Report di progetto - SudokuWhiz

DOROTEA SERRELLI and RAFFAELLA SABATINO

In questo report si descrive come si è pensato di costruire un agente intelligente capace di risolvere in modo corretto ed efficiente il gioco del Sudoku, utilizzando i seguenti algoritmi di ricerca: Backtracking, Ricerca A*, Simulated Annealing, Algoritmi Genetici Generazionali.

Tali algoritmi di ricerca sono stati confrontati tra loro in termini di prestazioni.

Si riporta, inoltre il link al repository Github per le demo, scritte in linguaggio Java, per gli algoritmi di ricerca precedentemente citati: https://github.com/SabatinoRaffaella/SudokuWhiz.

Additional Key Words and Phrases: Risoluzione di una griglia Sudoku mediante un agente intelligente, Backtracking, Ricerca A*, Simulated annealing, Algoritmi genetici

ACM Reference Format:

Dorotea Serrelli and Raffaella Sabatino. 2023. Report di progetto - SudokuWhiz. 1, 1 (December 2023), 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Authors' address: Dorotea Serrelli; Raffaella Sabatino.

ACM XXXX-XXXX/2023/12-ART

https://doi.org/10.1145/nnnnnnn.nnnnnnn

Contents

Comprensione del problema e dell'ambiente

Comprensione del problema e dell'ambiente

1 INTRODUZIONE

1.1 Il gioco Sudoku

Il Sudoku è un gioco di logica nel quale viene proposta una griglia di 9×9 celle, ciascuna delle quali può contenere un numero da 1 a 9, oppure essere vuota; la griglia è suddivisa in 9 righe orizzontali, 9 colonne verticali e in 9 "sottogriglie" di 3×3 celle contigue. Queste sottogriglie sono delimitate da bordi in neretto e chiamate regioni. Le griglie proposte al giocatore, in generale, hanno da 20 a 35 celle contenenti un numero.

Il gioco fu inventato dal matematico svizzero Eulero da Basilea (1707-1783). La versione moderna del gioco fu pubblicata per la prima volta nel 1979 dall'architetto statunitense Howard Garns all'interno del *Dell Magazines* con il titolo *Number Place*. In seguito fu diffuso in Giappone dalla casa editrice *Nikoli* nel 1984, per poi diventare noto a livello internazionale soltanto a partire dal 2005, quando fu proposto in molti periodici.

Lo scopo del gioco è quello di riempire le caselle bianche con numeri da 1 a 9 in modo tale che in ogni riga, in ogni colonna e in ogni regione siano presenti tutte le cifre da 1 a 9 senza ripetizioni.

1.2 Obiettivi del progetto

In questo progetto si intende costruire un agente intelligente in grado di risolvere il gioco nel minor tempo possibile, rispettando le regole del gioco.

La griglia completa del Sudoku verrà determinata applicando i seguenti algoritmi di ricerca: back-tracking, ricerca A*, simulated annealing, algoritmi genetici generazionali.

2 DESCRIZIONE DELL'AMBIENTE

2.1 Osservazioni preliminari per la formulazione della specifica PEAS dell'ambiente

Il gioco si svolge in matrici di aspetto 9×9 (le griglie), denotate *matrici Sudoku*, le cui caselle possono contenere un intero da 1 a 9 oppure il valore 0, per denotare la casella bianca o vuota. Una matrice Sudoku M è suddivisa in 9 blocchi di aspetto 3×3, denotati Bh,k con h,k = 1, 2, 3; il blocco Bh,k riguarda, per la matrice M, le righe relative agli indici 3h-2, 3h-1 e 3h e le colonne relative agli indici 3k-2, 3k-1 e 3k. In ogni riga, colonna e regione di una matrice Sudoku i valori interi non possono essere ripetuti.

La griglia Sudoku proposta o *matrice Sudoku incompleta* è una matrice Sudoku che presenta alcune celle bianche.

Lo scopo del gioco è trasformare la griglia proposta in una matrice completa, cioè in una matrice priva di celle bianche e, quindi, tale che in ogni sua riga, colonna e regione compaiano tutti i numeri dell'insieme 1,2,3,4,5,6,7,8,9, ciascuno una sola volta.

Affinché una matrice incompleta sia considerata valida ai fini del gioco, è necessario che la soluzione sia univoca, ovvero non devono sussistere due o più soluzioni differenti. Risulta essere necessario, dunque, definire il numero minimo di indizi presenti di partenza nella matrice Sudoku.

Il numero massimo di indizi di partenza non conta; tuttavia, se ce ne sono troppi il gioco diventa banale e si riduce ad una procedura meccanica di riempimento senza che occorra alcun procedimento logico.

La quantità minima di numeri di partenza è, invece, importante perché, sotto una certa soglia, il rompicapo diventa impossibile da risolvere; infatti, si rischia che la matrice Sudoku ammetta più di una soluzione e ciò lo renderebbe non valido (ed esteticamente poco interessante).

Per definire questo valore, ci si basa su uno studio della rivista *Nature* del 2013, condotto da un gruppo di matematici diretto da Gary McGuire dell'Università di Dublino. Tale gruppo di ricerca ha dimostrato che il numero minimo di indizi necessari per risolvere una griglia Sudoku 9×9 è 17; infatti, con meno indizi è impossibile riempire univocamente la griglia del gioco su cui giornalmente milioni di persone si cimentano.

Di seguito si riporta il link al nostro repository di Github per visionare l'abstract Solving the Sudoku Minimum: https://github.com/SabatinoRaffaella/SudokuWhiz/blob/main/Documents/Solving%20the%20Sudoku%20Minimum_abstract.pdf.

Pertanto, per garantire che il Sudoku abbia una sola soluzione, si pone come valore minimo di celle non vuote all'inizio del gioco il numero 17.

2.2 Specifica PEAS dell'ambiente

Di seguito si delinea la specifica PEAS dell'ambiente in cui opererà l'agente:

- **Performance**

  La misura delle prestazioni è la capacità dell'agente di risolvere una griglia Sudoku in modo corretto e efficiente nel tempo, trovando una sola soluzione.

- **Environment**

  L'ambiente è rappresentato dalla griglia Sudoku stessa, che è costituita da una griglia 9×9 parzialmente riempita, divisa in 9 regioni 3×3. L'ambiente include anche le celle vuote — denotate con il valore 0 – da riempire con numeri da 1 a 9, rispettando le regole del gioco. Nel paragrafo successivo si descriveranno le caratteristiche dell'ambiente.

- **Actuators**

  Gli attuatori disponibili dell'agente per compiere delle azioni: inserimento, cancellazione, spostamento.

- **Sensors**

  I sensori dell'agente sono utilizzati per percepire lo stato corrente della griglia Sudoku, ossia per rilevare i numeri già presenti nella griglia e quelli che l'agente può inserire in modo legale nelle celle vuote.

2.2.1 Caratteristiche dell'ambiente. L'agente si interfaccia in un ambiente con le seguenti caratteristiche:

- **completamente osservabile**: l'agente è a conoscenza in ogni istante della configurazione della griglia del Sudoku;
- **singolo agente**: l'unico agente che opera in questo ambiente è quello in oggetto;
- **deterministico**: la configurazione della griglia del Sudoku Mj è il risultato solo e soltanto dell'azione dell'agente eseguita sulla configurazione Mi della griglia del Sudoku corrente;

- **sequenziale**: la scelta dell'azione dell'agente sulla configurazione della griglia del Sudoku corrente dipende dalle azioni fatte nelle configurazioni precedenti (inserimento/cancellazione di un numero, …);
- **statico**: durante l'esecuzione dell'agente, la griglia del Sudoku non muta mentre l'agente pensa alla prossima azione;
- **discreto**: in ogni configurazione della griglia del Sudoku c'è un insieme finito di percezioni ed azioni; infatti, la griglia è formata da un numero di celle finito, ciascuna con un numero discreto (da 0 a 9).

3 FORMULAZIONE DEL PROBLEMA

Di seguito si riporta l'analisi del problema da risolvere:

- **Stato iniziale**

  Lo stato iniziale corrisponde alla configurazione iniziale della griglia 9×9, parzialmente compilata con almeno 17 indizi e massimo 30, è definita in un file Sudoku.txt.
  Ogni riga del file è una riga della griglia 9×9 ed i numeri presenti in una riga sono separati l'uno dall'altro mediante uno spazio. Le caselle vuote sono indicate con il valore 0.

- **Azioni**

  Le azioni possibili per l'agente sono le seguenti:
  - Inserimento di numeri da 1 a 9 nelle celle vuote, rispettando le regole del gioco.
  - Lettura di un numero in una cella;
  - Cancellazione di un numero inserito in una cella (ad eccezione delle celle inizialmente compilate, definite nel file);
  - Spostamento verso una nuova cella.

- **Modello di transizione**

  Il modello di transizione descrive come lo stato della matrice Sudoku cambia in risposta alle azioni dell'agente.
  Se l'agente inserisce un numero x in una cella vuota y (con 1 <= x <= 9), la matrice Sudoku corrente transita ad un nuovo stato, ossia la matrice Sudoku risultante avrà la cella y contenente il valore x. Il valore x scelto non sarà un numero già presente nella riga, nella colonna e nella regione in cui si trova la cella y.
  Se l'agente rimuove un numero x in una cella vuota y, la matrice Sudoku corrente transita ad un nuovo stato: la matrice Sudoku risultante avrà la cella y priva del valore x, ed è denotata con il valore 0.
  Se l'agente decide di leggere il valore di una cella, la configurazione della matrice Sudoku non transita in nuovo stato.
  Se l'agente decide di spostarsi in una nuova cella, la configurazione della matrice Sudoku non transita in nuovo stato, bensì cambia la posizione dell'agente.

- **Test-obiettivo**

  Il test-obiettivo consiste nel verificare se il Sudoku è stato risolto correttamente nel minor tempo possibile, ovvero se tutte le regole del gioco sono state rispettate.

- **Costo di cammino**

Il costo di cammino è definito come il costo dell'esecuzione di una o più azioni necessarie per risolvere il Sudoku. Si suppone che la lettura di un valore della cella e lo spostamento da una cella all'altra hanno un costo pari a 1, mentre le altre operazioni hanno un costo pari a 2.

Algoritmi di ricerca

Algoritmi di ricerca

4 ALGORITMI DI RICERCA ANALIZZATI

Il problema della risoluzione di una griglia Sudoku parzialmente riempita è stato affrontato guardandolo da diverse prospettive.

Innanzitutto, si è guardato al primo possibile approccio per risolvere il gioco, applicato da un giocatore naïve: il backtracking. Successivamente, si è pensato ad una possibile strategia che può utilizzare un giocatore che pratica regolarmente il gioco, però senza aver raggiunto il livello di esperto: la ricerca informata A*. ...

5 BACKTRACKING

5.1 Meccanismo di funzionamento

Il backtracking è un algoritmo utile per risolvere problemi con la ricorsione, costruendo una soluzione incrementalmente.

In generale, il backtracking intende espandere una soluzione parziale s = $(a_1, a_2, a_3, \ldots, a_k)$, dove l'elemento ai viene scelto da un insieme ordinato Si di possibili candidati per la posizione i.

L'espansione della soluzione s consiste nel costruire, a partire da s, l'insieme S_{k+1} dei possibili candidati per la posizione k+1 e scegliere un elemento $a_{k+1} \in S_{k+1}$. Fin quando l'estensione genera una soluzione parziale, si continua ad estenderla.

Se l'insieme dei possibili candidati S_{k+1} per la posizione k+1 della soluzione parziale s è vuoto, significa che non c'è la possibilità di estendere la soluzione corrente. Dunque, è necessario ritornare indietro e sostituire l'ultimo elemento a_k nella soluzione s con un altro candidato dell'insieme S_k, con $a_{k'} \neq a_k, a_{k'} \in S_k$.

5.2 Il backtracking in SudokuWhiz

Per il nostro gioco, la soluzione parziale che si intende espandere è la griglia 9×9 del Sudoku, che si suppone riempita fino alla cella posta nella riga i e colonna j ($0 \leq i < 8, 0 \leq j < 8$), denotata con c[i][j].

L'agente si trova nella cella c[i][j+1] e sta valutando l'insieme dei possibili candidati per questa cella, ossia i numeri da 1 a 9.

Si intende, quindi, inserire in modo iterativo i numeri da 1 a 9 uno per uno: se il numero inserito soddisfa tutte e 3 le condizioni, cioè quel particolare numero non è presente nella riga, colonna e regione della cella c[i][j], si conferma tale inserimento.

Poi, si passa alla cella vuota successiva e si compie la stessa iterazione dei numeri ma con la matrice Sudoku aggiornata. Man mano che si procede con la nuova cella vuota, si potrebbe raggiungere una cella, denotata con c[h][k] ($i \leq h \leq 8, j + 1 < k \leq 8$) in cui non si può inserire nessun numero dell'insieme 1, 2, 3, 4, 5, 6, 7, 8, 9.

Quindi, la soluzione costruita fino adesso viene respinta e si torna indietro, provando nuovi valori possibili in quelle celle precedenti. Poiché si assume, per configurazione iniziale della matrice Sudoku, che una matrice Sudoku ha solo una soluzione, si continuerà a respingere la soluzione fino a quando non si trova quella che soddisfa tutte le condizioni per ogni cella inizialmente vuota.

Per questo problema, l'algoritmo di backtracking cercherà di posizionare ogni numero in ogni riga e colonna vuota fino a quando non è risolto.

5.3 Complessità dell'algoritmo backtracking

Nell'algoritmo di backtracking viene costruito un albero delle soluzioni per il problema, dove ciascun nodo interno x è una soluzione parziale e l'arco tra il nodo x e il nodo y nell'albero viene creato se il vertice y è frutto dell'estensione della soluzione parziale x. Le foglie dell'albero sono le soluzioni.

Mediante la verifica del soddisfacimento delle condizioni di unicità per riga, colonna e regione, si effettua una potatura delle soluzioni che non soddisfano le condizioni richieste.

Il backtracking corrisponde ad effettuare una visita in profondità dell'albero delle soluzioni poiché l'intento è quello di trasformare la griglia Sudoku all'inizio incompleta (rappresentata dal nodo radice dell'albero) in una griglia completa (rappresentata da una foglia dell'albero).

L'algoritmo risulta essere completo poiché in modo esaustivo esplora tutte le possibili combinazioni e, per via delle assunzioni fatte sulla configurazione iniziale della matrice Sudoku, troverà una soluzione.

L'algoritmo, inoltre, risulta essere ottimo perché riesce a determinare la soluzione ottima, ma non in termini di efficienza; infatti, la complessità temporale richiesta è esponenziale $O(9^{n \times n})$ poiché, per ogni cella, si hanno a disposizione 9 valori su cui scegliere – anche se si debbano fare delle scelte nel caso peggiore per $81 - 17 = 64$ caselle affinché il Sudoku abbia una sola soluzione – .

La complessità spaziale richiesta è pari a $O(n^2)$ per tenere traccia di una matrice Sudoku di dimensione $n \times n$.

Si riporta in linguaggio Java uno screenshot relativo all'algoritmo di backtracking. …

Nella seguente tabella si riportano i tentativi fatti per diverse tipologie di Sudoku: nel caso peggiore (17 indizi iniziali), medio (24 indizi iniziali), facile (30 indizi iniziali). Le griglie Sudoku realizzate per fare il test si trovano nel package Griglie del progetto Java SudokuWhizWhiz.

Livello di difficoltà	MNE	MNS	Tempo medio per trovare una soluzione
Caso peggiore	311.307.624, 5	181	3, 6717028
Medio	3.499.731	170	0, 1290079
Facile	14.389, 5	106, 5	0, 0054264

Table 1. Prestazioni del backtraing su diversi livelli di difficoltà del gioco

È da sottolineare che, sottoponendo l'agente alla risoluzione delle diverse griglie Sudoku 9×9, si è notato che anche la diversa configurazione degli indizi iniziali nella griglia hanno un impatto forte sul tempo di ricerca della soluzione e sul numero di nodi da esplorare.

Come esempio di quanto appena affermato si riportano i dati della tabella relativi sulle griglie Sudoku_casoPeggiore e Sudoku_casoPeggiore2.

Griglia	MNE	MNS	Tempo medio per trovare una soluzione
Sudoku_casoPeggiore	37.652	103	0, 0057115
Sudoku_casoPeggiore2	622.577.597	259	7, 3376941

Table 2. Prestazioni del backtraing sulle due griglie Sudoku_casoPeggiore e Sudoku_casoPeggiore2 con livello di difficoltà *Caso peggiore*

6 RICERCA A*

6.1 Meccanismo di funzionamento

GLOSSARY

MNE acronimo di numero Medio dei Nodi Esplorati dall'algoritmo di ricerca durante la sua esecuzione . 8, 9

MNS acronimo di numero Medio dei Nodi Soluzione esplorati dall'algoritmo di ricerca durante la sua esecuzione per determinare la soluzione. 8, 9