Risoluzione di un Sudoku: Forward Checking VS Maintaining Arc Consistency

Anna Valanzano

June 7, 2020

Abstract

Si vuole confrontare il tempo di risoluzione di un Sudoku implementando le strategie di inferenza *Forward Checking* e *Maintaining Arc Consistency* e utilizzando come base l'algoritmo di *Backtracking Search*.

1 Introduzione

Il Sudoku è una griglia 9x9 in cui ogni cella può contenere un intero da 1 a 9 oppure essere vuota. La griglia può essere vista come costituita da 9 righe o da 9 colonne o da 9 sottogriglie 3x3. Il problema di risolvere un Sudoku consiste nel riempire le celle vuote con numeri da 1 a 9 in modo tale che in ogni riga, in ogni colonna e in ogni sottogriglia siano presenti tutte le cifre da 1 a 9, senza quindi ripetizioni.

2 Struttura del CSP

Si è implementato un risolutore di Sudoku basandosi sull'impostazione riportata in R&N 2010 e nell'articolo "Solving Sudoku Puzzle" dello stesso autore del libro, Peter Norvig. Pertanto, il Sudoku è stato considerato come un *Constraint Propagation Problem* con 81 variabili, una per ogni cella della griglia, con domini contenuti in {1, 2, 3, 4, 5, 6, 7, 8, 9} e con 27 vincoli *Alldiff*, ovvero 27 gruppi di variabili che devono avere tutti valori distinti. Considerando che le righe della griglia sono state etichettate con gli interi da 1 a 9 e le colonne con le lettere da A a I si sono ottenuti vincoli del tipo:

Alldiff (A1, A2, A3, A4, A5, A6, A7, A8, A9)
Alldiff (A2, B2, C2, D2, E2, F2, G2, H2, I2)
Alldiff (A1, A2, A3, B1, B2, B3, C1, C2, C3)

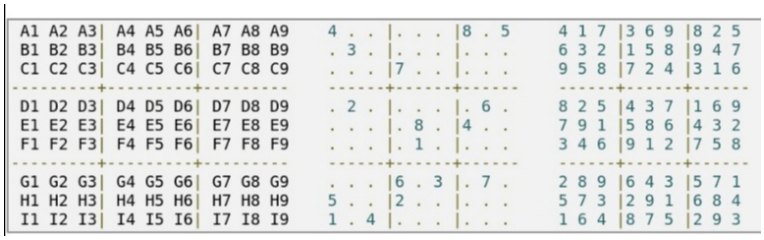

Figure 1: Esempio di Sudoku

3 Strutture dati

1. I domini delle variabili sono stati memorizzati attraverso due dizionari *domain* e *values* che ad ogni variabile associano i valori che in quel momento della ricerca si sa che quella variabile può assumere senza violare nessun vincolo. In particolare, i domini delle variabili corrispondenti a celle vuote nella configurazione iniziale del Sudoku sono inizializzati a {1,2,3,4,5,6,7,8,9}, mentre gli altri a {0} (per indicare che il loro valore può non può essere cambiato);

2. I vincoli sono memorizzati attraverso il dizionario *costraints* che associa a ogni variabile l'insieme delle 20 variabili con cui essa è vincolata: 8 nella riga, 8 nella colonna, e 4 nella sottogriglia corrispondente alla variabile in questione.

4 Backtrack

Il CSP è stato risolto tramite l'algoritmo di *Backtracking Search* seguendo lo pseudocodice riportato in R&N 2010. In particolare:

- SELECT-UNASSIGNED-VARIABLE(*assignment, Sudoku*):
  è stata implementando seguendo l'euristica **Minimum Remaining Value**, per cui viene selezionata la variabile che nel dizionario *values* ha meno valori disponibili ed è dunque più vincolata;

- ORDER-DOMAIN-VALUES(*var, Sudoku*):
  restituisce semplicemente i valori consistenti di *var* nell'ordine in cui sono riportati nel dizionario *values*;

- INFERENCE*(assignment, inferences, Sudoku, var, value, strategy)*:
  in base al valore di *strategy* (che è 0 o 1) chiama una funzione che implementa le due strategie da confrontare:

  - **Forward-Checking** (*assignment, inferences, Sudoku, var, value*):
    Controlla che, a seguito dell'assegnamento di *var*, nessuna tra le variabili ad essa vincolate abbia un unico valore disponibile e che quel valore sia esattamente

value: in tal caso, restituisce *failure* e si procede al *backtracking*. Se, invece, una variabile V vincolata a *var* può assumere una lista *remaining* di valori, si rimuove *value* da questa lista. Inoltre, se a questo punto *remaining* contiene un unico valore, allora la funzione si richiama per controllare che l'assegnazione forzata di quest' unico valore disponibile per V sia consistente.

– **Maintaning-Arc-Consistency**(*assignment, inferences, Sudoku, var, value*): L'implementazione di questa strategia è analoga a quella dell'algoritmo **AC3** riportato in R&N 2010. L'unica differenza è che, mentre nell' AC3 si inizializza una coda con tutti gli archi del grafo dei vincoli, in questo caso nella coda si mettono soltanto quelli della forma (Xi, *var*), dove Xi è una variabile non ancora assegnata.

5 Test

Per confrontare le due strategie di inferenza sono state utilizzate istanze di Sudoku generate attraverso un sito per giocare online (QQWing Sudoku). In particolare, è stato calcolato e registrato il tempo di risoluzione dei vari Sudoku utilizzando le due diverse strategie per le stesse 25 istanze. L'esperimento è stato ripetuto due volte, la prima volta con istanze con difficoltà bassa, la seconda con difficoltà elevata.

Dallo studio teorico delle due strategie ci si può aspettare che la Forward Checking sia meno costosa, perchè essa esegue le stesse operazioni della strategia MAC ma solo sugli archi con cui in questo algoritmo è inizializzata la coda. Infatti, a differenza della strategia MAC, la Forward checking, qualora vengano fatte modifiche ai domini delle variabili, non propaga ricorsivamente i vincoli inserendo ulteriori archi nella coda.
Come si può notare in Figure 2 e in Figure 3 il tempo calcolato utilizzando la strategia MAC, per quanto variabile, è sempre molto più alto, a volte addirittura il doppio di quello calcolato con la Forward Checking. Per maggior completezza, si riportano i tempi di risoluzione per alcune istanze di Sudoku in Table 2 e in Table 3. Inoltre, è stata anche riportata la media del tempo di risoluzione di tutti i 25 Sudoku in Table 1.

Difficoltà	Forward Checking	MAC
bassa	4.875	12.43
alta	3.124	8.038

Table 1: Media dei tempi di esecuzione calcolati con le due strategie di inferenza

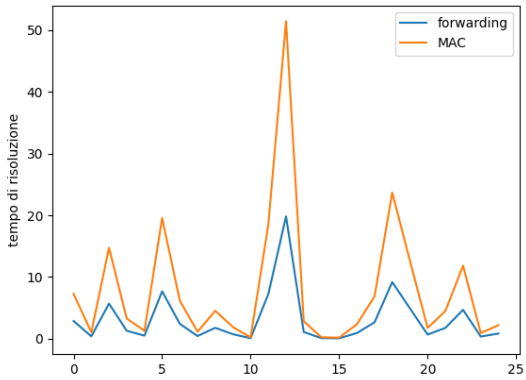

Figure 2: Grafico del confronto tra tempo di risoluzione di 25 istanze di Sudoku di difficoltà bassa calcolato attraverso *Forward Checking* e *MAC*.

Numero istanza	Forward Checking	MAC
2	5.680	14.729
6	2.411	6.163
10	0.0873	7.233
14	0.0914	0.236
19	4.912	12.61
22	4.683	11.84

Table 2: Tempi di risoluzione calcolati con le due strategie di inferenza per alcune istanze di Sudoku con difficoltà bassa.

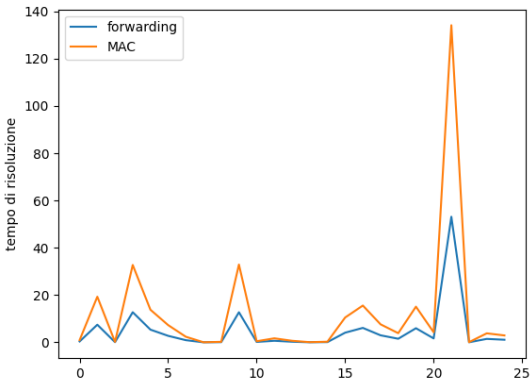

Figure 3: Grafico del confronto tra tempo di risoluzione di 25 istanze di Sudoku di difficoltà alta calcolato attraverso *Forward Checking* e *MAC*.

Numero istanza	Forward Checking	MAC
1	7.462	19.305
5	2.755	7.303
10	0.179	0.472
15	4.091	10.507
20	1.663	4.265
24	1.106	2.922

Table 3: Tempi di risoluzione calcolati con le due strategie di inferenza per alcune istanze di Sudoku con difficoltà elevata.