Applicazione dell'algoritmo Backtracking alla risoluzione di un cruciverba

Ruotolo Samuele

Agosto 2025

1 Introduzione

Per questo esercizio è stato scritto un risolutore di CSP (Constraint Satisfaction Problem). Tale risolutore viene applicato alla ricerca di una soluzione di un cruciverba utilizzando come variabili in un caso le lettere (singole celle della griglia del cruciverba) e in un altro le parole (sequenze di celle della griglia del cruciverba), come indicato alla parte 2 dell'esercizio `https://aimacode.github.io/aima-exercises/csp-exercises/ex_3/`.

2 Procedimento

Il linguaggio di programmazione utilizzato è Python 3.12.

2.1 Generazione dataset

Per eseguire i test è stato utilizzato un dataset di griglie di cruciverba. Per ogni griglia è stato verificato che non esistano celle isolate e, di conseguenza, che la lunghezza minima delle parole nella griglia sia due. Ogni cella della griglia ha una probabilità di 0.5 di essere bloccata (celle nere che si trovano nei cruciverba dove non si possono scrivere lettere). Le griglie hanno dimensioni di (5, 6), (6, 7) e (7, 8) (rispettivamente numero di righe e numero di colonne).
Il dataset delle parole da utilizzare come dizionario di parole ammissibili è stato invece creato a partire dalla lista di parole scaricabile da qui, selezionandole casualmente. A queste sono state aggiunte alcune parole di lunghezza pari a due e a tre, in modo tale da avere a disposizione anche parole di lunghezza minore[1]. Le liste di parole create ed utilizzate sono tre, di dimensioni 50000, 100000 e 150000.
Le griglie di cruciverba e le liste di parole sono state salvate in due file *.txt* che vengono letti dal programma che implementa l'algoritmo.

2.2 Risoluzione

Il risolutore implementa l'algoritmo **Backtracking** basandosi su quanto riportato al paragrafo 6.4 di [1]. Nel processo di inferenza è utilizzato l'algoritmo **AC-3** e, di conseguenza, l'algoritmo **Revise** per la revisione dei domini dopo ogni assegnazione di un valore ad una variabile.
Sono state inoltre utilizzate le euristiche **Minimum Remaining values** o **MRV** per l'ordinamento delle variabili e **Least Constraining Value** o **LCV** per l'ordinamento dei valori da provare.
All'inizio del programma, prima della prima chiamata all'algoritmo **Backtracking**, viene eseguito **AC-3** in modo da ridurre, se possibile, la dimensione dei domini.

[1] Nel file non sono infatti presenti parole di lunghezza due o tre dunque è stato necessario aggiungerle, in modo da permettere la risoluzione dei cruciverba.

2.3 Raccolta dei risultati

La procedura utilizzata per analizzare e confrontare le due implementazioni è la seguente.
Sono stati creati dei CSP che si suddividono in:

- Word CSP, ovvero CSP che utilizzano le parole come variabili;
- Letter CSP, ovvero CSP che ulizzano le singole celle come variabili.

Ogni CSP è collegato quindi ad una griglia di cruciverba e ad una lista di parole: ciò significa che per ogni griglia di cruciverba nel dataset, ci saranno più CSP corrispondenti che differiscono sia per il numero di parole utilizzate sia per la tipologia di implementazione del risolutore[2].
Una volta caricati i dataset dai file *.txt*, come specificato al paragrafo 2.1, viene eseguito l'algoritmo **Backtracking** e i risultati vengono scritti in un file *.txt*. Nel caso di successo, viene riportata la dimensione del cruciverba, il numero di parole utilizzate, la griglia risolta e il tempo di esecuzione. Nel caso di fallimento, invece, viene riporata la dimensione del cruciverba, il numero di parole utilizzate, un messaggio di soluzione non trovata e il tempo di esecuzione.
Per far fronte alla potenza di calcolo limitata dell'elaboratore utilizzato, è stato inserito un limite massimo di step[3] (20000) entro il quale trovare la soluzione. Ciò significa che il fallimento, e di conseguenza il messaggio di errore, si riferiscono sia a problemi per i quali non si è trovata la soluzione, sia a problemi che potrebbero avere soluzione ma necessitano di più tempo e risorse per poterla trovare.

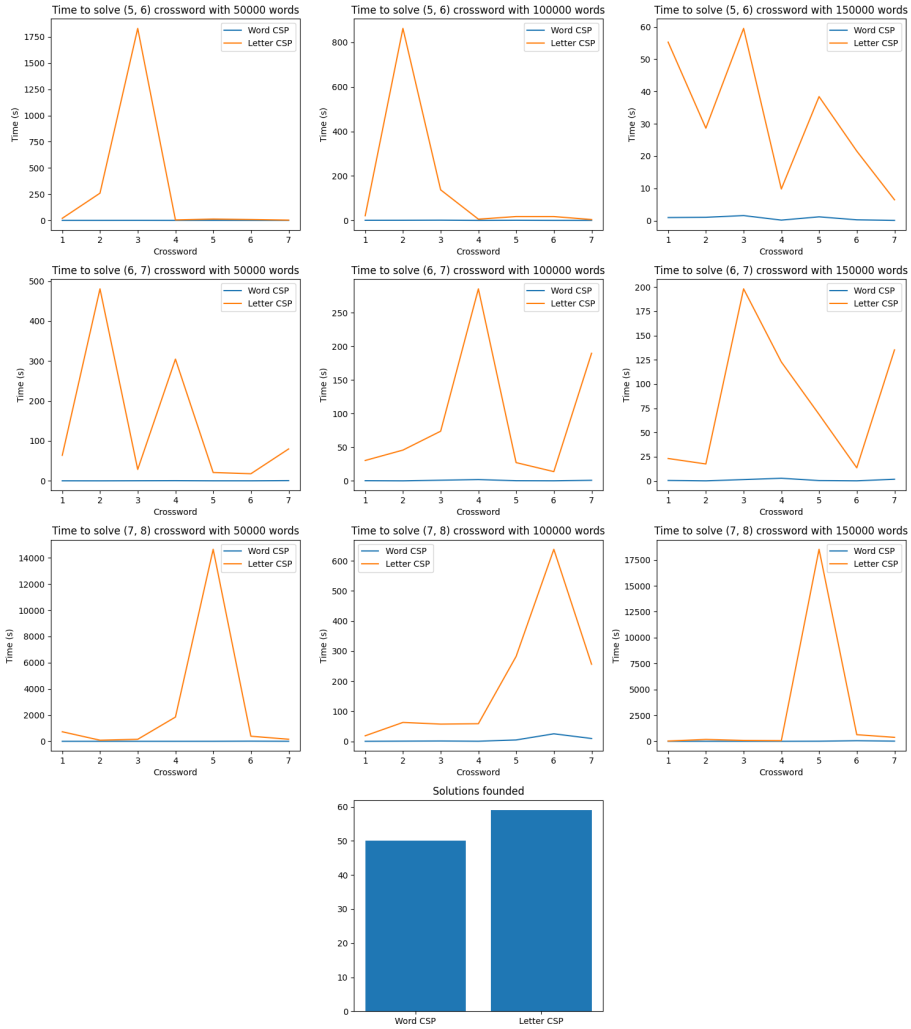
2.4 Plotting dei risultati

I risultati ottenuti, come descritto nel paragrafo precedente, sono stati quindi utilizzati per creare dei grafici tramite la libreria *matplotlib*. In particolare sono stati creati dei grafici per il tempo di esecuzione e per il numero di problemi risolti.
Tali grafici e relativi commenti e riflessioni sono riportati nel paragrafo Risultati.

[2] Ogni griglia di cruciverba è comunque analizzata da entrambe le implementazioni utilizzando le stesse liste di parole.

[3] Uno step corrisponde al passo dell'algoritmo in cui viene scelta la variabile da assegnare.

3 Risultati



Come si può osservare dai grafici, l'implementazione come Word CSP risolve i cruciverba in un tempo medio di esecuzione di **2.61 s** ed ha un tasso di successo del **79%**. L'implementazione come Letter CSP ha invece un tempo medio di esecuzione di **714.60 s** e un tasso di successo del **94%**.
Risulta importante osservare che l'implementazione come Word CSP non risente troppo dell'aumentare del numero di parole usate e della grandezza della griglia, mantenendo tempi di esecuzioni molto simili nei vari casi[4].

[4] In un solo caso si riscontra un tempo di esecuzione molto più alto degli altri ma comunque rimane inferiore ai 60 s.

Per quanto riguarda l'implementazione come Letter CSP, possiamo notare che non sembra esserci un pattern significativo nella variazione dei tempi di esecuzioni in funzione delle dimensioni della griglia o del numero di parole utilizzate ma piuttosto l'aumento dei tempi sembrerebbe poter essere collegato ad una maggiore difficoltà delle griglie da risolvere. Non considerando i due problemi peggiori per tempo di esecuzione, si può osservare che il tempo di esecuzione medio risulta **194.06 s** con un leggero aumento del tasso di successo che passa al **97%**. Tali problemi sono in effetti collegati alla stessa griglia che probabilmente risulta molto più complessa da analizzare per questa implementazione.
Tenendo conto di quanto detto sopra e alla fine del paragrafo 2.3, possiamo concludere che l'implementazione come Word CSP risulta più efficiente nel caso in cui si predilige una risoluzione più veloce possibile accettando che si possano però risolvere meno istanze del problema, mentre l'implementazione come Letter CSP è preferibile quando cerchiamo di risolvere più istanze possibili anche in un tempo maggiore.

References

[1] Stuart Russell e Peter Norvig: traduzione di Francesco Amigoni. *Intelligenza artificale: un approccio moderno. Volume 1.* Pearson Italia, 2021.