Implementazione Mastermind in CLIPS

Federico Giacardi

Strategia 1: Raffinamento di Pattern

Scopo di questa strategia è provare a codificare il modo in cui io giocherei a Mastermind. L'idea è quella di suddividere la partita in due fasi:

1. Individuazione dei quattro colori che compongono il codice segreto
2. Individuazione del loro corretto posizionamento

   ognuna delle quali svolta analizzando il feedback del sistema, nel tentativo di comporre progressivamente un pattern di colori, per poi affinarlo fino all'ottenimento del codice segreto.

Implementazione

Il flusso di attivazione delle regole è il seguente:

1. starting-guess
2. process-first-feedback
3. process-second-feedback
4. Fino all'individuazione dei 4 colori
    a. color-feedback
    b. compute-best-comb
5. Fino all'individuazione del codice segreto
    a. refine-pattern
    b. update-best-pattern

    Per la sottoposizione delle risposte, vengono usate le regole check-repetition, signal-repetition, e numbers-to-colors.

    Il passaggio tra le fasi di colors e positions guessing è gestito con opportuni fatti di controllo.

Struttura della conoscenza

La strategia è implementata nel modulo 4_Imp.

Tra i fatti più significativi per il suo funzionamento figurano:

- **phase**: traccia la macro-fase della strategia in corso (COLOR, per l'individuazione dei quattro colori e POSITION, per l'individuazione del posizionamento)
- **candidate-guess**: potenziale risposta prodotta dall'agente, potrebbe essere la ripetizione di un tentativo precedente
- **numeric-guess:** risposta dell'agente in cui i colori sono rappresentati da numeri compresi tra 1 ed 8, per semplificarne generazione e manipolazione. Sicuramente univoca.
- **best-comb** e **second-best**: rappresentazione dei due migliori tentativi effettuati dall'agente, comprensivi del corrispondente feedback di sistema

Semantica delle regole

- **starting-guess**: generazione del primo tentativo, composto dai colori blue green red yellow orange. Non potendo far ricorso a ripetizioni, si tratta della combinazione iniziale più informativa possibile, dato che in una sola mossa stabiliamo se quattro colori compaiano nel codice segreto.
- **process-first-feedback**: qualora i colori non siano stati individuati completamente nel primo tentativo, ne sostituisce (right-placed - miss-placed) scelti casualmente con altri, selezionati sempre in modo randomico, tra quelli non ancora presenti nella risposta. In caso di individuazione di tutti i colori al primo tentativo, avvia la fase di affinamento della posizione, asserendo l'apposito fatto di controllo.
- **process-second-feedback**: confronta il numero di right-placed e miss-placed dei primi due tentativi per stabilire quale costituisca la miglior combinazione e quale la seconda più promettente. Il valore di una combinazione, in questa fase, è determinato come numero di colori corretti al suo interno, cioè right-placed + miss-placed.
- **color-feedback**: L'idea è la stessa di process-first-feedback, ad essere sostituiti però, sono i colori della miglior combinazione che non compaiono anche nella seconda più promettente. In questo modo, consideriamo come più promettenti i colori rimasti a far parte delle due migliori soluzioni per più tempo. Notare che la natura casuale della sostituzione, dato che non sappiamo esattamente quali colori siano corretti, rende questo meccanismo un'approssimazione del comportamento ideale, che potrebbe finire con il favorire, mantenendoli nelle risposte date, colori che sono in realtà non corretti. Il nuovo colore da introdurre è nuovamente estratto casualmente.
- **compute-best-comb**: eventuale aggiornamento delle due migliori combinazioni provate fino a questo momento.
- **refine-pattern**: Si arriva a questa fase dopo l'individuazione di tutti i 4 colori che compongono il codice segreto. L'affinamento della posizione avviene effettuando una permutazione di due colori scelti casualmente dalla miglior risposta.
- **check-repetition**: verifica che la combinazione generata dall'agente non sia già stata sottoposta al sistema negli step precedenti. Necessaria a causa della natura pseudo-casuale delle funzioni di sostituzione e permutazione di colori. Genera la numeric-guess corrispondente alla candidate guess.
- **signal-repetition**: qualora si tratti di una combinazione ripetuta, ritratta la candidate guess generata dall'agente, causando la riattivazione di color-feedback oppure refine-pattern (a seconda della fase) per generare una nuova risposta
- **numbers-to-colors:** genera la guess con colori simbolici corrispondenti ai valori presenti nella numeric-guess asserita dall'agente

Strategia di Swaszek

Quanto descritto è basato sull'articolo The Mastermind Novice, di Peter F. Swaszek [1].

Anche questa strategia può essere divisa in due macro fasi

1. La generazione di tutti i possibili codici segreti (combinazioni di quattro numeri, ciascuno dei quali compreso tra 1 ed 8, in questo caso semplificata dal non ammettere ripetizioni).
2. La potatura dell'insieme di cui al passo 1, effettuata scartando tutti i codici che, qualora fossero il codice segreto usato dal sistema, avrebbero generato una risposta diversa (in termini di right-placed a miss-placed) rispetto a quella prodotta a seguito dell'ultimo tentativo dell'agente

Ad ogni passo, l'agente sottopone come tentativo il primo codice consistente individuato durante la potatura. Come affermato in [1] (anche se i risultati in questione sono relativi al Mastermind con ripetizioni), questo criterio di selezione porta a concludere il gioco mediamente in 4.638 mosse.

Come emergerà durante l'analisi dei risultati, sussiste comunque la possibilità che l'agente perda e, senza ripetizioni, la strategia sembra meno performante, anche se il campione di partite prese in esame è limitato.

Implementazione

1. generate-secret-codes
2. starting-guess
3. Fino all’individuazione del codice segreto, oppure all’esaurimento dei tentativi
    a. remove-inconsistent
    b. reset-general-counter

Anche qui, la regola numbers-to-colors è usata per convertire la rappresentazione numerica della risposta nel formato simbolico usato dal sistema, ed il passaggio tra le fasi è regolato da opportuni fatti di controllo.

Struttura della conoscenza

La strategia è implementata nel modulo 5_Swa.

Tra i fatti più rilevanti per il suo funzionamento troviamo:

- **numeric-guess**: stessa semantica della strategia precedente
- **candidate-secret-code**: rappresentazione dei potenziali codici segreti
- **candidate-codes-number**: contatore del numero di potenziali codici segreti ancora plausibili, aggiornato durante le fasi di potatura.
- **general-counter**: contatore del numero di codici ancora da controllare durante la fase di potatura.
- **candidate-answer**: rappresentazione della potenziale risposta dell'agente, valorizzata al primo codice candidato compatibile con l'ultima risposta ottenuta
- **candidate-sol-counter**: segnala l'individuazione di una potenziale risposta, evitando che ne vengano considerate altre
- **phase**: traccia la fase corrente della strategia

Semantica regole

- **generate-secret-codes:** effettua la generazione dei possibili codici segreti, ovvero tutte le sequenze di 4 numeri, ciascuno compreso tra 1 ed 8, senza ripetizioni
- **starting-guess**: come nella strategia precedente
- **remove-inconsistent**: implementa la potatura dell'albero dei codici candidati. L'idea è quella di scorrere i codici rimasti, confrontandoli con l'ultimo tentativo dell'agente, in modo da calcolare i right-placed ed i miss-placed che il sistema averebbe fornito qualora il codice candidato in questione fosse davvero quello segreto. Se questi valori non risultano compatibili con la risposta effettivamente fornita dal sistema, allora il candidato viene scartato. Il primo codice plausibile viene salvato nella working memory, in modo da poter essere utilizzato come futuro tentativo. La potatura termina quando candidate-sol-counter raggiunge 0.
- **reset-general-counter:** sottopone il codice compatibile al sistema, imposta general-counter a candidate-codes-number, in modo da sapere quanti codici andranno processati nella successiva fase di potatura.

Analisi dei risultati

Purtroppo, il mancato funzionamento della funzione random di CLIPS (anche il passaggio a Clippy non risolve del tutto il problema, probabilmente a causa dell'elevato numero di valori casuali da generare in rapida successione), ha impedito di testare adeguatamente la prima strategia. Il fatto che la funzione random generi sempre la stessa sequenza di numeri porta spesso l'agente a ripetere più volte le stesse combinazioni. La presenza delle regole check e signal repetition evita che queste vengano sottoposte più volte al sistema, ma porta al loop dell'agente. Eliminandole, un consistente numero di tentativo viene sprecato a causa delle ripetizioni, senza che l'agente riesca ad individuare effettivamente almeno i colori presenti nel codice segreto.

Nelle partite in cui questo non si verifica, l'agente riesce effettivamente ad individuare i colori, impiegando tuttavia un numero molto elevato di mosse (circa 8), lasciando poco spazio al raffinamento del posizionamento. In nessun caso quindi, l'agente è effettivamente riuscito ad individuare il codice segreto.

Questo è spiegabile con il fatto che i colori da sostituire e le posizioni da invertire vengano selezionate casualmente, perchè dal feedback del sistema non è possibile sapere esattamente quali siano i colori errati oppure fuori posto. L'agente, almeno in questa implementazione, paga quindi dazio rispetto all'intelligenza umana, che riesce ad implementare logiche di sostituzione ed affinamento del pattern più complesse.

Tuttavia, non è difficile immaginare implementazione di queste logiche in un sistema esperto, basandosi magari su una strategia di ricerca nello spazio contenente lo storico delle mosse, in modo da comporre il pattern tenendo conto di un numero di tentativi superiore agli ultimi 2.

In relazione a quest'ultimo punto, l'idea di mantenere i colori presenti anche nei due migliori tentativi del sistema non si è rivelata promettente come atteso, finendo spesso con "l'insistere" su colori che non sono corretti, contribuendo ad allungare la fase di individuazione dei colori. Un possibile miglioramento prevederebbe di ispezionare tutta la sequenza di tentativi, mantenendo i colori presenti in sequenze di tentativi che hanno effettivamente portato ad un miglioramento dello score.

Analisi risultati

Stante quanto presentato in [1] ed il modo sistemi di procedere, ci si attendeva un miglioramento delle prestazioni nel caso medio da parte della strategia di Swawzek. Così è stato, con l'agente capaci di riportare, su una sequenza di 20 partite, effettuate avendo cura che il codice segreto fosse sempre diverso, 8 vittorie, in 5.375 mosse di media, e 12 sconfitte.

La spiegazione dei risultati ruota attorno alla natura combinatoria della strategia stessa. La generazione di tutti i possibili codici segreti infatti, garantisce l'individuazione della soluzione, ma rende anche la strategia più lenta dal punto di vista computazionale e più difficile da utilizzare per un giocatore umano.

Soprattutto, stante il ridotto numero di mosse a disposizione, non è detto che la potatura sia sufficientemente rapida da garantire l'individuazione del codice entro la fine del gioco, situazione ulteriormente peggiorata dallo scegliere come risposta dell'agente al passo i-esimo il primo codice segreto candidato compatibile.

A riprova di ciò, vi è il fatto che, in 20 partite giocate con numero di mosse massimo aumentato a 20, l'agente non ha mai riportato sconfitte.

Bibliografia

1. P. F. Swaszek. The mastermind novice. Journal of Recreational Mathematics, 30(3):193–198, 2000.