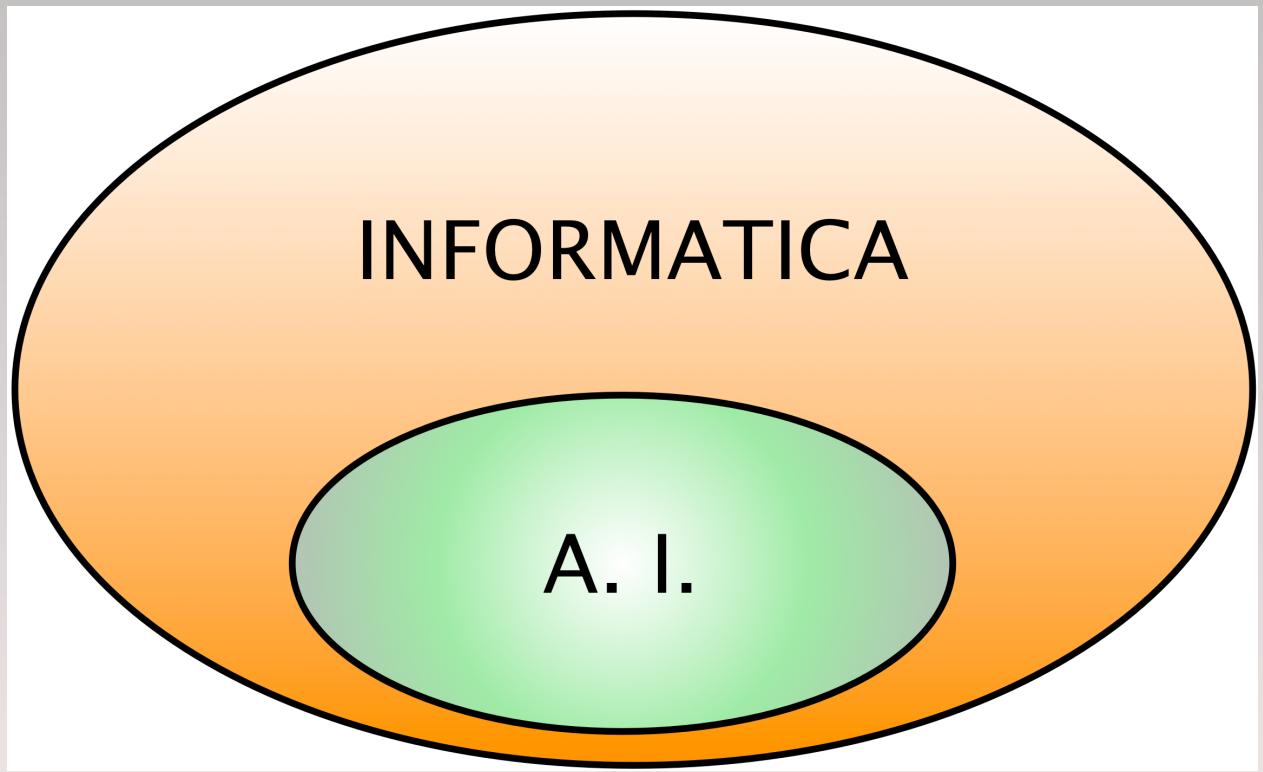
Intelligenza Artificiale: Codifica e Risoluzione di Rompicapi

Agostino Dovier

Dipartimento di Matematica e Informatica
Università di Udine

2014/2015

Artificial Intelligence



L'Intelligenza Artificiale (AI) è una delle principali e più radicate aree dell'Informatica

Informatica

L'informatica (contrazione di informazione automatica) studia i fondamenti teorici, le tecniche e gli strumenti per la gestione ed elaborazione automatica dell'informazione.

Il nome anglosassone è Computer Science — scienza del calcolatore.

Computer Science is no more about computers than astronomy is about telescopes. [Edsger Wybe Dijkstra]

Il padre dell'informatica

Alan Turing (London 1912 - Cheshire 1954)

Il modello matematico del calcolatore universale:
La **macchina di Turing**.

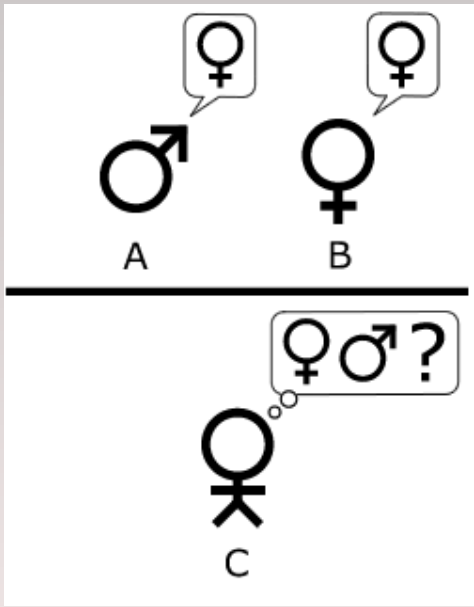
Il padre dell'informatica

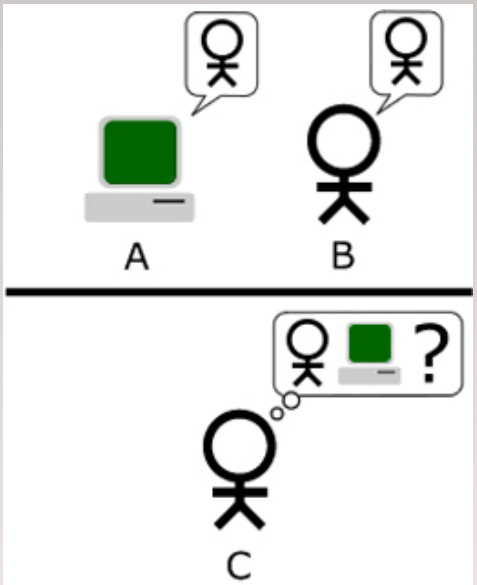
Alan Turing (London 1912 - Cheshire 1954)

Il modello matematico del calcolatore universale:
La **macchina di Turing**.
THE IMITATION GAME è ancora al cinema. Non perdetevelo!!!

Turing e l'Intelligenza Artificiale

Il Test di Turing è un criterio per determinare se una macchina sia in grado di pensare [Mind, 1950]:
I propose to consider the question, "Can machines think?" . . .





A (maschio cerca di ingannare), *B* (femmina, cerca di aiutare).
Una macchina è intelligente se si comporta in modo indistinguibile da *A* nel gioco dell'impersonazione.

La nascita dell'Intelligenza Artificiale

L'espressione "Artificial Intelligence" fu coniata nel 1955 dal matematico americano John McCarthy (a sinistra): "the science and engineering of making intelligent machines"

Minsky (a destra) dice che lo scopo di questa nuova disciplina sarebbe stato quello di "far fare alle macchine delle cose che richiederebbero l'intelligenza se fossero fatte dagli uomini"

AI per risolvere problemi: KR (and reasoning)

Knowledge representation is one of the most important subareas of artificial intelligence. If we want to design an entity (a machine or a program) capable of behaving intelligently in some environment, then we need to supply this entity with sufficient knowledge about this environment. To do that, we need an unambiguous language capable of expressing this knowledge, together with some precise and well understood way of manipulating sets of sentences of the language which will allow us to draw inferences, answer queries, and to update both the knowledge base and the desired program behavior.

Chitta Baral and Michael Gelfond. JLP 19/20:73–148, 1994.

Expressing information in declarative sentences is far more modular than expressing it in segments of computer programs or in tables. Sentences can be true in a much wider context than specific programs can be used. The supplier of a fact does not have to understand much about how the receiver functions or how or whether the receiver will use it. The same fact can be used for many purposes, because the logical consequences of collections of facts can be available. [McC59]

J. McCarthy. Programs with Common Sense, 1959.

AI per risolvere problemi: Linguaggi

Vengono introdotti linguaggi di programmazione adatti alla rappresentazione della conoscenza e al ragionamento su essa.

Mc Carthy propone il LISP (LISt Processor) nel 1958.

Kowalski propone il PROLOG (PROgramming with LOGic) nel 1974.

Le comunità di Programmazione Funzionale (LISP) e di Programmazione Logica (PROLOG) sono molto attive ancora oggi. Da questi due linguaggi nascono un numero enorme di nuovi loro dialetti e altri linguaggi (dichiarativi) a loro ispirati.

AI per risolvere problemi: Search

L'idea di questi linguaggi è che il programmatore deve concentrarsi sulla rappresentazione del problema (rappresentazione della conoscenza–KR).

Il linguaggio possiede tecniche di ragionamento che, data la codifica del problema cerca la soluzione (search).

Nasce allora il problema per l'informatico di studiare tecniche efficienti e generali da inserire come cuore "intelligente" di questi linguaggi.

Ma come si fa a capire se le tecniche sviluppate sono buone o meno?

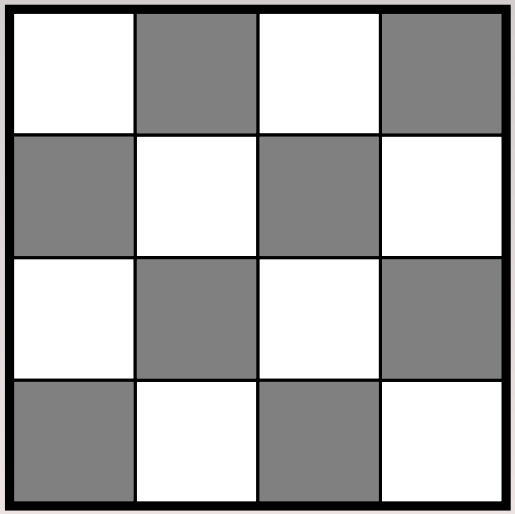
Qui entrano in campo le sfide dei sistemi contro i rompicapi!

Sono sfide in espressività (il linguaggio che permette la codifica più bella) e soprattutto in velocità (tempo necessario a trovare le soluzioni su esempi).

Esempio: le *N* regine

Cerchiamo di intuire alcune delle tecniche concentrandosi su un problemino.

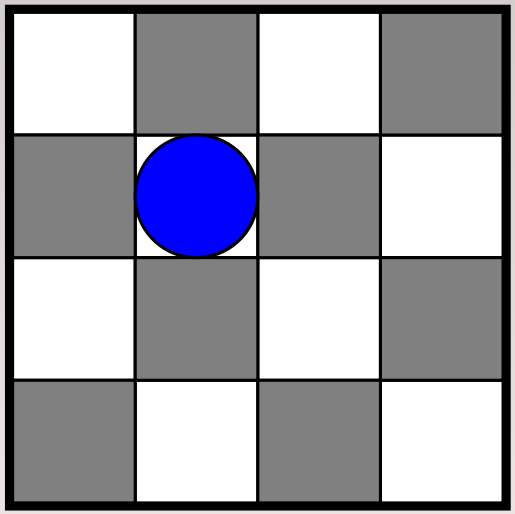
4-Regine: sistemare 4 regine su una scacchiera 4×4 in modo tale che le regine non si attacchino.



Esempio: le *N* regine

Cerchiamo di intuire alcune delle tecniche concentrandosi su un problemino.

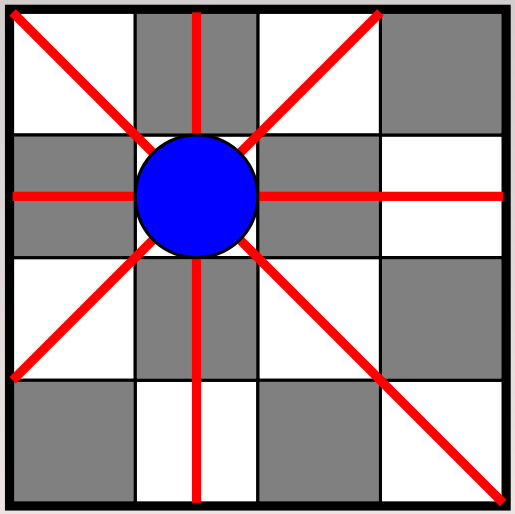
4-Regine: sistemare 4 regine su una scacchiera 4×4 in modo tale che le regine non si attacchino.



Esempio: le *N* regine

Cerchiamo di intuire alcune delle tecniche concentrandosi su un problemino.

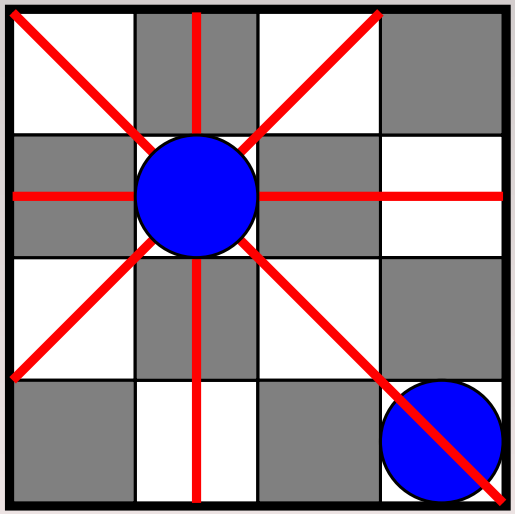
4-Regine: sistemare 4 regine su una scacchiera 4×4 in modo tale che le regine non si attacchino.



Esempio: le *N* regine

Cerchiamo di intuire alcune delle tecniche concentrandosi su un problemino.

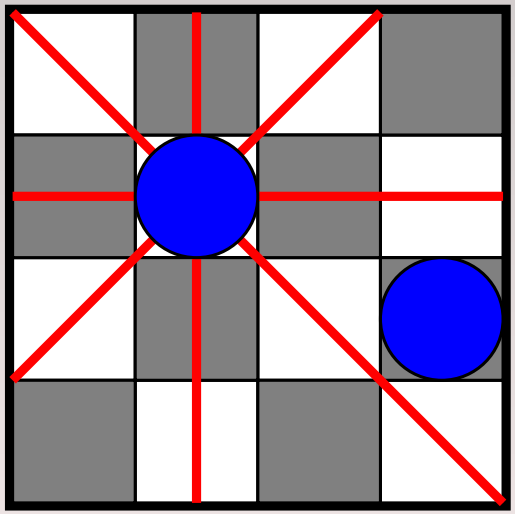
4-Regine: sistemare 4 regine su una scacchiera 4×4 in modo tale che le regine non si attacchino.



Esempio: le *N* regine

Cerchiamo di intuire alcune delle tecniche concentrandosi su un problemino.

4-Regine: sistemare 4 regine su una scacchiera 4×4 in modo tale che le regine non si attacchino.



Codifica (in linguaggi “a vincoli”)

- Variabili: $X_1, \ldots, X_4$
- Domini (valori possibili):

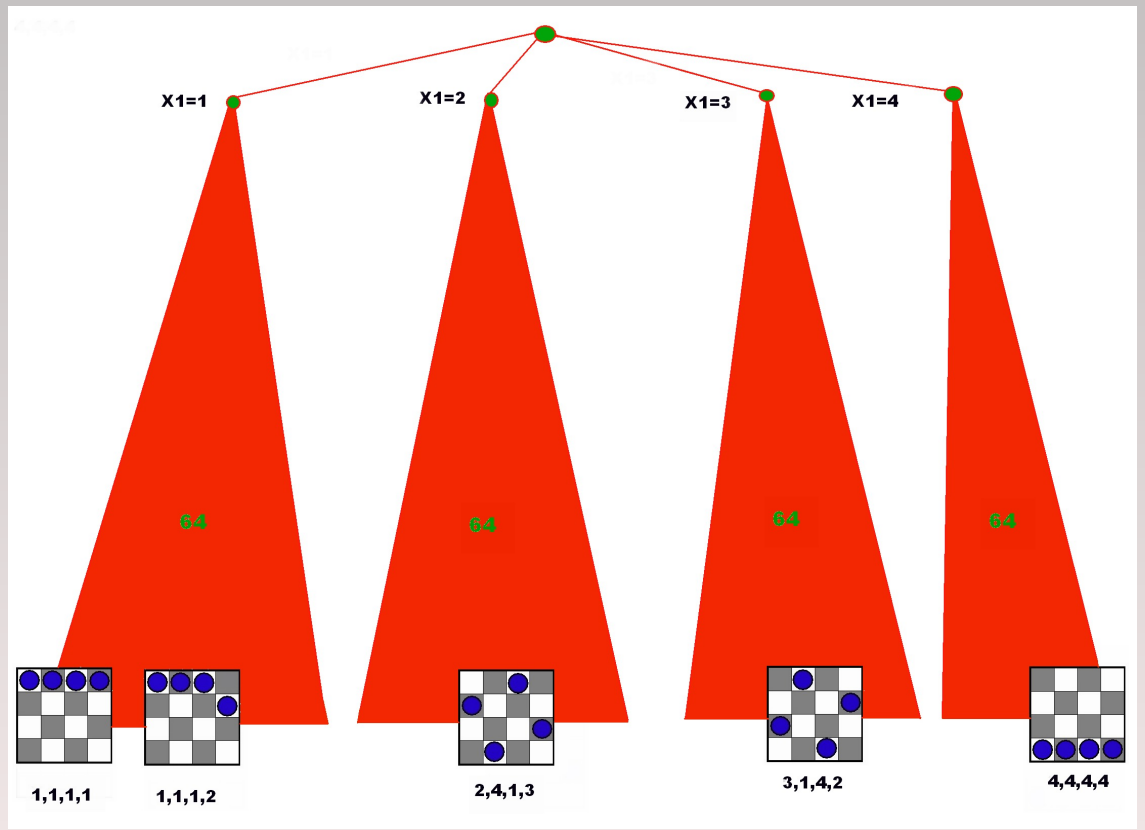
$$Dom(X_1) = Dom(X_2) = Dom(X_3) = Dom(X_4) = \{1, \ldots, 4\}$$

  $X_i = j$ significa “una regina sta in colonna i, riga j”
- Vincoli (per $i, j = 1..4$ e $i < j$)

$$X_i \neq X_j \quad \text{(attacco orizzontale)}$$
$$|X_j - X_i| \neq j - i \quad \text{(attacco in diagonale)}$$

Ricerca delle soluzioni

Forza Bruta: Generate and Test



Si visita tutto l'albero di ricerca:

Si rischia di fare $4^4 = 256$ tentativi (N^N in generale)

Ricerca delle soluzioni

La crescita esponenziale

- Quant'è spesso un foglio di carta? Provate a misurare lo spessore di un quaderno e a dividere per il numero di fogli.

Ricerca delle soluzioni

La crescita esponenziale

- Quant'è spesso un foglio di carta? Provate a misurare lo spessore di un quaderno e a dividere per il numero di fogli.
  Assumiamo sia $\frac{1}{10} mm$.

Ricerca delle soluzioni

La crescita esponenziale

- Quant'è spesso un foglio di carta? Provate a misurare lo spessore di un quaderno e a dividere per il numero di fogli. Assumiamo sia $\frac{1}{10}mm$.

1. Pieghiamo un foglio in 2. Diventa spesso $\frac{2}{10}mm = \frac{2^1}{10}mm$

Ricerca delle soluzioni

La crescita esponenziale

- Quant'è spesso un foglio di carta? Provate a misurare lo spessore di un quaderno e a dividere per il numero di fogli. Assumiamo sia $\frac{1}{10}mm$.

1. Pieghiamo un foglio in 2. Diventa spesso $\frac{2}{10}mm = \frac{2^1}{10}mm$
2. Pieghiamo ancora in 2. Diventa ora (attenzione!) $\frac{4}{10}mm = \frac{2^2}{10}mm$

Ricerca delle soluzioni

La crescita esponenziale

- Quant'è spesso un foglio di carta? Provate a misurare lo spessore di un quaderno e a dividere per il numero di fogli.
  Assumiamo sia $\frac{1}{10}mm$.

1. Pieghiamo un foglio in 2. Diventa spesso $\frac{2}{10}mm = \frac{2^1}{10}mm$
2. Pieghiamo ancora in 2. Diventa ora (attenzione!) $\frac{4}{10}mm = \frac{2^2}{10}mm$
3. Pieghiamo ancora in 2. Diventa ora $\frac{8}{10}mm = \frac{2^3}{10}mm$

Ricerca delle soluzioni

La crescita esponenziale

- Quant'è spesso un foglio di carta? Provate a misurare lo spessore di un quaderno e a dividere per il numero di fogli.
  Assumiamo sia $\frac{1}{10}mm$.

1. Pieghiamo un foglio in 2. Diventa spesso $\frac{2}{10}mm = \frac{2^1}{10}mm$
2. Pieghiamo ancora in 2. Diventa ora (attenzione!) $\frac{4}{10}mm = \frac{2^2}{10}mm$
3. Pieghiamo ancora in 2. Diventa ora $\frac{8}{10}mm = \frac{2^3}{10}mm$
4. Pieghiamo ancora in 2. Diventa ora $\frac{16}{10}mm = \frac{2^4}{10}mm = 1.6mm$

Ricerca delle soluzioni

La crescita esponenziale

- Quant'è spesso un foglio di carta? Provate a misurare lo spessore di un quaderno e a dividere per il numero di fogli.
  Assumiamo sia $\frac{1}{10}mm$.

1. Pieghiamo un foglio in 2. Diventa spesso $\frac{2}{10}mm = \frac{2^1}{10}mm$
2. Pieghiamo ancora in 2. Diventa ora (attenzione!) $\frac{4}{10}mm = \frac{2^2}{10}mm$
3. Pieghiamo ancora in 2. Diventa ora $\frac{8}{10}mm = \frac{2^3}{10}mm$
4. Pieghiamo ancora in 2. Diventa ora $\frac{16}{10}mm = \frac{2^4}{10}mm = 1.6mm$
5. Pieghiamo ancora in 2. Diventa ora $\frac{32}{10}mm = \frac{2^5}{10}mm = 3.2mm$

La crescita esponenziale

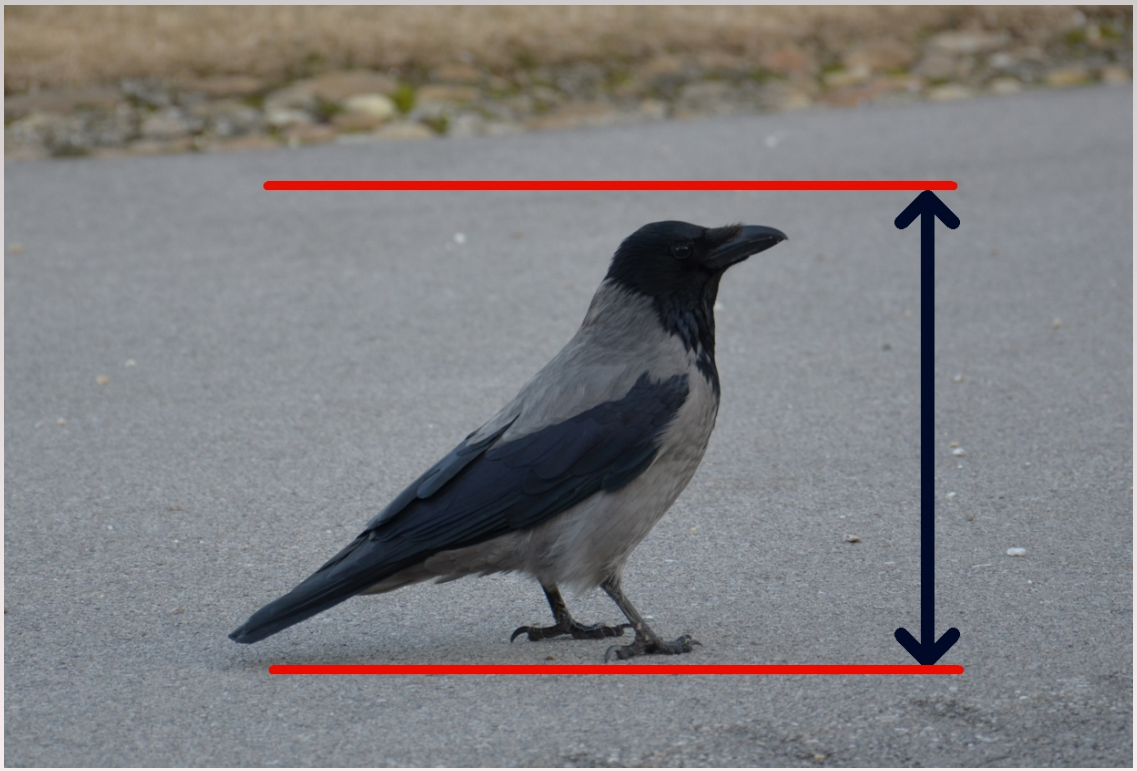
11 piegature

$$\frac{2^{11}}{10} mm = \frac{2048}{10} mm \approx 20 cm$$

La crescita esponenziale

11 piegature

$$\frac{2^{11}}{10} mm = \frac{2048}{10} mm \approx 20 cm$$



La crescita esponenziale

14 piegature

$$\frac{2^{14}}{10} mm = \frac{16384}{10} mm \approx 1.64 m$$

La crescita esponenziale

14 piegature

$$\frac{2^{14}}{10} mm = \frac{16384}{10} mm \approx 1.64m$$

La crescita esponenziale

15 piegature

$$\frac{2^{15}}{10} mm = \frac{32768}{10} mm \approx 3.27 m$$

La crescita esponenziale

15 piegature

$$\frac{2^{15}}{10} mm = \frac{32768}{10} mm \approx 3.27 m$$

La crescita esponenziale

20 piegature

$$\frac{2^{20}}{10} mm = \frac{1048576}{10} mm \approx 104m$$

La crescita esponenziale

20 piegature

$$\frac{2^{20}}{10} mm = \frac{1048576}{10} mm \approx 104m$$

La crescita esponenziale

42 piegature

$$\frac{2^{42}}{10} mm \approx 439804 Km$$

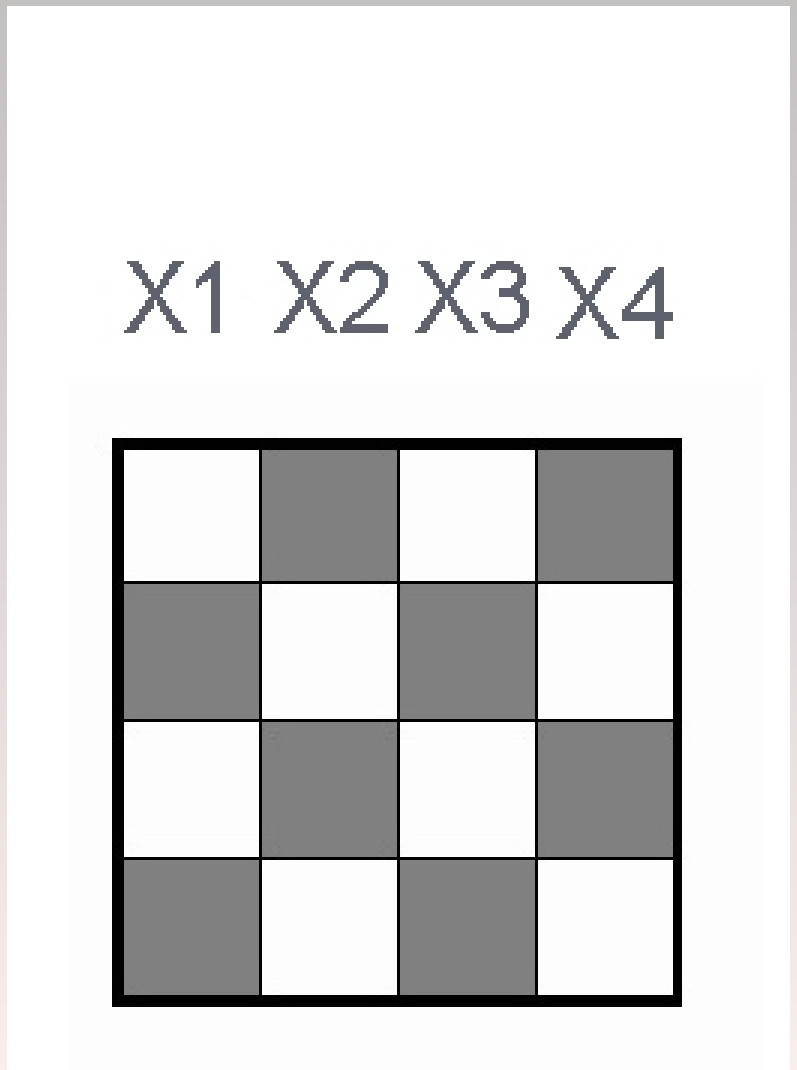
La crescita esponenziale

42 piegature

$$\frac{2^{42}}{10} mm \approx 439804 Km$$

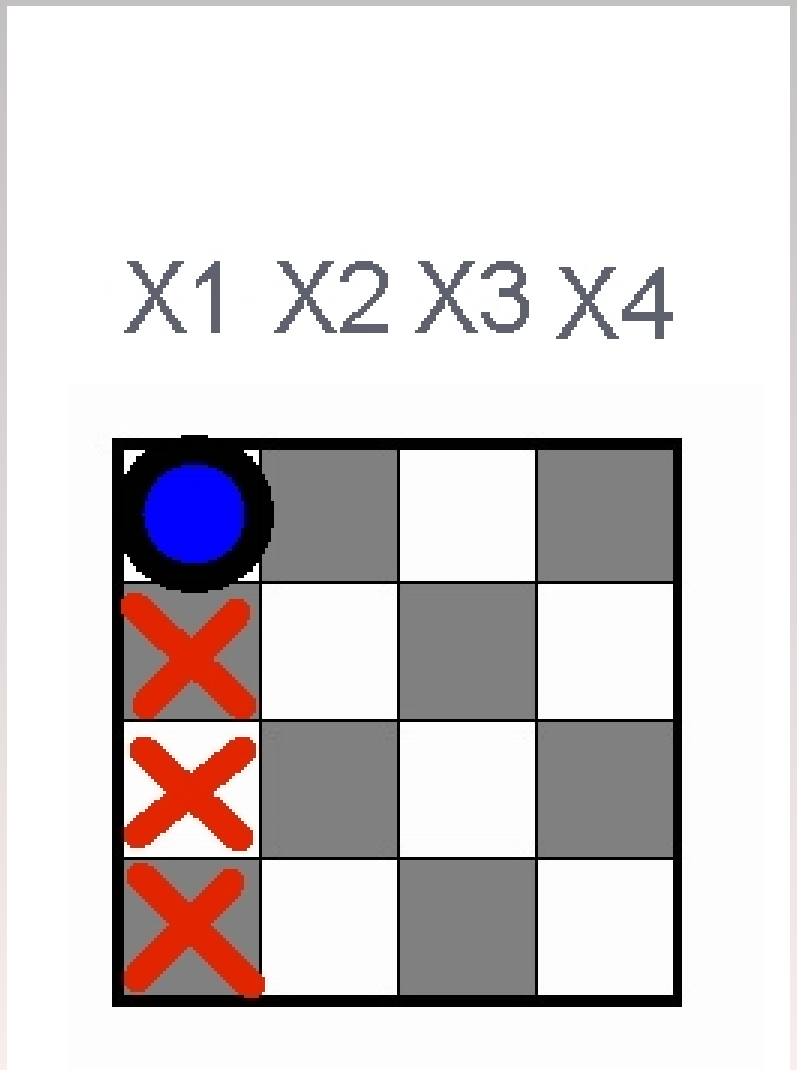
Constraint Propagation

Assegnamo $X_1 = 1$



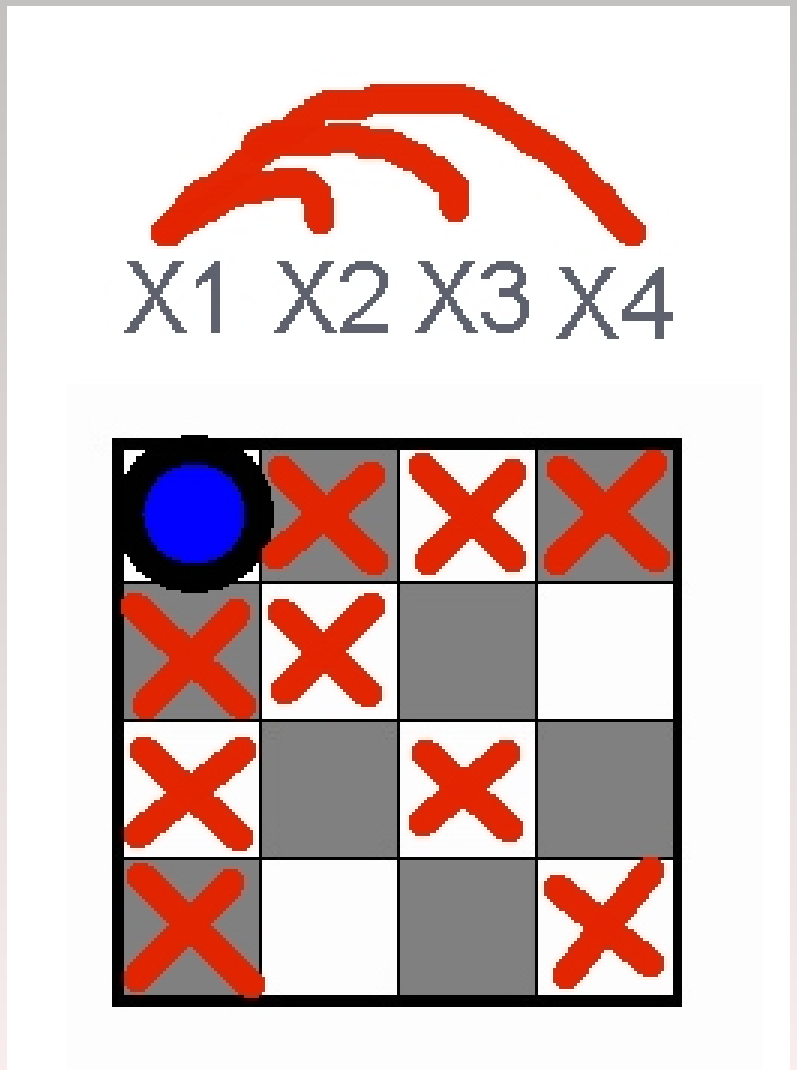
Constraint Propagation

Assegnamo $X_1 = 1$



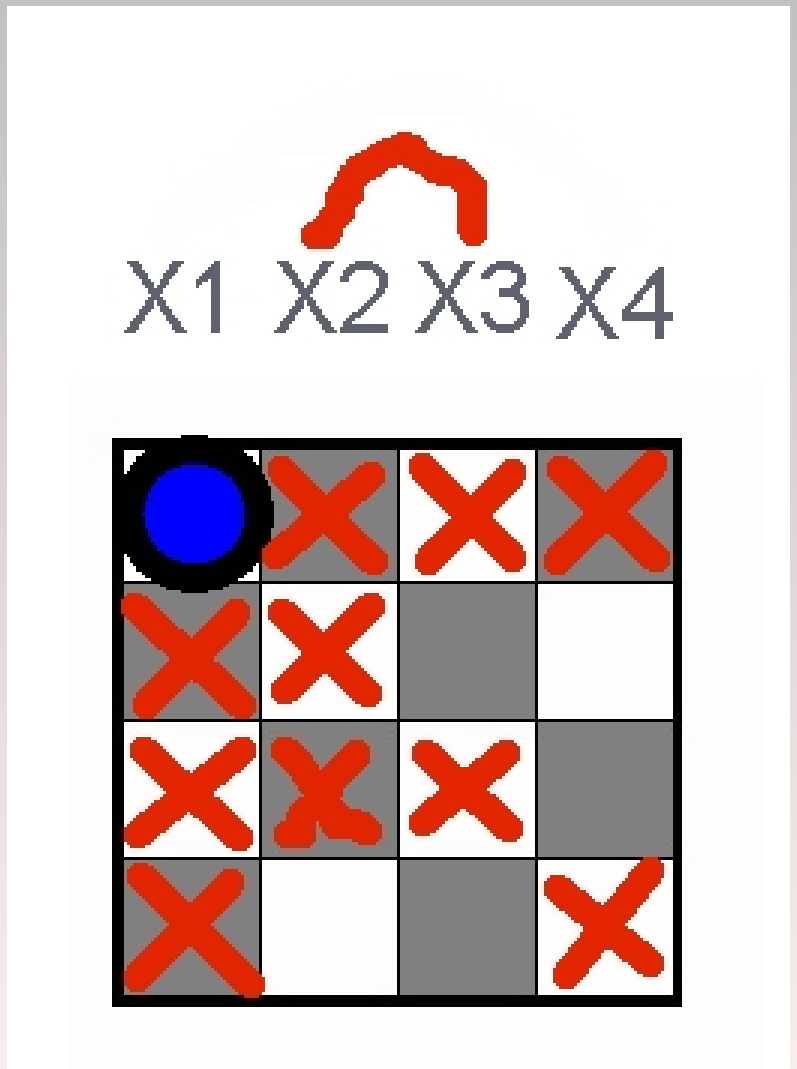
Constraint Propagation

Assegnamo $X_1 = 1$



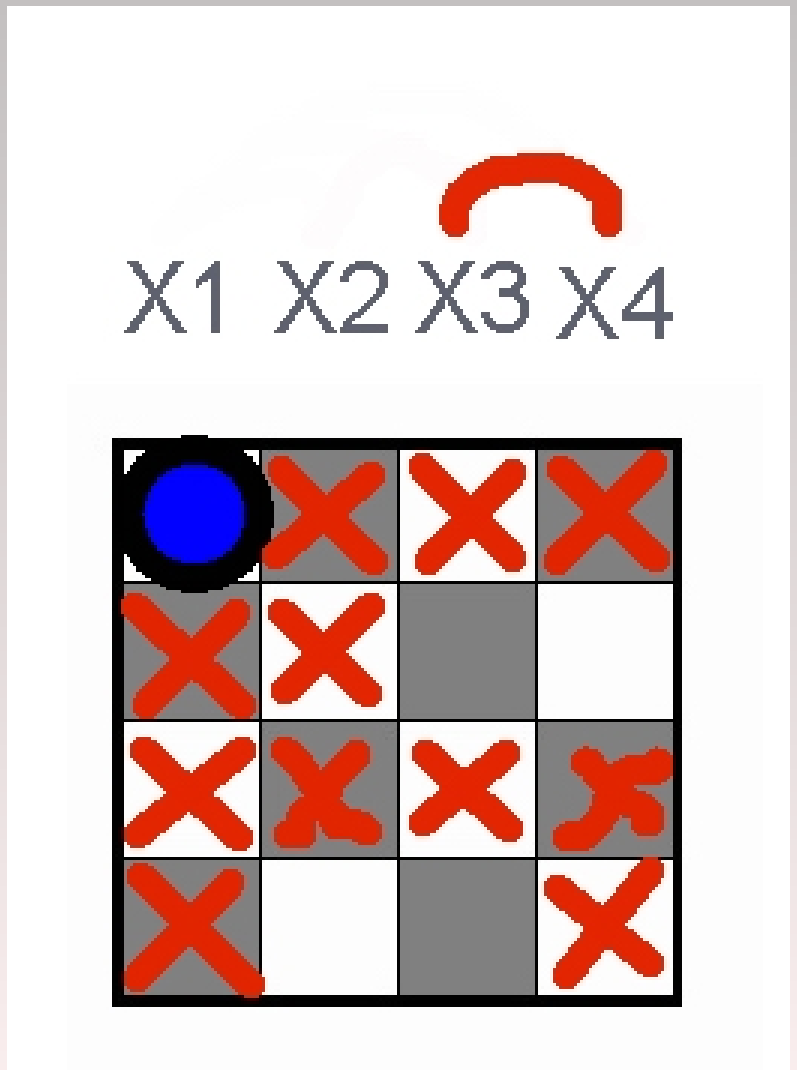
Constraint Propagation

Assegnamo $X_1 = 1$



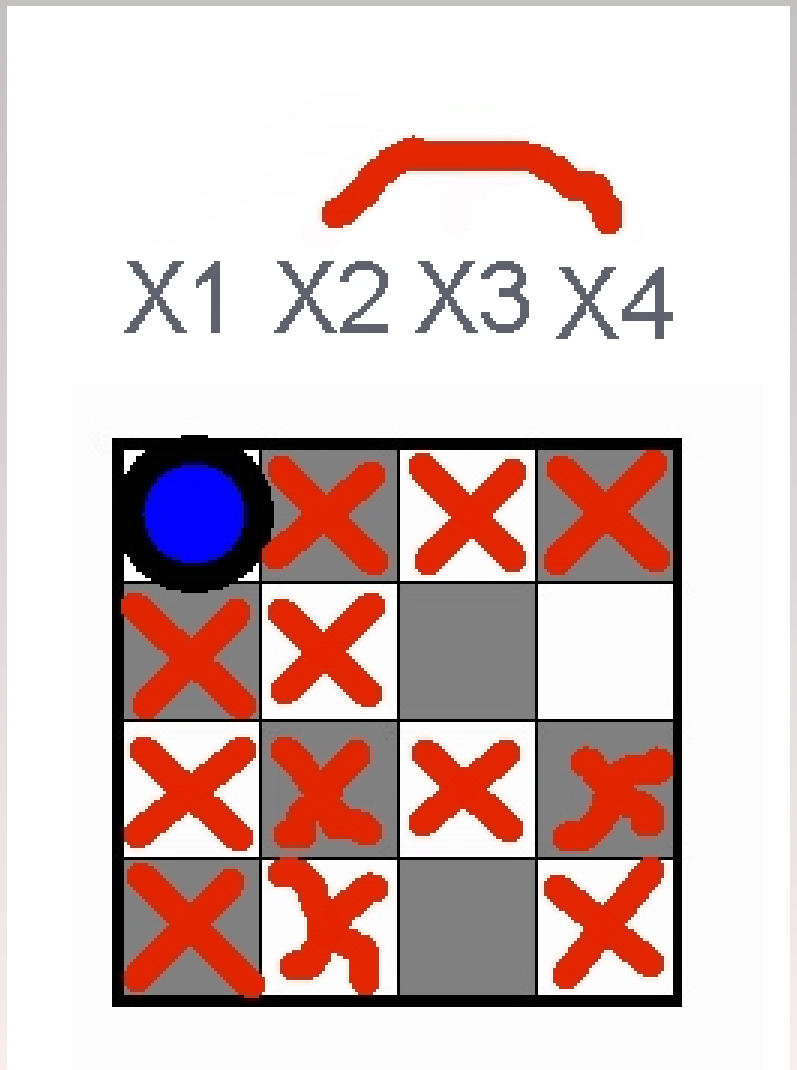
Constraint Propagation

Assegnamo $X_1 = 1$



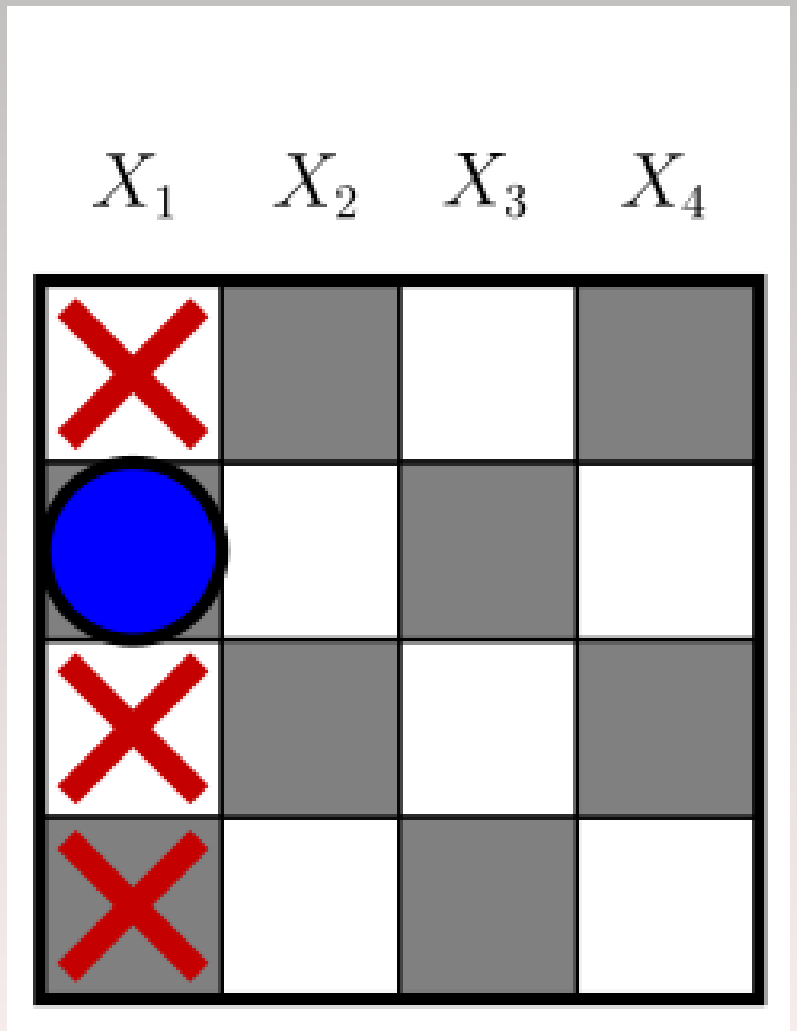
Constraint Propagation

Assegnamo $X_1 = 1$



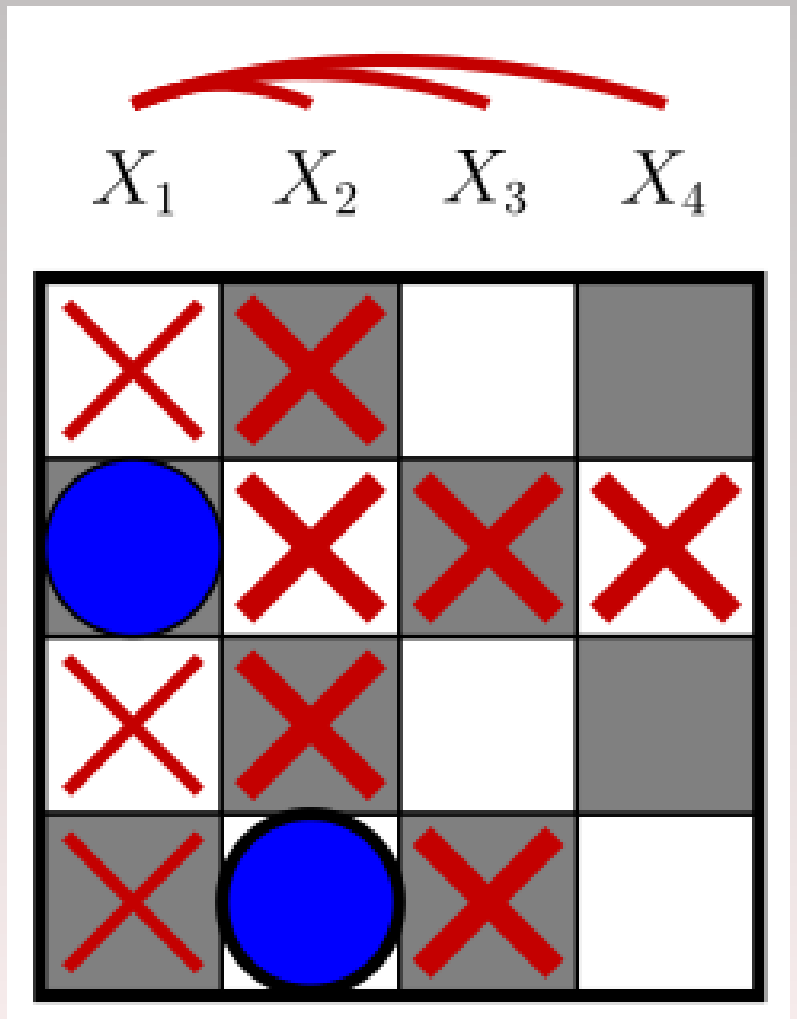
Constraint Propagation

Assegnamo: $X_1 = 2$



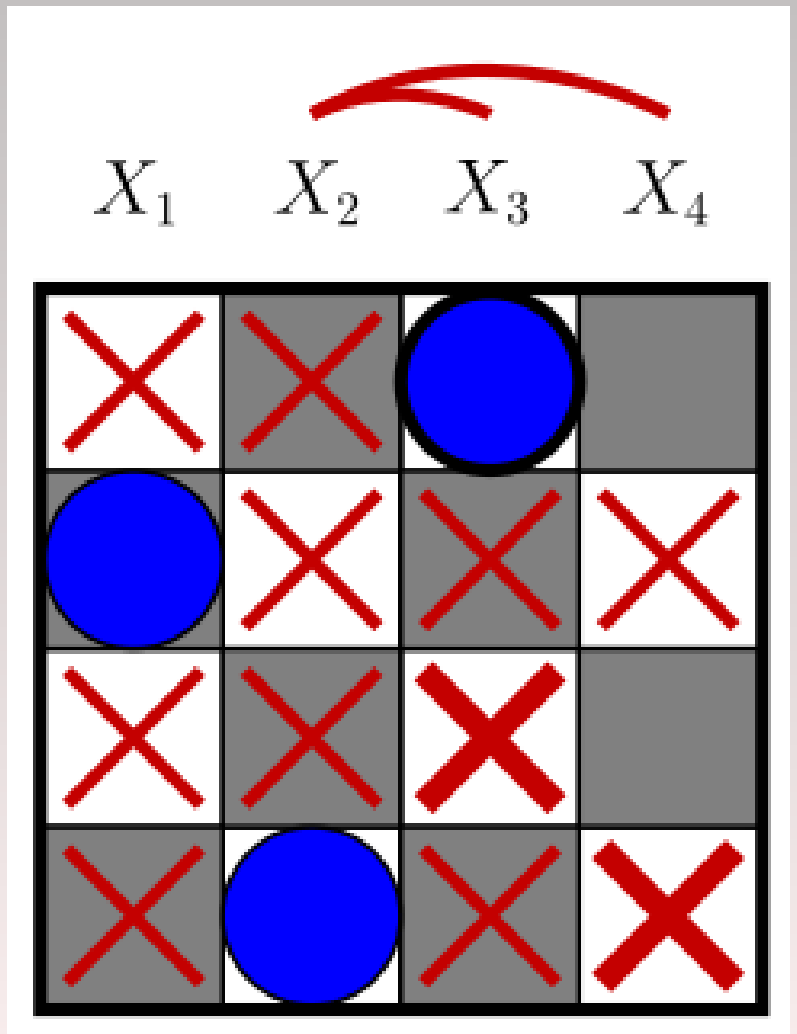
Constraint Propagation

Assegnamo: $X_1 = 2$



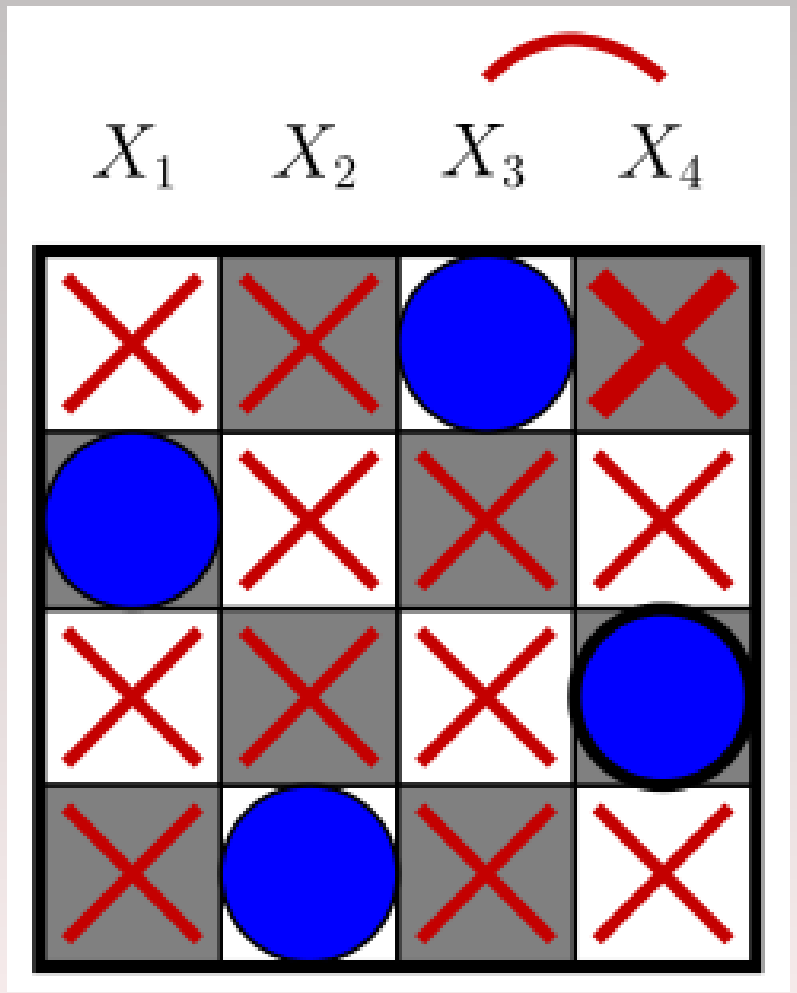
Constraint Propagation

Assegnamo: $X_1 = 2$



Constraint Propagation

Assegnamo: $X_1 = 2$



Search & Constraint Propagation

Con la sola idea di alternare assegnamenti e propagazioni sono passato da 256 tentativi a 2.

Il tentativo è a caso, la propagazione simula (con buoni risultati) un ragionamento

I linguaggi per l'AI contengono molte tecniche come queste (spesso invisibili al programmatore) per trovare le soluzioni ai problemi codificati.

Quali problemi?

Esempi di problemi AI

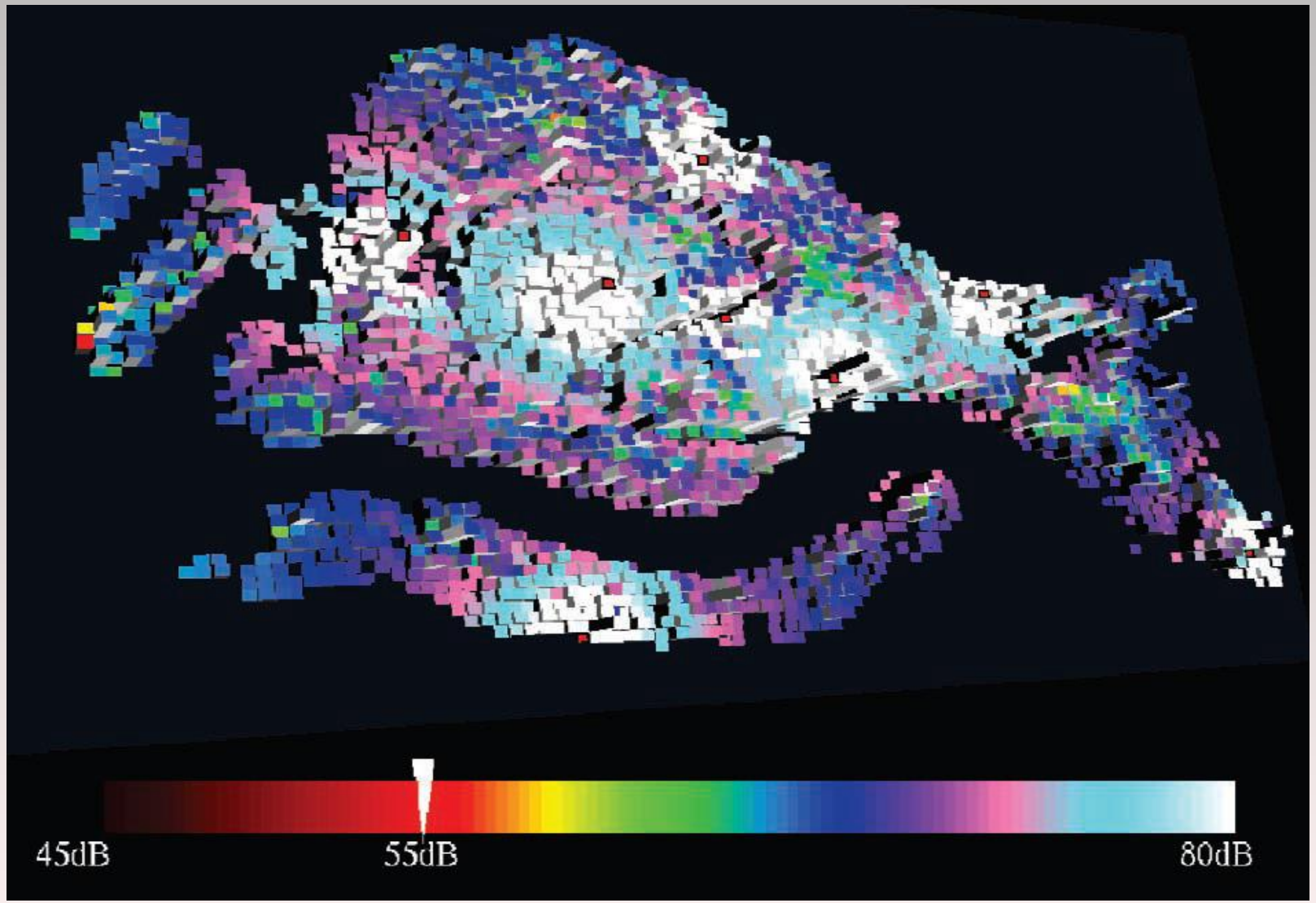
Esempi di problemi AI

Esempi di problemi AI

Esempi di problemi AI

Esempi di problemi AI

Esempi di problemi AI



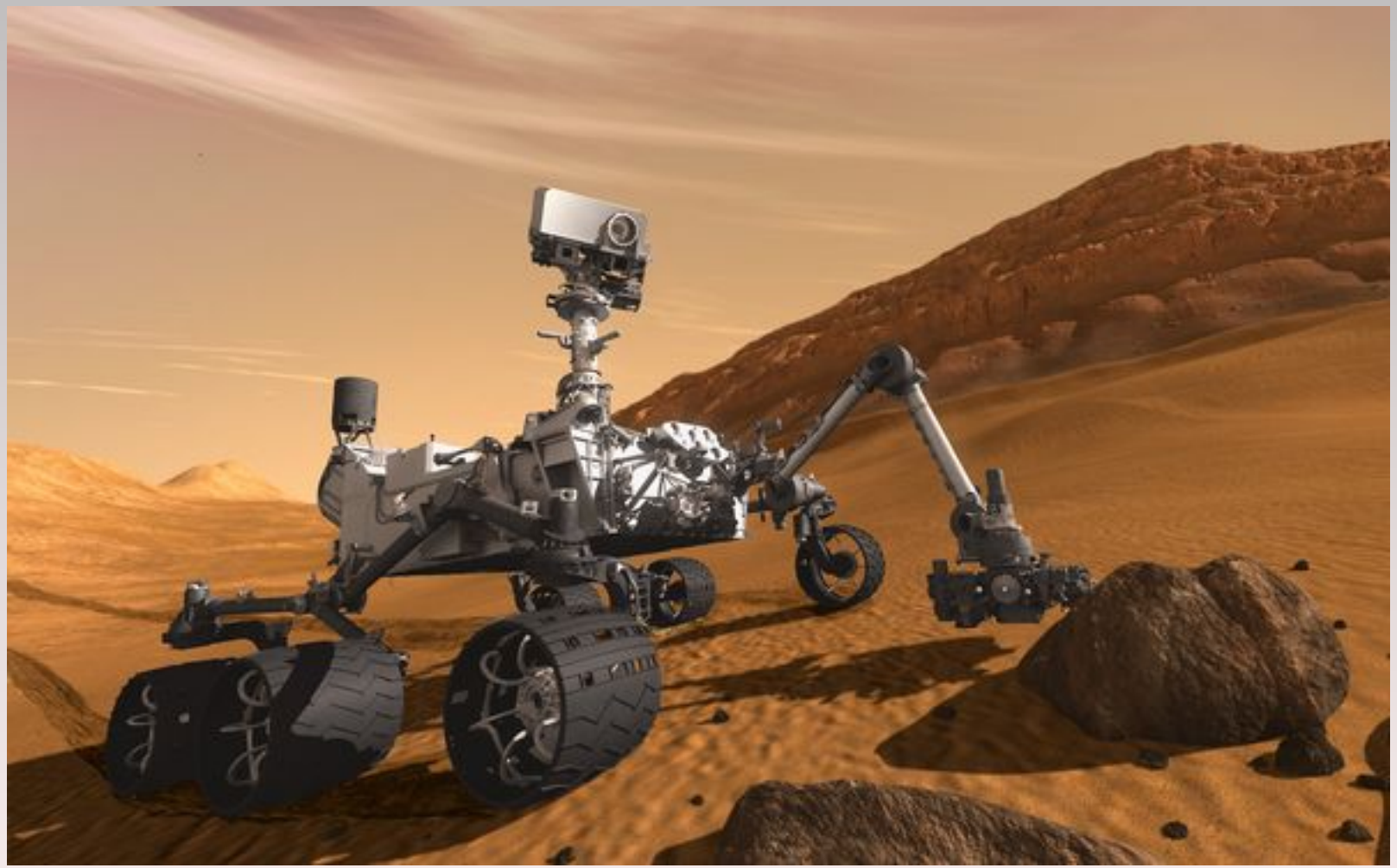
Esempi di problemi AI

Esempi di problemi AI

Esempi di problemi AI

Esempi di problemi AI

Esempi di problemi AI



Rompicapi come benchmarks

Come dicevamo, per capire se le tecniche sviluppate sono buone o meno si sfidano i linguaggi (o i solver) su una serie di problemi detti benchmarks.

Nelle varie competizioni, gran parte dei benchmarks sono dei giochi (dei rompicapi)

Vediamone alcuni.

La capra e il cavolo

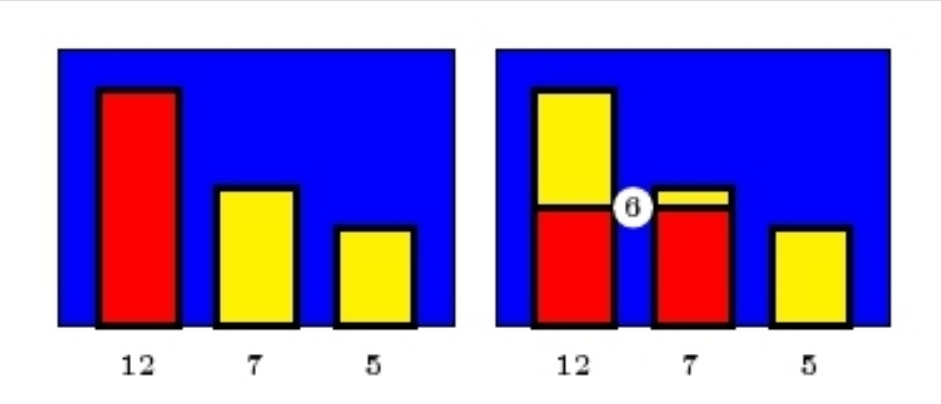
Un pastore deve attraversare un fiume portando sull'altra riva un lupo e una capra affamati e un cavolo gigante. Ha a disposizione una barca a remi con la quale può traghettare un solo oggetto o animale alla volta. Ma, attenzione, non può lasciare da soli:

- il lupo e la capra perché il lupo si mangia la capra;
- la capra ed il cavolo pechè la capra si mangia il cavolo.

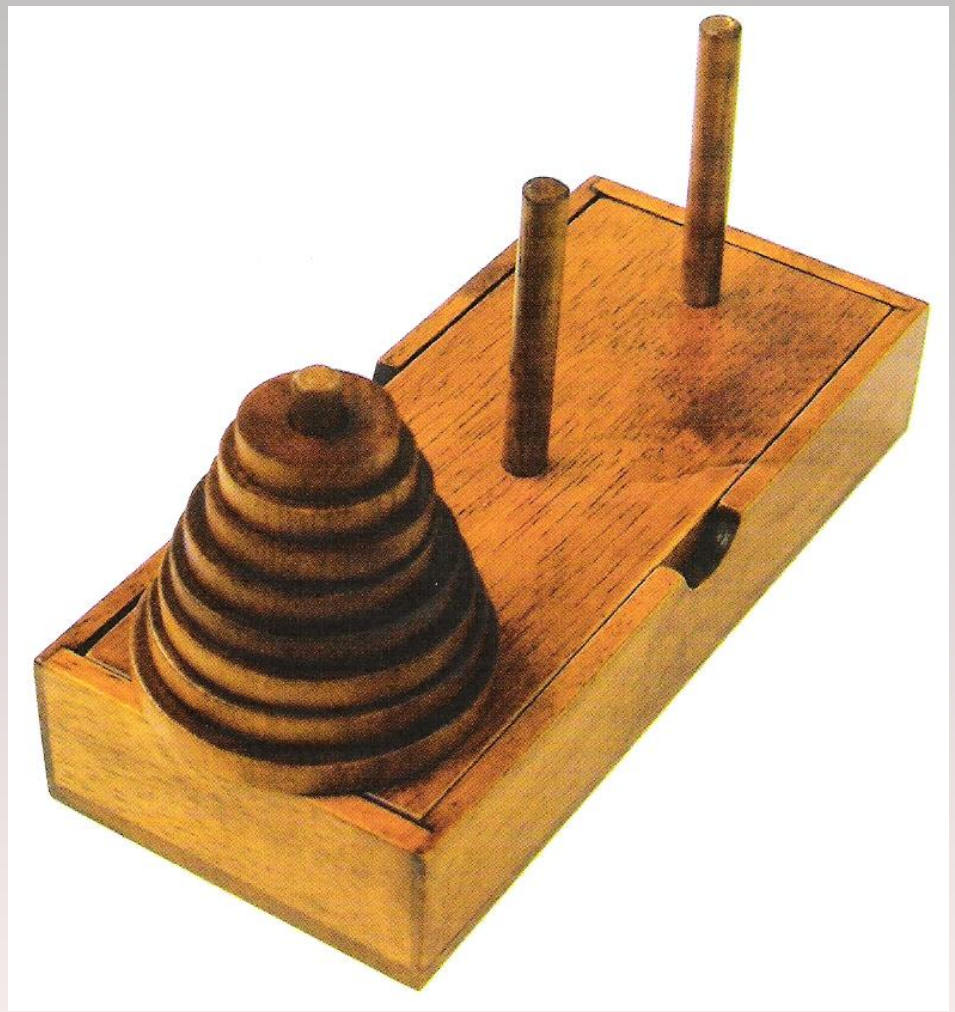
Quanti viaggi deve fare per portare sull'altra riva il lupo, la capra ed il cavolo?

The three barrels

"Ci sono tre botti di capacità N (pari), $N/2+1$, e $N/2-1$ (esempio: 12,7,5). All'inizio la botte più capace è piena di vino e le altre sono vuote. Si vuol raggiungere una conformazione in cui la più piccola è vuota e le altre due contengono esattamente la stessa quantità. Le sole azioni permesse sono di svuotare una botte in un'altra (non è permesso di misurare i flussi o l'altezza del vino o pesare le botti)."

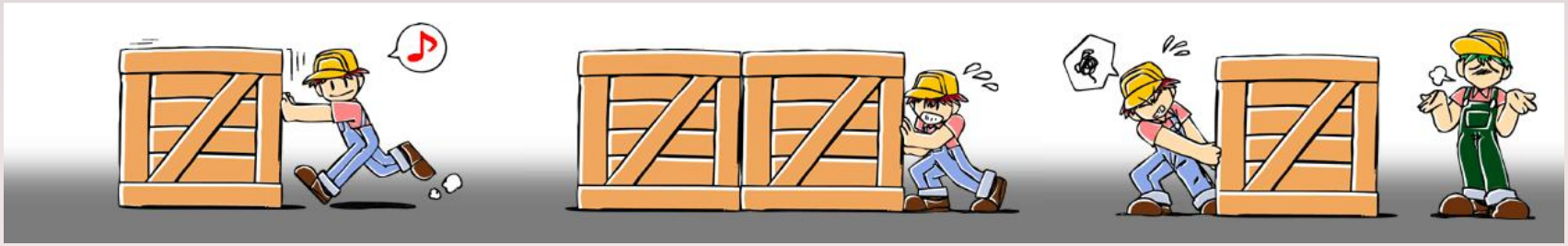


La torre di Hanoi

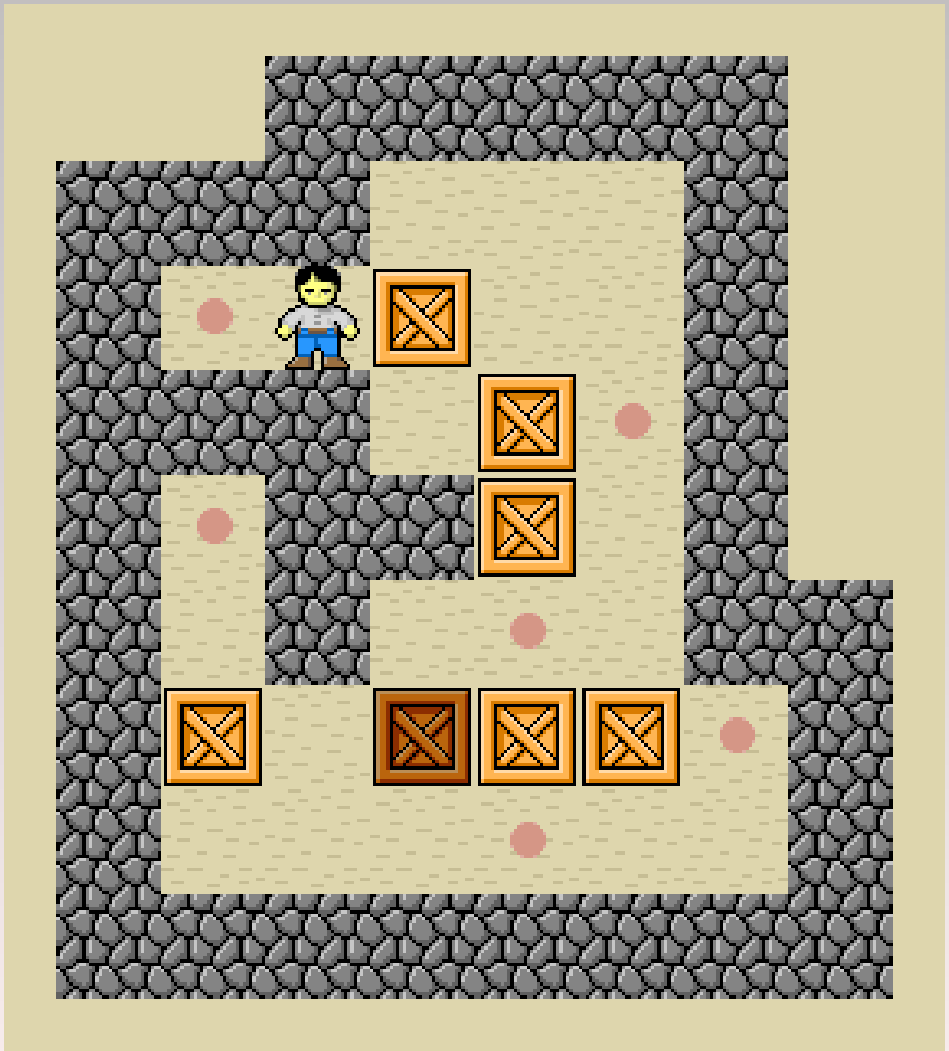


Sokoban

- Sokoban è un rompicapo che coinvolge problemi di trasporto, magazzini, robotica inventato da *Hiroyuki Imabayashi* nel 1980
- Sokoban significa magazziniere in Giapponese
- E' uno dei primi videogiochi. Ha solo tre regole:

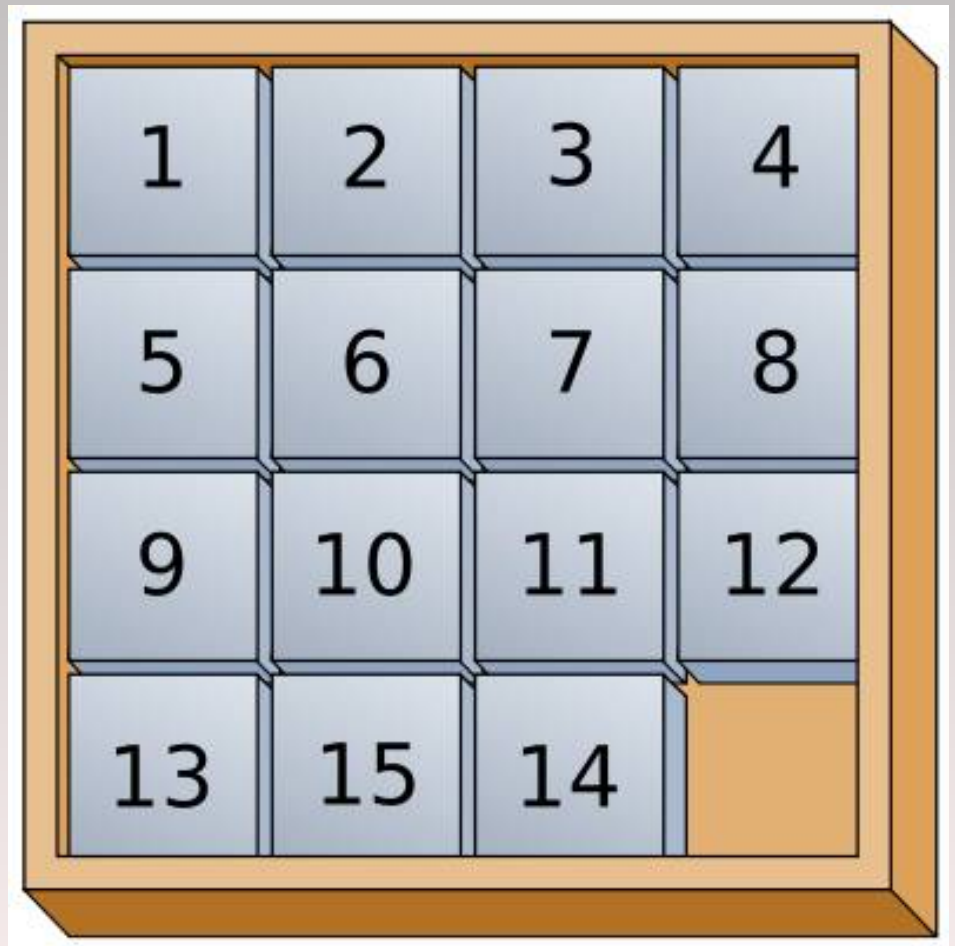


Sokoban @ work

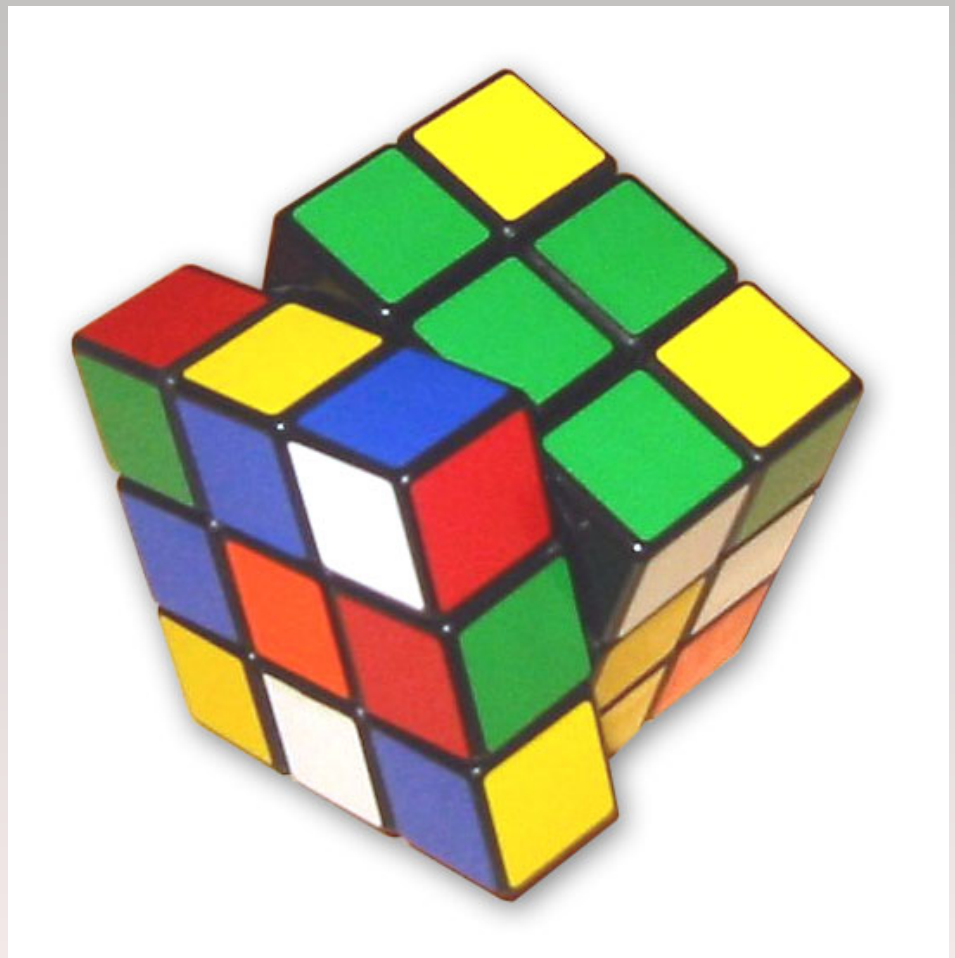


Peg solitaire

Sam Lloyd's Puzzle



Il cubo di Rubik



Le 12 monete

- Ci sono 12 monete
- Sappiamo che una di esse è falsa
- Quella falsa ha un peso diverso dalle altre 11 (non sappiamo se più pesante o più leggera)

Le 12 monete

- Ci sono 12 monete
- Sappiamo che una di esse è falsa
- Quella falsa ha un peso diverso dalle altre 11 (non sappiamo se più pesante o più leggera)
- Per scoprirlo disponiamo di una bilancia come questa:

Le 12 monete

- Ci sono 12 monete
- Sappiamo che una di esse è falsa
- Quella falsa ha un peso diverso dalle altre 11 (non sappiamo se più pesante o più leggera)
- Per scoprirlo disponiamo di una bilancia come questa:

- Come possiamo farlo in sole 3 pesate?

Sudoku

		1						
		2		3				4
			5			6		7
5			1	4				
	7						2	
				7	8			9
8		7			9			
4				6		3		
						5		

Questo ve lo svelo

		1						
		2		3				4
			5			6		7
5			1	4				
	7						2	
				7	8			9
8		7			9			
4				6		3		
						5		

$\text{domain}(X_{1,1}, \ldots, X_{9,9}) = [1..9]$

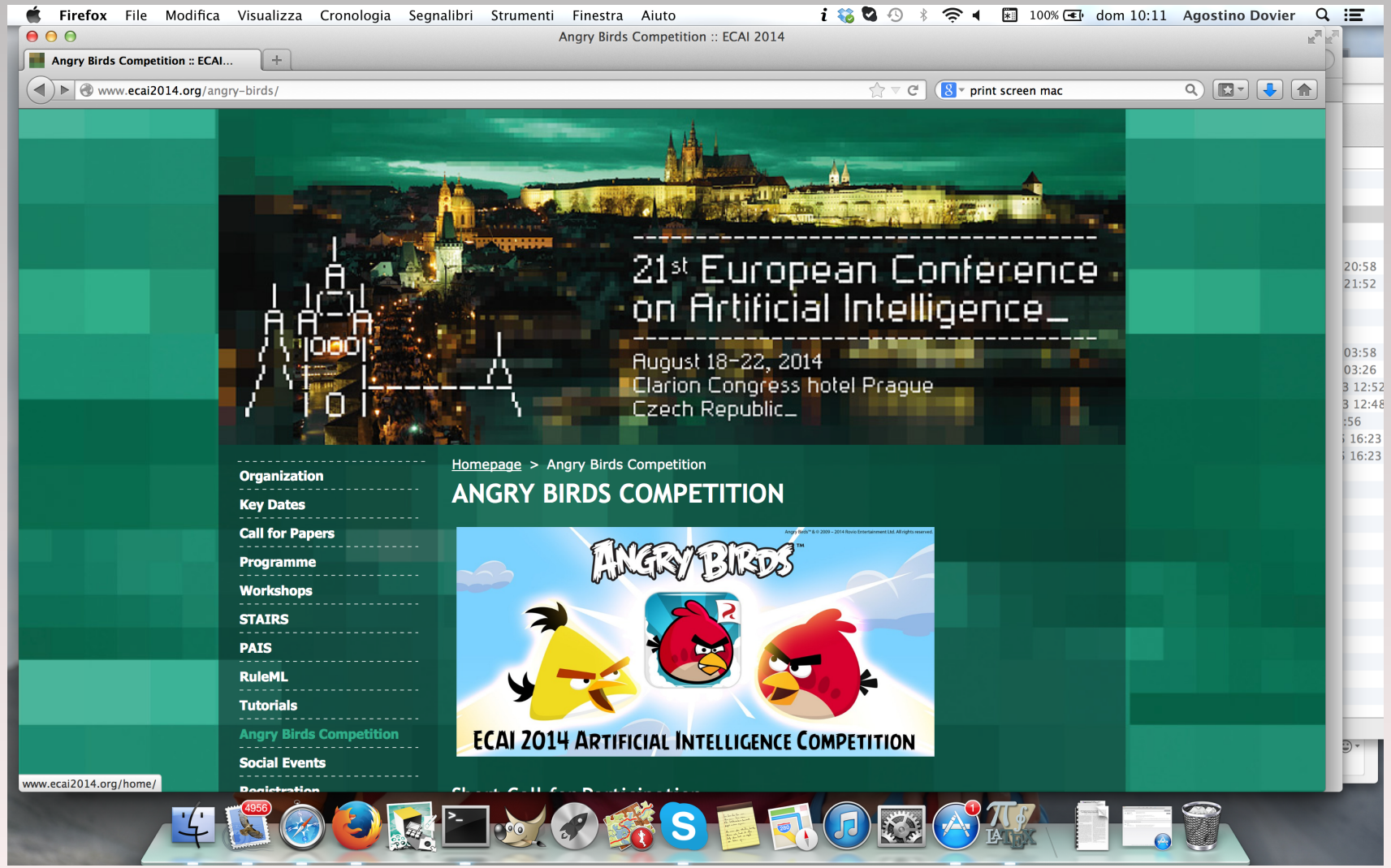
$X_{1,3} = 1, X_{2,3} = 2, X_{2,5} = 3, X_{2,9} = 4, \ldots, X_{9,7} = 5$

$\text{alldifferent}(X_{1,1}, \ldots, X_{1,9}) \ldots \text{alldifferent}(X_{9,1}, \ldots, X_{9,9})$

$\text{alldifferent}(X_{1,1}, \ldots, X_{9,1}) \ldots \text{alldifferent}(X_{1,9}, \ldots, X_{9,9})$

$\text{alldifferent}(X_{1,1}, \ldots, X_{3,3}) \ldots \text{alldifferent}(X_{7,7}, \ldots, X_{9,9})$

Angry Birds — ECAI 2014



Conclusioni

Spero di avervi incuriosito circa la ricerca in (un'area della) Intelligenza Artificiale

Incorporare tecniche intelligenti di ricerca delle soluzioni nei linguaggi di programmazione permette ricadute applicative molto vaste.

Ci si diverte a testarle sui giochi

Come si inizia a lavorare in AI? Per esempio iscrivendosi al corso di laurea in Informatica dell'Università di Udine (attivo dal 1977)

Ma (chi lo vorrà) potrà farlo già in questa attività di *Laborarorio PLS*
PLS = Piano Lauree Scientifiche
`www.dimi.uniud.it/scuole/pls/`

Conclusioni

Vedrete come codificare in Answer Set Programming, un dialetto del Logic Programming (in italiano Programmazione Logica)

`www.programmazionelogica.it`
`http://logicprogramming.org/`

Come Software vi basta un editor di testi (qualunque: ne avete già almeno uno nel vostro PC) per creare un file `nomefile.lp`

E il tool libero CLINGO che si scarica da

`http://potassco.sourceforge.net/`

Poi si esegue da linea di comando

`clingo <A> nomefile.lp <B>`

Ove al posto di `<A>` ci possono essere degli assegnamenti di valori (es `-c n=5`) e al posto di `<B>` il numero di soluzioni che cerchiamo (0 vuol dire tutte).

Per ogni domanda contattatemi pure via email (cercate dovier con un motore di ricerca)