Documentazione Progetto: Agente Intelligente per il Gioco del Risiko

Gruppo di lavoro

- [Claudio de Candia], [777582], [c.decandia15@studenti.uniba.it]

Link Repository: [Cliccami]

AA: 2025-2026

Introduzione

Il Dominio: Risiko

Il progetto “Agente intelligente per il gioco Risiko” verte sulla realizzazione di un agente intelligente in grado di giocare al gioco da tavolo strategico *Risiko!*. Il dominio presenta sfide significative per l'Intelligenza Artificiale: uno spazio degli stati estremamente vasto, natura stocastica (lancio dei dadi, pesca delle carte) e informazioni imperfette (intenzioni degli avversari). L'obiettivo è sviluppare un agente capace non solo di usare regole basate su tattiche locali (es. "attacca il territorio più debole"), ma di pianificazione strategica globale orientata al raggiungimento di obiettivi specifici (es. conquista di continenti o distruzione avversari).

Sommario del Sistema (KBS)

Il sistema è un'architettura ibrida che integra un motore di gioco procedurale in Python con un modulo di ragionamento logico basato su Prolog.

Il cuore del sistema segue il ciclo *Sense-Think-Act*:

1. **Sense:** Python estrae lo stato del grafo (mappa) e lo sincronizza nella Knowledge Base (KB).
2. **Think:** La KB Prolog deduce stati di alto livello (es. "giocatore in pericolo", "territori vulnerabili", "linea del fronte"). Un modulo di ricerca A* pianifica percorsi ottimi, supportato da un calcolatore probabilistico (Catene di Markov) per stimare l'esito delle battaglie.
3. **Act:** L'agente esegue le azioni (rinforzo, attacco, spostamento) alternando ad ogni fase del turno la percezione e basandosi su una *policy* parametrica ottimizzata tramite simulazioni massive.

Elenco argomenti di interesse

Il progetto dimostra competenze pratiche sui seguenti temi del programma:

1. **Rappresentazione della Conoscenza e Ragionamento Relazionale** (Cap. 5, 15, 16): Uso di Prolog per modellare geografia, tattiche, obbiettivi, comportamento e analisi globale.

2. **Ricerca** di **Soluzioni** (Cap. 3): Implementazione di A* per la pianificazione di catene di attacco con euristiche topologiche.
3. **Apprendimento Supervisionato e Incertezza** (Cap. 7, 9, 10): Uso di modelli probabilistici per le battaglie e ottimizzazione degli iperparametri tramite analisi Random Forest su dati sintetici generati da simulazioni.

Sezione 1: Rappresentazione della Conoscenza (Relazionale/Prolog)

Sommario

La Knowledge Base (KB) è stata sviluppata in **Prolog** per sfruttare la sua capacità dichiarativa nel gestire relazioni complesse (adiacenze, appartenenza a continenti) e regole condizionali. La KB non è statica: viene asserita dinamicamente da Python ad ogni fase dei turni (occupato/3, carte/2), permettendo al motore inferenziale di dedurre lo stato strategico corrente senza dover scrivere complessi *if-else* annidati nel codice procedurale, inoltre permette all'agente di percepire e comprendere l'effetto delle proprie mosse (ad esempio una battaglia vinta/persa) e decidere di conseguenza come procedere.

Strumenti utilizzati

- **SWI-Prolog:** Motore inferenziale logico.
- **PySWIP:** Bridge Python-Prolog per l'iniezione dei fatti dinamici e l'interrogazione delle regole.

Decisioni di Progetto

La KB è stata modularizzata in cinque file logici per separare le responsabilità:

1. **Geografia:** Definizione statica del grafo (confina/2) e dei continenti.

   Introducendo la regola intercontinentale/1 per marcare i territori "ponte", cruciali per la difesa e per entrare in continenti obbiettivo.
2. **Obiettivi:** Logica flessibile che traduce l'obiettivo testuale (es. "conquista Nord America") in predicati verificabili (quanto_manca/3 e territorio_obiettivo/3), permettendo all'agente di identificare quali territori specifici sono rilevanti per la vittoria e comprendere quanto e cosa manca per raggiungerla.
3. **Tattica:** Definizione di concetti derivati complessi:
   - fronte(T, G): Territorio del giocatore **G** adiacente ad almeno un nemico.
   - vulnerabile(T, G): Territorio al fronte con truppe insufficienti rispetto alla minaccia massima vicina (calcolata aggregando le armate nemiche confinanti).
   - forte_al_fronte(T, G): Territori sicuri da cui lanciare attacchi.

4. **Analisi Globale:** Classificazione dello stato del giocatore (dominante, in_svantaggio, rischio_eliminazione) basata su un punteggio euristico composito (territori + bonus continenti + carte).
5. **Comportamento (Policy):** Un sistema a regole che assegna pesi alle azioni (es. peso_azione(sopravvivenza, difesa, 100)) in base allo stato globale dedotto. Questo permette all'agente di cambiare "personalità" (da aggressivo a difensivo) automaticamente. È in questa KB che sono definiti gli iperparametri che definiscono il comportamento dell'agente in base alla situazione di gioco.

Valutazione

L'approccio logico ha ridotto drasticamente la complessità del codice Python. Ad esempio, determinare se un continente è "conteso" richiederebbe cicli complessi in Python, mentre in Prolog è risolto con un setof e una regola di conteggio. La KB si è dimostrata robusta: la separazione tra fatti dinamici (aggiornati ogni turno) e regole statiche ha prevenuto incoerenze nello stato del gioco.

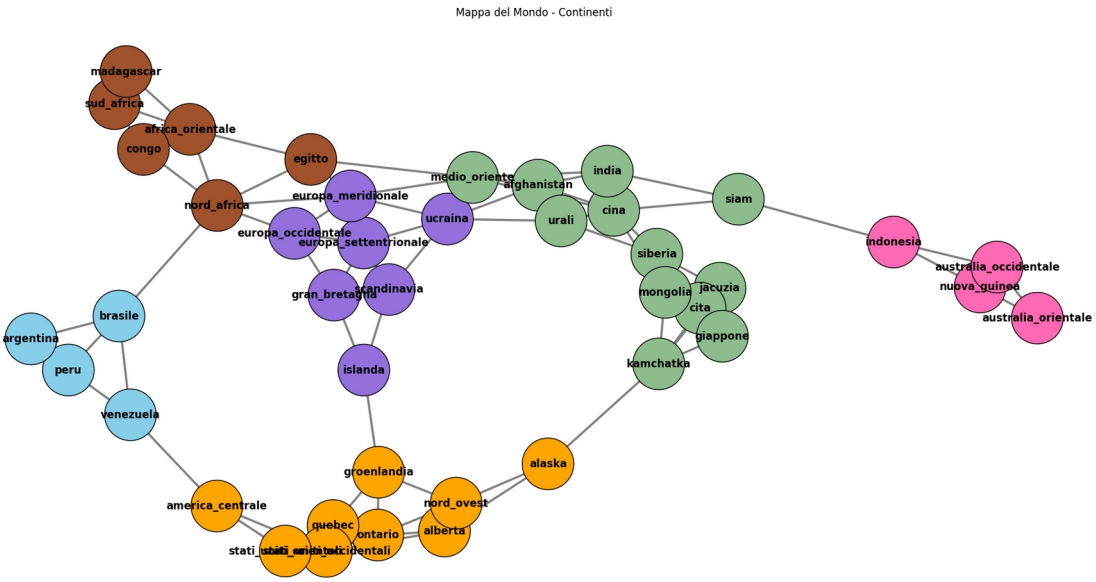
Sezione 2: Ricerca di Soluzioni (A* e Pathfinding)

Sommario

Nel Risiko, attaccare un territorio adiacente è banale, ma pianificare una "invasione" per raggiungere un obiettivo lontano richiede ricerca su grafo. È stato implementato l'algoritmo **A*** (A-Star) per trovare la catena di attacco ottima (sequenza di territori nemici da conquistare) per collegare un territorio di partenza a un territorio obiettivo strategico.

Strumenti utilizzati

- **Libreria heapq (Python):** Per la gestione efficiente della frontiera (coda con priorità) nell'algoritmo A*.
- **NetworkX:** Per la gestione della struttura dati del grafo sottostante.



Decisioni di Progetto

L'applicazione standard di A* (distanza spaziale) non è efficace nel Risiko. Sono state prese le seguenti decisioni implementative originali:

1. **Costo del Cammino (g(n)):** Non è la distanza in passi, ma la **resistenza nemica**. Il costo di attraversamento di un nodo è pari a armate_nemiche + 1. Questo spinge l'agente a cercare il "ventre molle" della difesa avversaria, aggirando le roccaforti.
2. **Euristica (h(n)):** È stata utilizzata la **distanza topologica (BFS)** verso l'obiettivo. Essendo ogni territorio occupato da almeno 1 armata, la distanza in passi è un'euristica ammissibile (non sovrastima mai il costo in armate).
3. **Vincolo di Invasione:** A differenza del pathfinding classico (navigazione), qui il percorso deve attraversare *solo* territori nemici. Non è possibile "passare attraverso" i propri territori in una catena di attacco continua nello stesso turno (regola del *blitz*).

   Viceversa per gli spostamenti di rinforzo che hanno una distanza massima di 1(e solo tra territori alleati) per cui non serve un algoritmo di ricerca su grafo quindi proceduralmente viene effettuato lo spostamento sul territorio fronte adiacente con meno armate.
4. **Integrazione Probabilistica:** Una volta trovato il percorso ottimo da **A***, questo viene validato da un calcolatore probabilistico (vedi Sez. 3) che determina se l'agente ha effettivamente abbastanza truppe per completare l'intera catena con una probabilità soglia (min_chain_win_prob).

Valutazione

L'uso di A* ha permesso all'agente di compiere manovre "profonde" (conquistare 3-4 territori in sequenza) impossibili per un agente *greedy* (che guarda solo ai vicini immediati). La combinazione di costo basato sulle armate ed euristica topologica ha garantito percorsi efficienti in termini di risorse spese.

Sezione 3: Apprendimento e Incertezza (Simulazione e Ottimizzazione)

Sommario

Il gioco include forte incertezza (dadi, carte bonus). È stato sviluppato un modulo per il calcolo esatto delle probabilità di vittoria (Markov) e un sistema di **Apprendimento dai Dati (Data-Driven Optimization)**. L'agente non impara "online" durante la partita, ma i suoi iperparametri sono stati ottimizzati attraverso un processo di simulazione massiva (Grid Search ibrido Monte Carlo) e analisi statistica (Random Forest classificatore e regressore).

Strumenti utilizzati

- **Programmazione Dinamica (Memoization):** Per il calcolo delle probabilità di battaglia e catene di attacco (tour_prob).

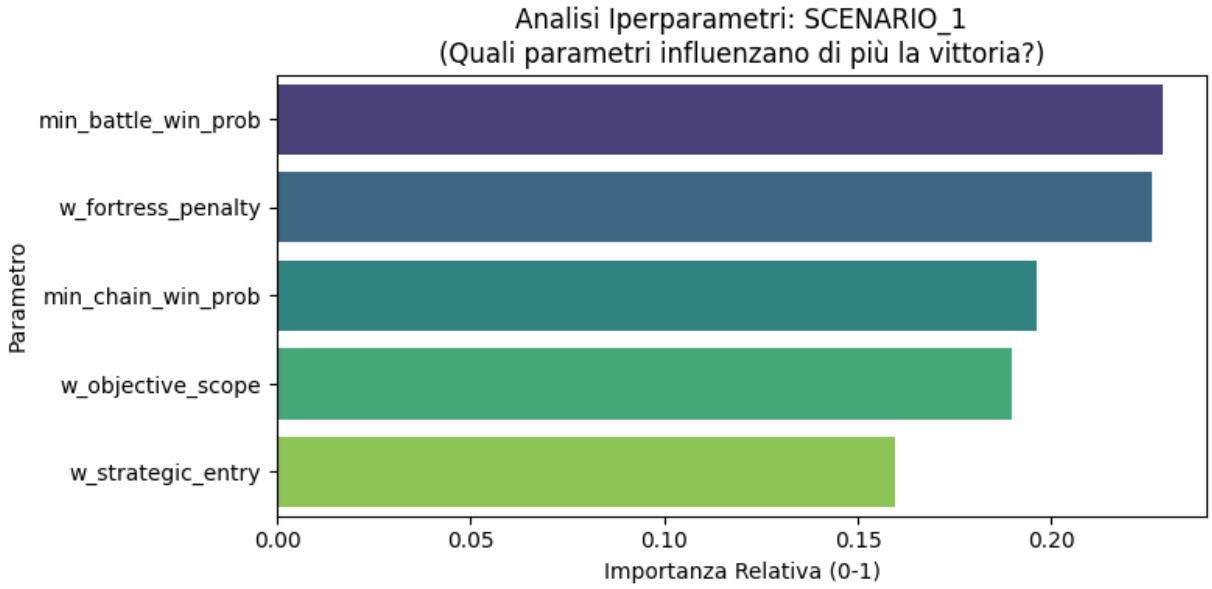
- **Scikit-Learn (Random Forest):** Per valutare l'importanza delle feature (iperparametri) rispetto all'esito della partita.
- **Matplotlib/Seaborn:** Per la visualizzazione dei risultati (distribuzione vittorie).

Decisioni di Progetto

1. **Calcolo Probabilistico:** Invece di semplici euristiche (es. "attacca se Armate > Nemico"), è stata pre-calcolata la matrice di transizione per gli scontri 3vs3 (regole italiane). Il metodo tour_prob calcola ricorsivamente la probabilità congiunta di conquistare un'intera sequenza di territori, considerando le perdite attese ad ogni passo (Markov).
2. **Generazione Dati Sintetici:** Vengono eseguite in serie due tipi di simulazioni tutte con un massimo di 100 turni a partita, un solo agente AI (blu) e due agenti casuali , uno totalmente random (verde) e l'altro (rosso) che attacca solo l'agente AI: la prima serie di simulazioni è basata su profili standard degli iperparametri suggeriti dall'esperto (il sottoscritto in questo caso): prudente (che predilige gli attacchi sicuri e la difesa dei propri territori) , bilanciato , aggressivo (che intraprende gli attacchi potenzialmente difficili e concentra le truppe al fronte), queste simulazioni vengono effettuate per 8 scenari (gli obbiettivi presenti nel gioco) al fine di generare il grosso dei dati per le analisi successive.
   Dopo aver completato le simulazioni su tutti gli scenari, è stato scelto uno da ottimizzare (poiché le simulazioni richiedono molto tempo: 2h circa per 30 partite), tuttavia il processo è modularizzato ed eseguibile per tutti gli scenari.

   La seconda fase di simulazione effettua un ibrido tra Grid-Search e Monte Carlo utilizzando una tecnica OfaT (One factor at Time) che fissa 4 su 5 iperparametri a valori standard e casualmente viene modificato il parametro escluso nel suo range di valori (tra 0 e 100 o tra 0 e 1 (probabilità) o tra 0 e -100 (penalità)). Ad ogni ciclo di simulazione si itera tra gli iperparametri da ricercare.

   Al fine di ottenere un Dataset sintetico sul quale basare le analisi statistiche dell'influenza degli iperparametri e addestrare un RF regressor a stimare i migliori parametri per un determinato scenario(obbiettivo dell'agente)
3. **Feature Importance:** I log delle partite (contenenti configurazione parametri e esito vittoria/sconfitta ed altri dettagli finali delle partite) sono stati usati per addestrare un classificatore **Random Forest**. L'analisi della *Feature Importance* ha rivelato quali parametri influenzano maggiormente la vittoria (es. in scenari di conquista continentale, il peso dato all'entrata strategica w_strategic_entry è risultato più decisivo della soglia di sicurezza min_battle_win_prob per il primo scenario).



4. Data-Driven Optimization: i log relativi alla simulazione ibrida di uno scenario vengono aggregati in un unico dataframe pandas per addestrare un regressore Random Forest che stima i valori per la migliore configurazione degli iperparametri dell'agente per un determinato scenario.
5. **Validazione:** Il profilo migliore estratto dall'analisi è stato validato in un set di simulazioni di partite di controllo, dimostrando un incremento del Win Rate rispetto ai profili standard.

Valutazione

Il sistema di probabilità ha reso l'agente estremamente cauto o audace in modo giustificato, evitando attacchi suicidi ("suicide runs") tipici dei bot casuali. L'approccio di ottimizzazione offline ha portato il Win Rate dell'agente (su scenari complessi come "Conquista Oceania e Nord America" che sarebbe il primo scenario) da una media base (~73% contro random) a valori superiori all'80% nel set di validazione, dimostrando l'efficacia del tuning guidato dai dati.

Conclusioni

Il progetto ha dimostrato come l'integrazione di tecniche diverse (KB logica per il ragionamento strategico, Ricerca euristica per la tattica, Probabilità per la gestione dell'incertezza) possa produrre un agente di gioco competente.

L'agente batte sistematicamente le baseline casuali e mostra comportamenti emergenti simili a quelli umani, come la protezione dei confini (dedotta dai predicati vulnerabile), la concentrazione delle forze per sfondamenti mirati (A*) e l'uso mirato dei bonus disponibili, sfruttando il momentum tipo dei giocatori reali di Risiko.

Sviluppi futuri potrebbero includere l'implementazione di *Reinforcement Learning* (Q-Learning) per permettere all'agente di adattare i pesi della policy in tempo reale durante la partita, invece che offline.

Report statistici:

1. Sintesi Configurazioni AI

In questa sezione vengono confrontati i parametri comportamentali impostati per ciascun profilo.

Parametro AI	P. Bilanciato	P. Aggressivo	P. Prudente	P. Validazione (Ottimizzato)
Min Battle Win Prob	0.4	0.25	0.65	0
Min Chain Win Prob	0.2	0.1	0.4	0
Peso: Strategic Entry	50	70	30	2
Peso: Objective Scope	25	15	40	8
Peso: Fortress Penalty	-30	-10	-60	-94

2. Performance di Vittoria

Confronto dei tassi di vittoria su 30 partite simulate per scenario (America & nord Oceania). *Nota: Il BLU è il colore di riferimento (agente AI).*

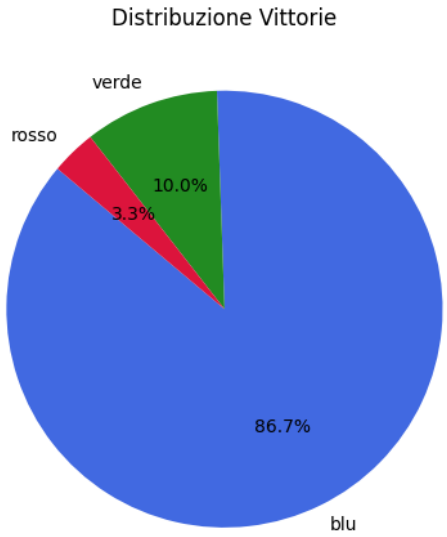
Profilo	Vittorie BLU	Vittorie ROSSO	Vittorie VERDE	PAREGGI (Draw)	Note Prestazionali
Bilanciato	**80.0%** (24)	20.0% (6)	0.0% (0)	0.0%	Buone prestazioni
Aggressivo	73.3% (22)	3.3% (1)	**23.3%** (7)	0.0%	Il Verde guadagna terreno; il Rosso crolla.
Prudente	83.3% (25)	3.3% (1)	13.3% (4)	0.0%	Simile al bilanciato ma riapre chance al Verde.
Validazione	86.7%	3.3% (1)	10.0% (3)	**0.0%**	Prestazioni simili al profilo

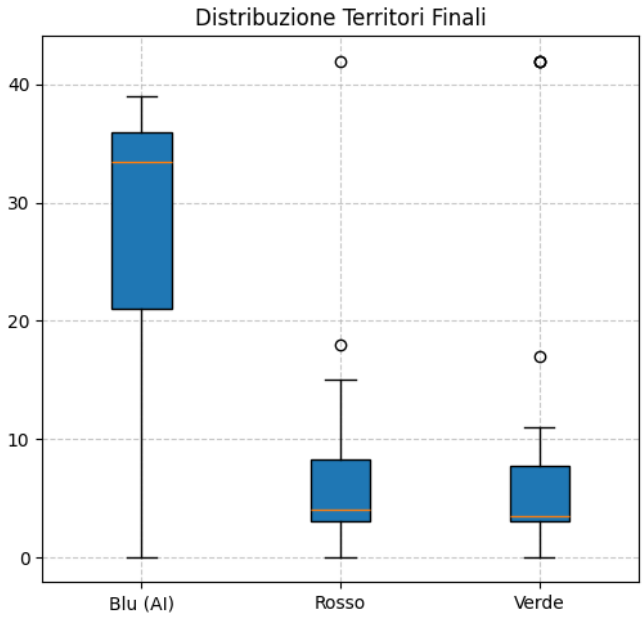
Profilo	Vittorie BLU	Vittorie ROSSO	Vittorie VERDE	PAREGGI (Draw)	Note Prestazionali
	(26)				bilanciato

3. Dinamiche di Gioco e Durata

Analisi della lunghezza delle partite (turni) e dell'espansione territoriale finale. I dati si riferiscono alle sole partite vinte.

Profilo	Media Turni	Deviazione Std.	Min / Max Turni	Territori Finali BLU (Media + dev std)
Bilanciato	29.80	51.47	2 / 196	24.20 +/- 11.52
Aggressivo	**10.83**	24.96	0 / 140	20.93 +/- 14.39
Prudente	28.30	49.27	2 / 183	22.63 +/- 13.01
Validazione	17.77	30.39	3/178	27.23 +/- 12.52





Riferimenti Bibliografici

[1] Russell, S., & Norvig, P. *Artificial Intelligence: A Modern Approach*. (Capitoli relativi a Search, Knowledge Representation, Uncertainty).

[2] Documentazione SWI-Prolog: https://www.swi-prolog.org/

[3] NetworkX Documentation: