figura 4.4 è mostrato il success rate ottenuto su 1.000 episodi. Il

grafico evidenzia un solo episodio di successo isolato, mentre nel resto delle esecuzioni l'agente fallisce. Anche in questo caso la media mobile rimane appiattita sul valore zero, indicando che nonostante il numero di episodi sia aumentato questo non produce nessun miglioramento delle prestazioni. L'LLM non riesce quindi ad adattare il suo comportamento al crescere della complessità dell'ambiente.

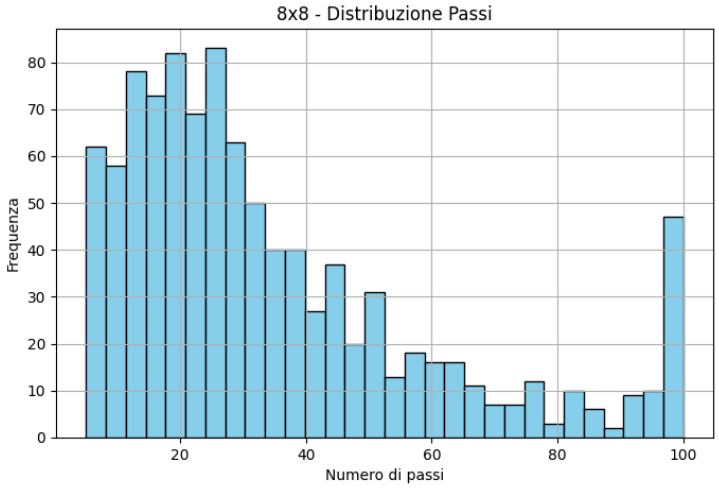

Figura 4.5: Distribuzione numero passi llm action mappa 8x8

La distibuzione dei passi (Figura 4.5) si estende su un intervallo molto più ampio rispetto al caso della mappa 4x4, arrivando fino al numero massimo di passi consentiti. È presente un picco intorno ai 20-30 passi, che indica come l'agente tenda a muoversi più a lungo prima di fallire. Inoltre, si può notare una concentrazione di episodi al valore massimo di passi, segnale che molti episodi terminano per timeout anzichè per il raggiungimento del goal o la caduta in una cella hole. Questo comportamento suggerisce che l'agente vaga spesso nell'ambiente senza una direzione chiara, evidenziando l'incapacità del modello di pianificare un percorso efficace verso l'obiettivo.

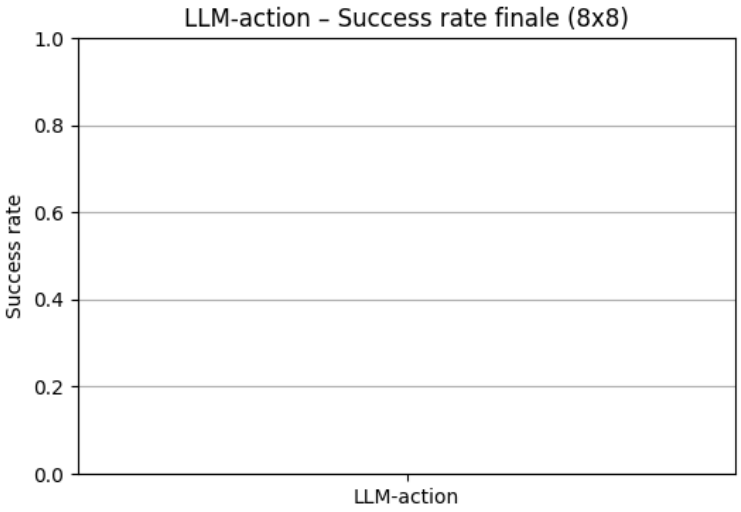

Figura 4.6: Success rate llm action finale mappa 8x8

Il grafico di sintesi (Figura 4.6) conferma che il success rate complessivo sulla mappa 8x8 è nullo. La barra associata al successo risulta assente, indicando che l'approccio fallisce completamente in ambienti di maggiore complessità e dimensionalità.

4.2 LLM come reward model

Nella seconda variante, l'**LLM** non sceglie direttamente l'azione, ma viene utilizzato come modello di valutazione delle azioni, ossia **come reward model**.

In questo approccio l'agente di RL continua a interagire normalmente con l'environment, selezionando le azioni in base alla propria policy. Ogni azione viene eseguita nell'ambiente e, a partire dallo stato corrente e dall'azione scelta, l'LLM viene coinvolto per fornire una valutazione qualitativa del comportamento dell'agente. Tale valutazione viene poi convertita in una ricompensa numerica, che si affianca al reward dell'environment e viene utilizzata durante la fase di aggiornamento della policy. In questo modo l'LLM non sostituisce il processo di apprendimento dell'agente, ma ne guida l'evoluzione fornendo un feedback aggiuntivo basato su una comprensione semantica dello stato.

Il modello linguistico valuta l'azione considerando criteri quali avvicinamento o allontanamento dal goal, rischio di cadere in un buco e la coerenza della mossa rispetto a un percorso sicuro.

La reward fornita dall'LLM sostituisce o integra la reward standard dell'environment, influenzando direttamente l'aggiornamento della Q-table dell'agente.

Questo approccio permette di introdurre una forma di **reward shaping** intelligente, in cui il segnale di ricompensa non è limitato a un valore binario, ma incorpora una valutazione più ricca e informata del comportamento dell'agente.

L'utilizzo dei LLM nel contesto del RL è motivato da diversi fattori:

- la possibilità di sfruttare la capacità di ragionamento e generalizzazione degli LLM;
- la riduzione della dipendenza da reward sparse, tipiche di ambienti come Frozen Lake;
- l'introduzione di un feedback più informativo durante l'apprendimento.

In particolare, l'approccio basato sul reward model risulta interessante poiché mantiene la struttura classica del RL, ma arricchisce il segnale di apprendimento con informazioni di alto livello fornite dal modello linguistico.

4.2.1 Implementazione

In questa configurazione, l'agente apprende tramite RL, ma la funzione di ricompensa viene arricchita con indicazioni fornite da un LLM. L'idea è combinare il meccanismo classico di apprendimento per rinforzo con la capacità del modello linguistico di valutare la sicurezza delle azioni, guidando così l'agente verso comportamenti più intelligenti senza modificare la struttura base del Q-learning.

L'environment Frozen Lake viene configurato in modalità stocastica, con dinamiche "slippery", e vengono considerate sia la mappa 4×4 che la 8×8, così da analizzare il comportamento dell'agente al crescere della complessità. La

Q-table viene inizializzata a zero e aggiornata tramite la regola del Q-learning standard, ma la ricompensa utilizzata in ogni passo è una combinazione tra il reward fornito dall'environment e un bonus proporzionale alla valutazione dell'LLM. In particolare, l'LLM suggerisce quale azione sarebbe più sicura nello stato corrente, e l'agente riceve un reward positivo se la sua scelta coincide con quella del modello. Tale reward LLM viene ponderato con un fattore minore rispetto alla ricompensa originale, così da influenzare l'apprendimento senza dominare completamente il segnale ambientale.

Per ridurre i tempi di calcolo, le valutazioni dell'LLM vengono memorizzate in una **cache**, in modo che azioni già valutate in stati precedentemente visitati non richiedano ulteriori interrogazioni al modello. L'agente continua comunque a selezionare azioni casuali con una probabilità ϵ, che decresce progressivamente con il numero di episodi, garantendo un equilibrio tra exploration ed exploitation.

Gli episodi di addestramento vengono scelti in funzione della complessità della mappa: per la 4×4 si utilizzano 3.000 episodi, mentre per la 8×8 si utilizzano 5.000 episodi, con un numero massimo di passi per episodio aumentato nella mappa più grande per consentire all'agente di esplorare adeguatamente lo spazio degli stati. Durante ogni episodio viene monitorato il successo nel raggiungere il goal, e viene registrata la storia delle ricompense per analizzare l'andamento dell'apprendimento.

Al termine dell'addestramento, la policy appresa viene valutata in una fase di test separata, eseguendo un numero prefissato di episodi indipendenti. Vengono calcolati il success rate e il numero medio di passi per episodio, fornendo una misura quantitativa dell'efficacia della policy.

Inoltre, vengono generati grafici che mostrano l'evoluzione del successo episodio per episodio, con una media mobile per evidenziare trend di apprendimento, e mappe di calore della Q-table, utili per interpretare quali stati vengono considerati più vantaggiosi dall'agente.

4.2.1.1 Analisi dei risultati

In questa sezione vengono analizzati i risultati ottenuti dall'agente che combina un approccio di RL con il supporto di un LLM. L'analisi parte dalla mappa 4×4 per poi passare alla mappa 8×8.

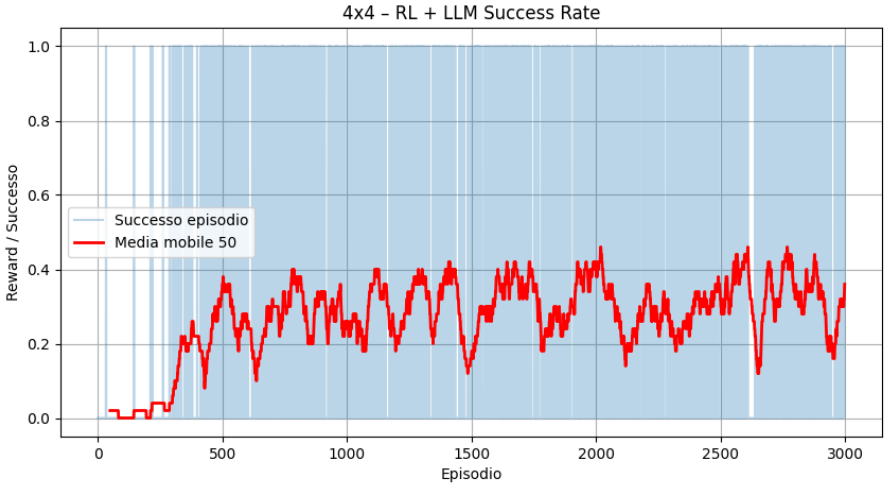

Figura 4.7: Success rate RL + LLM mappa 4x4

Nelle fasi iniziali dell'addestramento il success rate è vicino allo zero, segno che l'agente non ha ancora una strategia e si muove in modo esplorativo. Con l'aumentare del numero di episodi, la media mobile del successo inizia a crescere gradualmente. Intorno ai 300 episodi, l'agente comincia a raggiungere il goal con una certa regolarità, stabilizzandosi su un success rate compreso tra il 30% e il 40%. Questo andamento suggerisce che il contributo del modello linguistisco riesce a guidare l'esplorazione, favorendo più velocemente la scoperta di azioni vantaggiose rispetto a un agente basato solo su RL.

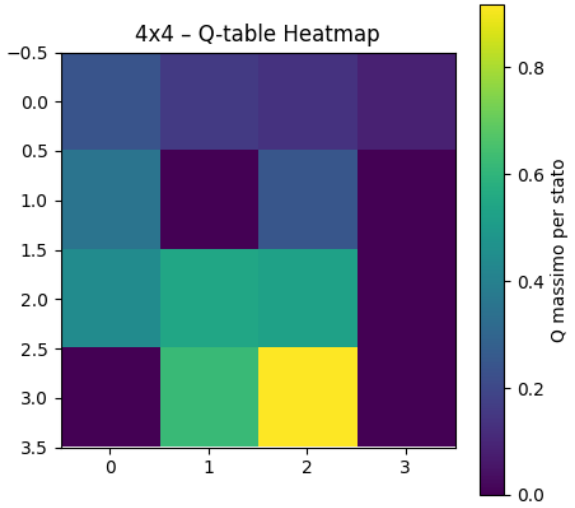

Figura 4.8: Q-table RL + LLM mappa 4x4

La heatmap della Q-table (Figura 4.8) mostra chiaramente quali stati l'agente considera migliori: quelli vicini al goal hanno valori alti, mentre buchi e percorsi rischiosi hanno valori bassi. La distribuzione non è casuale, ma segue la struttura della mappa, indicando che l'agente ha imparato a riconoscere gli stati importanti. In questo processo, il modello linguistico aiuta a suggerire azioni più sensate e coerenti con l'obiettivo.

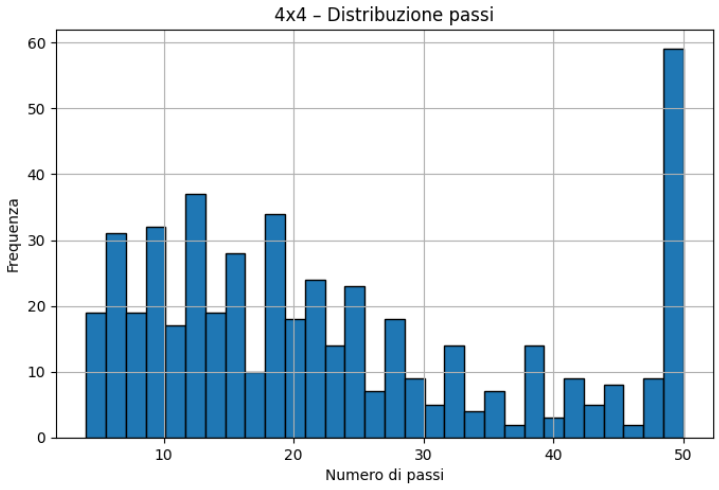

Figura 4.9: Distribuzione numero passi RL + LLM mappa 4x4

L'istogramma del numero di passi per episodio (Figura 4.9) mostra una distribuzione eterogenea. Molti episodi terminano in pochi passi, indicando fallimenti rapidi, mentre altri raggiungono il limite massimo consentito, segno che in alcune situazioni l'agente esplora a lungo senza riuscire a convergere verso il goal (comportamento classico del **compromesso tra exploration ed exploitation**).

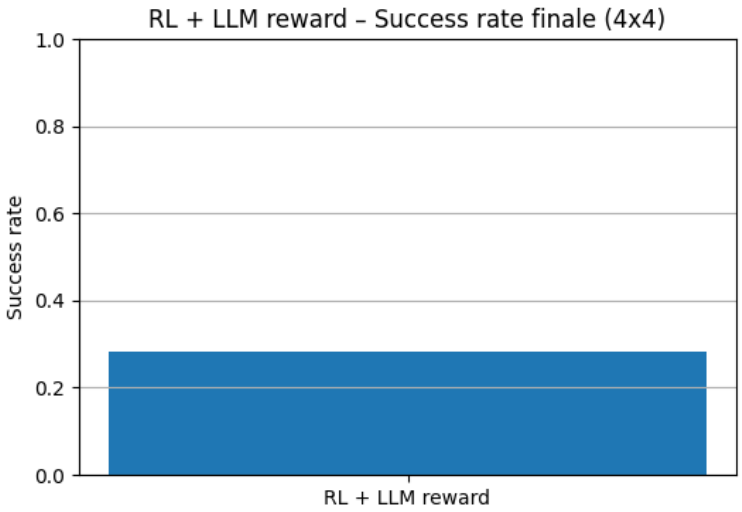

Figura 4.10: Success rate finale RL + LLM mappa 4x4

Il grafico riassuntivo del success rate finale (Figura 4.10) mostra un success rate medio pari a circa il 28%. Nonostante il risultato non sia ottimale, l'agente dimostra di aver appreso una policy maggiore rispetto a un approccio senza supporto semantico.

Il passaggio, invece, alla mappa 8x8 introduce una complessità maggiore, mettendo in evidenza i limiti di tale approccio in presenza di spazi di stato più ampi e ricompense sparse.

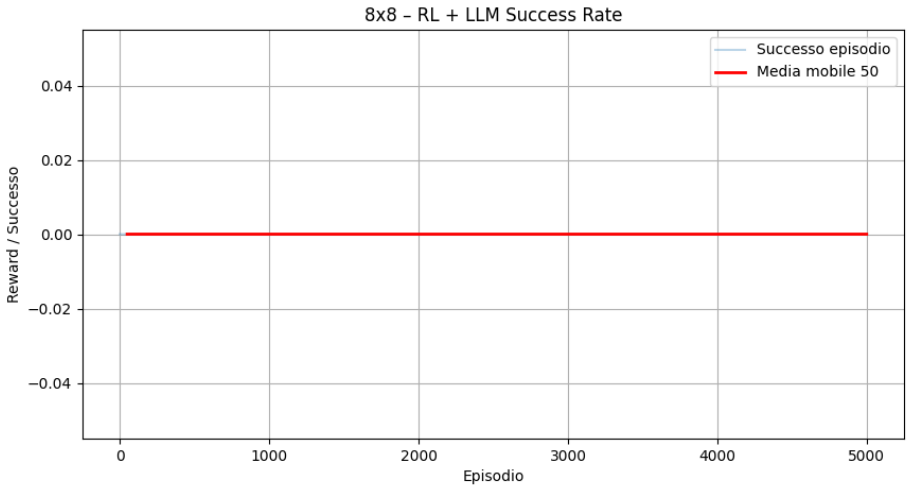

Figura 4.11: Success rate RL + LLM mappa 8x8

In questo caso il success rate rimane nullo per tutta la durata dell'addestramento. Nonostante il supporto del modello linguistico, l'agente non riesce mai a raggiungere il goal. Questo risultato suggerisce che, in ambienti di dimensioni maggiori, il contributo dell'LLM non è sufficiente a compensare la difficoltà esplorativa e l'assenza di feedback positivi frequenti.

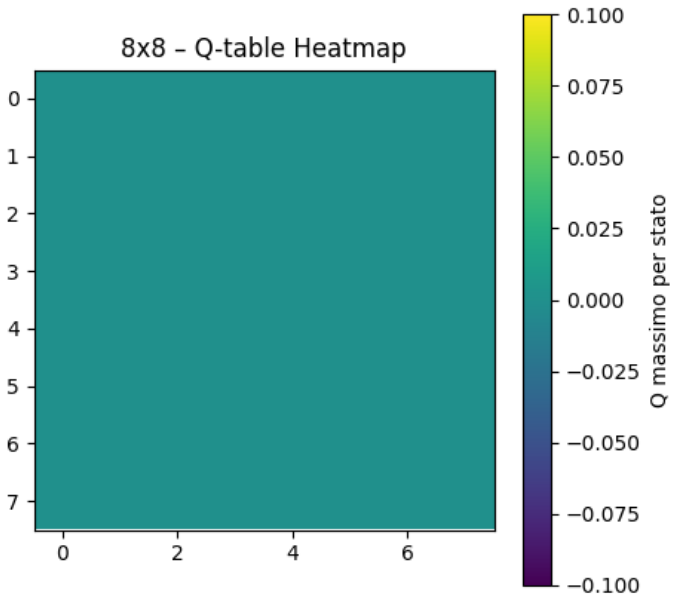

Figura 4.12: Q-table RL + LLM mappa 8x8

La heatmap della Q-table (Figura 4.12) risulta uniforme, indicando che i valori di stato non sono stati aggiornati in modo significativo. L'assenza di successi impedisce la propagazione del segnale di ricompensa, rendendo inefficace anche il contributo del modello linguistico come guida iniziale all'apprendimento.

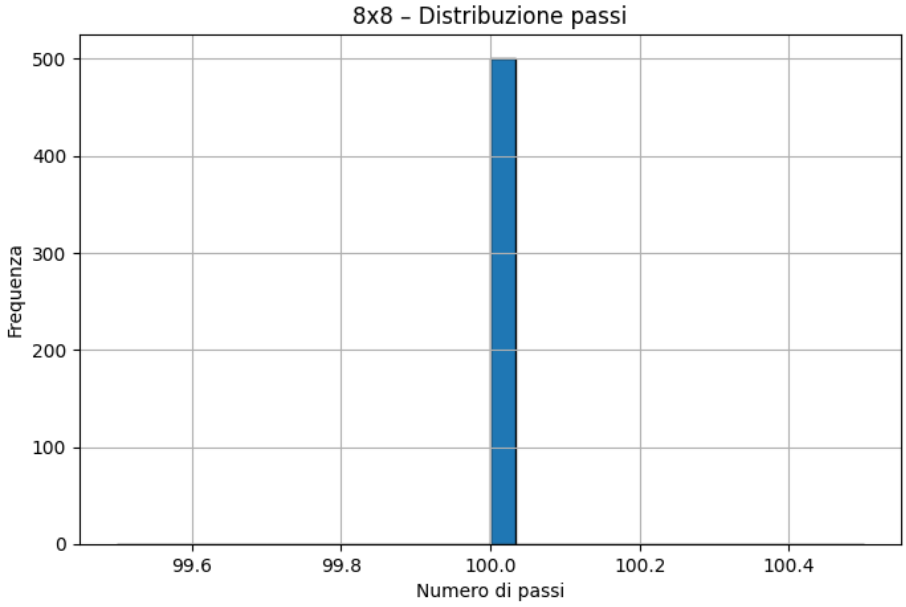

Figura 4.13: Distribuzione numero passi RL + LLM mappa 8x8

L'istogramma del numero dei passi (Figura 4.13) mostra un picco esclusivo in corrispondenza del numero massimo di passi consentiti. Ciò indica che l'agente continua a muoversi nell'ambiente senza raggiungere stati terminali informativi, rimanendo intrappolato in una fase di esplorazione inefficiente.

Figura 4.14: Success rate finale RL + LLM mappa 8x8

Il success rate complessivo è pari allo 0%. Questo risultato mette in evidenza come, nonostante l'integrazione di un modello linguistico, il problema della ricompensa sparsa rimanga critico negli ambienti complessi. Senza un meccanismo di reward shaping più incisivo o una strategia di pianificazione a lungo termine, l'agente non riesce a scalare efficacemente verso mappe di dimensioni maggiori.

CAPITOLO 5

CONCLUSIONI

In conclusione, in questo lavoro sono stati analizzati e confrontati tre approcci differenti alla risoluzione dell'ambiente Frozen Lake: un agente basato esclusivamente su Reinforcement Learning, un agente guidato da un Large Language Model come selettore delle azioni, e un agente ibrido che combina il Reinforcement Learning con l'LLM.

L'agente basato solo su RL rappresenta il riferimento metodologico. I risultati mostrano che, in ambienti di dimensioni ridotte come la mappa 4×4, l'agente è in grado di apprendere una policy efficace attraverso l'interazione ripetuta con l'ambiente e l'aggiornamento della Q-table. Tuttavia, all'aumentare della dimensione della mappa, le prestazioni degradano rapidamente. Nel caso della mappa 8×8, il problema della ricompensa sparsa e l'elevata complessità esplorativa rendono l'apprendimento estremamente lento o addirittura inefficace, evidenziando i limiti del Reinforcement Learning tabellare in ambienti complessi.

L'approccio basato sull'utilizzo diretto di un Large Language Model come generatore delle azioni mostra risultati significativamente inferiori. In assenza di un meccanismo di apprendimento iterativo, l'agente non è in grado di migliorare le proprie prestazioni nel tempo. I rari successi osservati, soprattutto sulla mappa 4×4, risultano rari e attribuibili al caso. Sulla mappa 8×8,

l'agente fallisce quasi sistematicamente, dimostrando che la sola capacità di ragionamento semantico del modello linguistico non è sufficiente per affrontare problemi sequenziali caratterizzati da incertezza e necessità di pianificazione a lungo termine.

L'agente ibrido, che integra il Reinforcement Learning con il supporto di un modello linguistico, rappresenta il contributo principale. I risultati indicano che questa combinazione può migliorare le prestazioni rispetto agli approcci presi singolarmente, ma solo entro certi limiti. Sulla mappa 4×4, l'agente ibrido riesce a sfruttare l'informazione semantica fornita dall'LLM per guidare l'esplorazione, accelerando l'apprendimento e portando a una policy più strutturata rispetto al caso del solo RL. La presenza di una Q-table con valori informativi e di un success rate stabile conferma l'efficacia dell'approccio in contesti semplici.

Al contrario, sulla mappa 8×8 anche l'agente ibrido non riesce a superare le difficoltà legate alla dimensionalità dell'ambiente e alla scarsità di ricompense. Nonostante il supporto del modello linguistico, l'agente non riesce a raggiungere il goal, mostrando che l'integrazione proposta non è sufficiente a garantire scalabilità verso ambienti più complessi.

Nel complesso, il confronto tra i tre agenti evidenzia come il Reinforcement Learning rimanga essenziale per l'apprendimento da interazione, mentre i modelli linguistici possono fornire un contributo utile come guida o bias informativo, ma non possono sostituire i meccanismi di apprendimento basati su ricompensa. L'approccio ibrido emerge come una direzione promettente, ma richiede ulteriori sviluppi per affrontare in modo efficace problemi di maggiore complessità.

Possibili estensioni future includono l'introduzione di tecniche di reward shaping più sofisticate, l'uso di approcci model-based o gerarchici, e una più profonda integrazione tra il ragionamento simbolico dei modelli linguistici e i processi di apprendimento del Reinforcement Learning. Tali sviluppi potrebbero permettere di superare i limiti osservati e rendere l'approccio ibrido realmente scalabile.