Fondamenti di Intelligenza Artificiale M

Tesina di Approfondimento

-

Generatore di Cruciverba

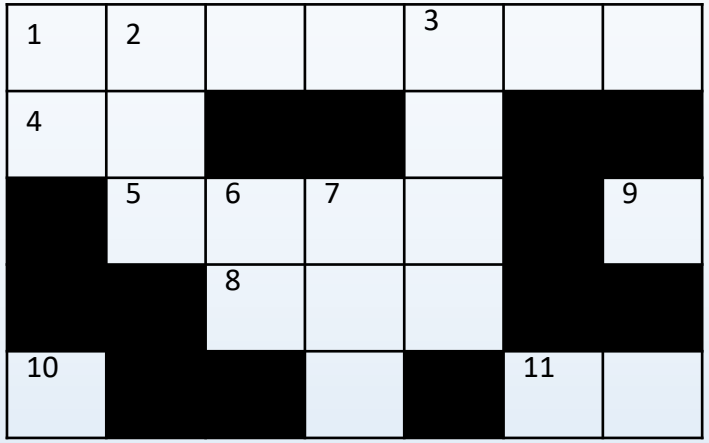
Docente:
Paola Mello

Autore:
Imerio Spagnuolo

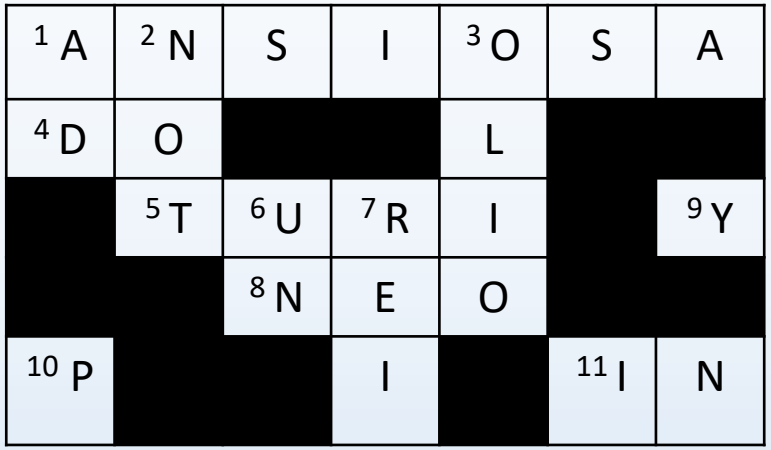
Carattestiche del progetto

- Il progetto è stato realizzato in Java utilizzando la libreria **Aima** (versione 1.8.1) .
- La generazione di un cruciverba è un problema che può essere affrontato in maniera ottimale attraverso tecniche di risoluzione di problemi CSP, ma ho provato ad affrontare il problema anche attraverso gli algoritmi di ricerca per poter confrontare successivamente i risultati ottenuti.
- Dal punto di vista implementativo il modello alla base del progetto è lo stesso sia per la risoluzione tramite CSP sia per la risoluzione come un problema di ricerca.
- Per poter testare la demo ho utilizzato un file di dizionario contenente circa 64mila parole italiane senza apostrofi e lettere accentate.

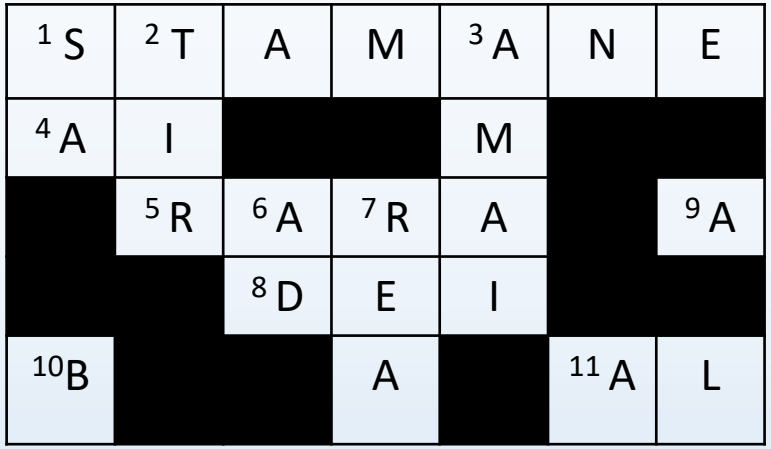
Descrizione generale



Generico cruciverba 5x7 con 11 parole da incastrare



Una possibile soluzione



Un'altra possibile soluzione

Generare un cruciverba significa incastrare delle parole presenti nel dizionario fra di loro.
Ad esempio la parola che corrisponde al 3 verticale deve avere:

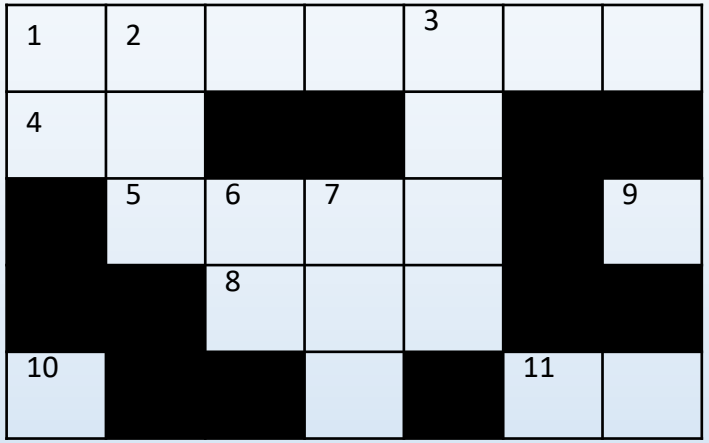
- La prima lettera uguale alla lettera in posizione 5 della parola 1 Orizzontale
- La terza lettera uguale all'ultima lettera della parola 5 orizzontale
- L'ultima lettera uguale all'ultima lettera della parola 8 orizzontale

Inoltre le varie parole devono essere diverse fra loro.

Non esiste un'unica soluzione ottima poiché ogni soluzione può andare bene, a patto che non si introduca il concetto di difficoltà.

Generazione di un Cruciverba come problema di Ricerca nello spazio degli stati

- Stato iniziale:
  - Mappa del cruciverba con le parole vuote
- Funzione successore:
  - Seleziona tutte le parole candidate ad esser inserite nel cruciverba e restituisce una serie di coppie Azione/Stato:
    - **Azione**: Parola da inserire in determinate celle del cruciverba
    - **Stato**: Il nuovo cruciverba con la nuova parola inserita



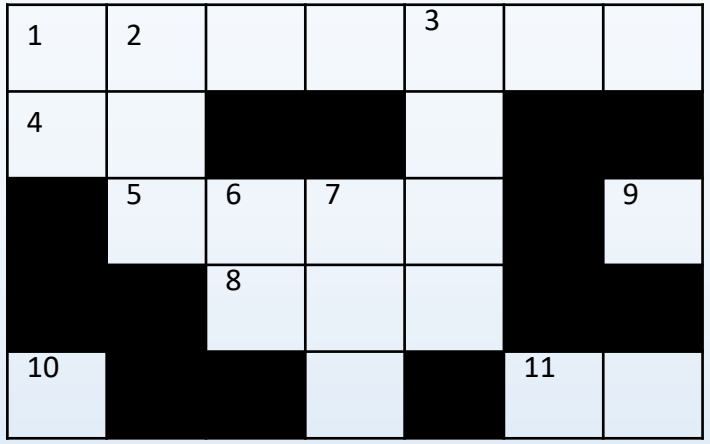
Stato iniziale

- Goal test:
  1. Non devono esserci celle vuote.
  2. Tutte le parole inserite devono essere diverse.
  3. Tutte le parole inserite devono essere presenti nel dizionario.

- Costo:
  - Numero di azioni eseguite.

Calcoli sul costo degli algoritmi di ricerca

- **26 parole di dim: 1**
- **44 parole di dim: 2**
- **170 parole di dim: 3**
- **929 parole di dim: 4**
- 2732 parole di dim: 5
- 4774 parole di dim: 6
- **8637 parole di dim: 7**
- 12015 parole di dim: 8
- 14856 parole di dim: 9
- 15353 parole di dim: 10
- 370 parole di dim: 11
- 226 parole di dim: 12
- 141 parole di dim: 13
- 68 parole di dim: 14
- 44 parole di dim: 15
- 17 parole di dim: 16
- 7 parole di dim: 17



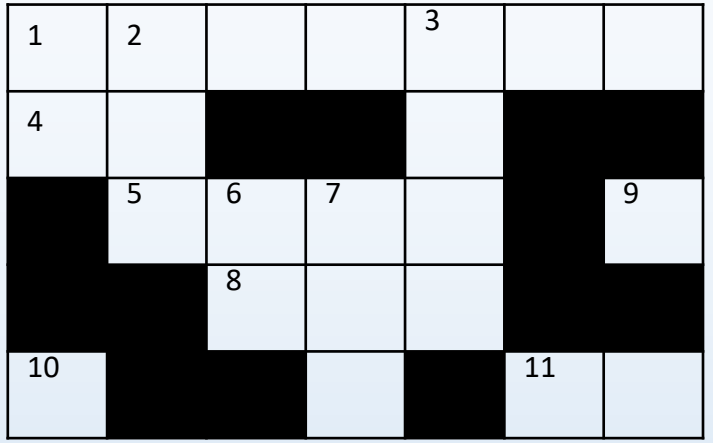
Eseguendo un calcolo approssimativo con riferimento all'esempio in cui vi sono 11 parole da inserire, dallo stato iniziale si possono eseguire oltre 12000 azioni (1 azione = 1 parola candidata ad esser inserita), l'ultimo nodo potrebbe richiedere l'inserimento di una parola da 1 lettera (26 diverse azioni possibili) quindi in media il **fattore di ramificazione** può calcolarsi come: $\frac{12000+26}{11} = 1090$. Dove 11 è la profondità massima.

Ipotizzando che ogni azione richieda 1 picosecondo per essere realizzata, nel caso peggiore ci vogliono: $1090^{11} = 2 * 10^{33}$ ps circa, ovvero $2 * 10^{21}$s, cioè 634 millenni!

Criterio	Depth First	Iterative deepening	Breadth First
Tempo	1090^{11}	1090^{11}	1090^{11}
Memoria	1090*11	1090*11	1090^{11}

Ricerca informata

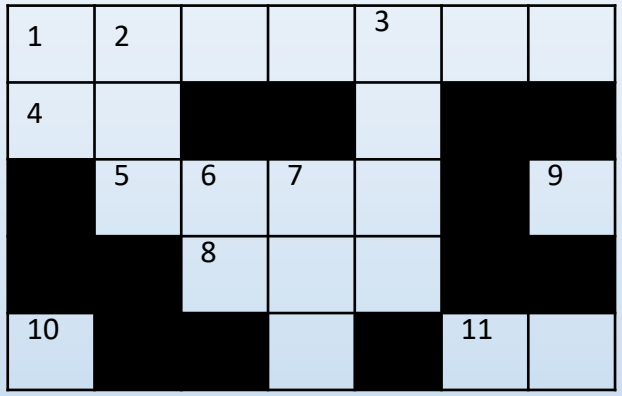
- Possibili euristiche:
  1. Numero di parole totali mancanti. (11).
  2. Numero di parole orizzontali mancanti (5).
  3. Precedenza alle parole che contengono lettere più frequenti.



L'euristica usata consiste in H(n) = numero di parole totali mancanti – peso delle lettere delle parole inserite

Generazione di un Cruciverba come problema di CSP

- Variabili = Parole da inserire
- Domini delle variabili = Parole presenti nel dizionario che possono essere inserite
- Vincoli = Tutte le variabili devono essere diverse, le celle che appartengono a più parole devono avere stesso valore



1 orizzontale = P1.
4 orizzontale = P2.
5 orizzontale = P3.
8 orizzontale = P4.
11 orizzontale = P5.

1 verticale = P6.
10 verticale = P7.
2 verticale = P8.
6 verticale = P9.
7 verticale = P10.

3 verticale = P11.
9 verticale = P12.

- Variabili: P1 , P2, …. , Pk. K = 12 (5 orizzontali e 7 verticali)
- Domini: P1 ϵ {ABBAIAI, ABBAINI, ABBAINO, ABBASSA, ABBASSI,, …. }, … P12 ϵ {A,B,C,…}.
- Vincoli:
  1. P1 ≠ P2 ≠ P3 ≠ P4 ≠ P5 ≠ P6 ≠ P7 ≠ P8 ≠ P9 ≠ P10 ≠ P11 ≠ P12.
  2. P1[0,0] = P6[0,0] = P2[0,0]
  3. P2[1,1] = P7[1,1]
  4. ….

Strategia di risoluzione ottimale

- Forward Checking con euristiche MRV e LVC.
  - MRV, sceglie la variabile con il dominio di cardinalità minore.
  - LVC, sceglie prima il valore che si ritiene abbia più probabilità di successo.

[1] S	[2] T	A	M	[3] A	N	E
[4] A	I	■	■	R	■	■
■	[5] R	[6] I	[7] S	I	■	[9] B
■	■	[8] O	C	A	■	■
[10] A	■	■	I	■	[11] A	L

Assegnamenti:

1. P6 = A
2. P11= B
3. P1 = AI
4. P5 = SA
5. P7 = TIR
6. P8 = IO
7. P2 = RISI
8. P3 = OCA
9. P9 = SCI
10. P10= ARIA
11. P4 = AL
12. P0 = STAMANE

Soluzione trovata in 838 ms con un numero di assegnazioni pari a 12
(uguale al numero di variabili).

Conclusioni

- Generatore di cruciverba affrontabile solo come problema di CSP.
- Forward Checking con euristiche MKV e LCV è risultata la migliore strategia, riesce sempre a trovare una possibile soluzione in tempi ragionevoli anche con un numero di variabili abbastanza elevato.
- Strategia di ricerca locale Min-Conflicts è risultata molto affidabile se il cruciverba ha delle dimensioni piccole (max 20 parole), con l'aumentare delle parole risulta poco efficiente.