ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Introduzione ai Constraint Satisfaction Problems

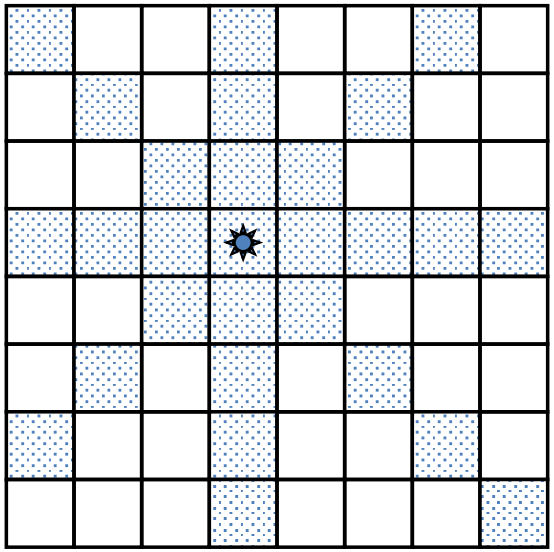
Paola Mello - Federico Chesani

DISI – Università di Bologna

Constraint Satisfaction Problems
Problemi di soddisfacimento di vincoli

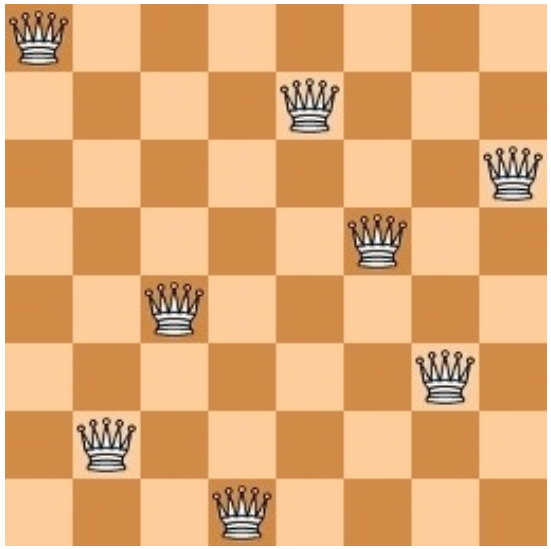
- Molti problemi di AI possono essere visti come problemi di soddisfacimento di vincoli.
- Obiettivo: trovare uno stato del problema che soddisfi un dato insieme di vincoli.
- Si basa sul concetto di **variabili** e **domini** di valori che esse possono assumere, e di **vincoli** tra i valori che le variabili assumono

- **Esempio: Il Problema delle Otto Regine**
  - È data una scacchiera (8x8): il problema consiste nel posizionarvi otto regine in modo da evitare un attacco reciproco.
  - Le mosse possibili per la regina prevedono tutte le posizioni sulla stessa riga, colonna e diagonale a partire dalla casella.



Romicapo delle 8 (N) regine

- E' un problema matematico ispirato al gioco degli scacchi (1884).
- Alla soluzione del problema si dedicò anche il noto matematico C. F. Gauss che trovò 72 diverse soluzioni. Le soluzioni possibili sono in realtà 92.
- Il problema può essere generalizzato a una scacchiera di *N* caselle di lato sulla quale si debbano collocare un pari numero di regine.
- È stato dimostrato matematicamente che per ogni *N* maggiore di 3 esistono un certo numero positivo di soluzioni; questo numero varia in base al variare di *N*.



PROBLEMA DELLE 8 REGINE: Un modello (1)

- Le posizioni della scacchiera sono rappresentate da NXN variabili (8X8). Molto numerose...

- L'istanziazione di una variabile **N** al valore **1** indica che e' posizionata una regina, se il valore e' **0** la posizione e' libera. Dominio di possibili valori: [1,0].

- I vincoli sono che non possono esserci due 1 contemporaneamente sulla verticale, orizzontale o diagonale.

PROBLEMA DELLE 8 REGINE: Un altro modello (2)

- Le otto regine vengono rappresentate con le variabili

$$X_1, X_2, \ldots, X_8$$

- Il pedice si riferisce alla colonna occupata dalla corrispondente regina.

- **Istanziare** una variabile significa assegnarle un **valore**
- L'istanziazione di una variabile X_i a un valore **k** indica che la regina corrispondente viene posizionata sulla **k**-esima riga della i-esima colonna.
- Se k rappresenta la riga della scacchiera → k assumerà come valori l'intervallo [1..8]. Formalmente:

$$X_i :: [1..8]$$

- Le variabili hanno come insieme di possibili valori gli interi compresi tra 1 e 8 che corrispondono alle righe occupate.

Adottiamo il modello (2)...
VINCOLI: DUE TIPI

a) Quelli che vincolano i valori delle variabili al dominio suddetto
b) Quelli che devono impedire un attacco reciproco e che impongono, quindi, relazioni tra le variabili o, meglio detto, **relazioni tra i valori assunti dalle variabili**.

$\mathbf{1 \leq X_i \leq 8}$	per $\mathbf{1 \leq i \leq 8}$
$\mathbf{X_i \neq X_j}$	per $\mathbf{1 \leq i < j \leq 8}$
$\mathbf{X_i \neq X_j + (j-i)}$	per $\mathbf{1 \leq i < j \leq 8}$
$\mathbf{X_i \neq X_j - (j-i)}$	per $\mathbf{1 \leq i < j \leq 8}$

Il primo vincolo impone che i valori assunti dalle variabili del problema siano compresi tra i numeri interi 1 e 8: vincoli unari, perchè coinvolgono una sola variabile

I tre vincoli successivi definiscono relazioni tra le variabili e, in particolare, tra due variabili alla volta: vincoli binari, perchè coinvolgono due variabili

Due tipi di vincoli (continua)

1) $1 \leq X_i \leq 8$ per $1 \leq i \leq 8$
2) $X_i \neq X_j$ per $1 \leq i < j \leq 8$
3) $X_i \neq X_j + (j-i)$ per $1 \leq i < j \leq 8$
4) $X_i \neq X_j - (j-i)$ per $1 \leq i < j \leq 8$

- Il secondo vincolo impone che due regine non siano posizionate sulla stessa riga. In caso contrario si attaccherebbero.

- Il terzo e il quarto vincolo riguardano le posizioni sulle due diagonali a partire dalla casella iniziale: impongono che il posizionamento di due regine sulla medesima diagonale sia scartato come soluzione non ammissibile.

Altro esempio: CRIPTOARITMETICA

```
  S  E  N  D  +
  M  O  R  E  =
---------------
M  O  N  E  Y
```

Sostituire tutte le stesse lettere con una cifra che non compare già nel rompicapo; tipicamente a lettera diversa corrisponde cifra diversa.

Obiettivo: il rompicapo rappresenta una somma corretta.

Quali variabili?
Quali domini?
Quali vincoli?

Altro esempio: SUDOKU

- E' data una griglia di 9x9 caselle
- Alcune caselle sono già fissate, le altre vanno riempite con numeri dall'1 al 9
- La tavola è suddivisa in 9 quadranti di 3x3 caselle
- Su ogni quadrante devono essere messi tutti e 9 i numeri, senza ripetizioni
- Ogni riga orizzontale e ogni riga verticale dell'intera tavola non deve contenere ripetizioni di numeri

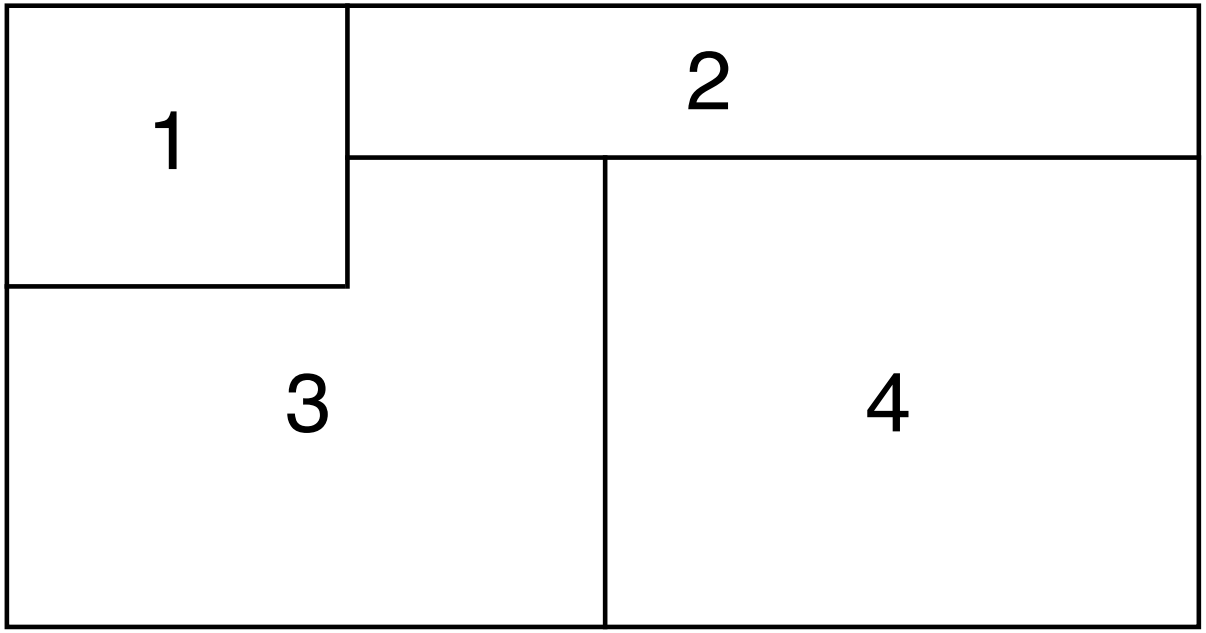
		9				7		
	4		5		9		1	
3				1				2
	1			6			7	
		2	7		1	8		
	5			4			3	
7				3				4
	8		2		4		6	
		6				5		

Quali variabili?
Quali domini?
Quali vincoli?

Altro esempio: MAP COLORING

Supponiamo di dover colorare delle porzioni di un piano, in modo tale che due regioni contigue siano colorate da colori diversi. Supponiamo anche di aver a disposizione i colori: red (**r**), green (**g**) e blu (**b**)

Per non confonderci, numeriamo le regioni...



Altro esempio: MAP COLORING

- Quattro colori sono sufficienti per ogni mappa (dimostrato solo nel 1976).
- Variante facile del problema del Graph Coloring– al massimo quatro regioni sono connesse con tutte.

Quali variabili? le regioni...

Quali domini? i colori permessi

Quali vincoli? regioni adiacenti devono avere colori diversi.

CSP – formalmente...

Un CSP (Constraints Satisfaction Problem) può essere definito come:

1) Un insieme finito di **variabili** $(X_1, X_2, \ldots, X_n)$
2) i cui valori appartengono a **domini** finiti di definizione $(D_1, D_2, \ldots, D_n)$
3) e su un insieme di **vincoli**.

Cosa è un **vincolo**?

- Un vincolo $c(X_{i1}, X_{i2}, \ldots X_{ik})$ tra k variabili è un sottoinsieme del prodotto cartesiano $D_{i1} \times D_{i2} \times \ldots \times D_{ik}$ che specifica quali valori delle variabili sono compatibili con le altre.
- Tale sottoinsieme non deve essere definito esplicitamente ma è rappresentato in termini di relazioni.

Cosa è una **soluzione**? Una soluzione ad un CSP prevede un assegnamento di tutte le variabili, tale che soddisfi tutti i vincoli.

CSP – Albero Decisionale

- Un possibile **albero decisionale** per un CSP si ottiene (dopo aver stabilito un ordinamento per le variabili) facendo corrispondere ad ogni livello dell'albero l'assegnamento di una variabile e ad ogni nodo la scelta di un possibile valore da dare alla variabile corrispondente al livello del nodo stesso.

- Ogni foglia dell'albero rappresenterà quindi un assegnamento di valori a tutte le variabili. Se tale assegnamento soddisfa tutti i vincoli, allora la foglia corrispondente rappresenterà una soluzione del problema, altrimenti rappresenterà un fallimento.

- La **ricerca di una soluzione** è equivalente all'esplorazione dell'albero decisionale per trovare una foglia-soluzione.
- In un problema di n variabili ed in cui i domini hanno tutti la stessa cardinalità d, il numero di foglie di un albero decisionale così costruito è pari a d^n.

Albero decisionale – Esempio

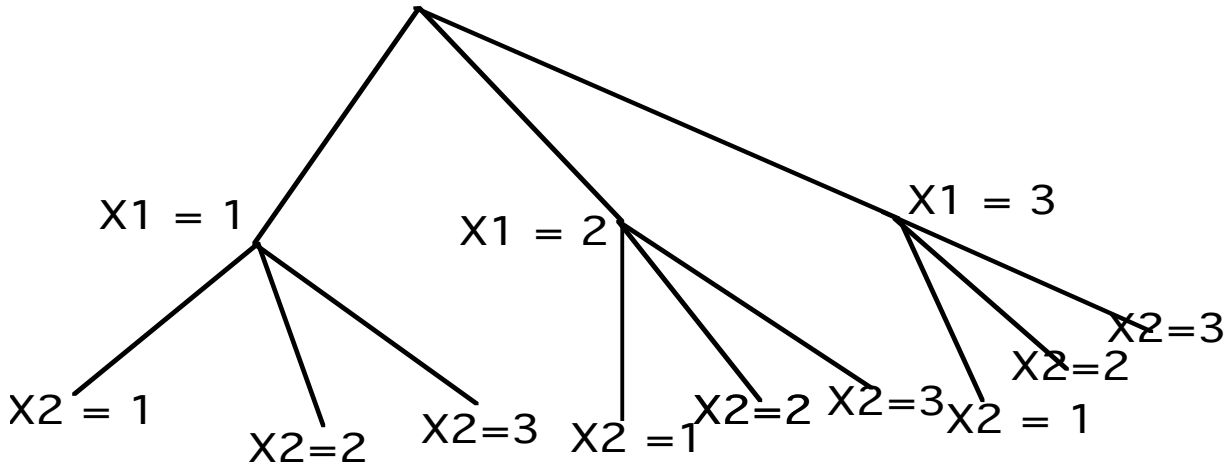
In un albero che rappresenta un problema di 10 variabili ed in cui ogni dominio ha cardinalità 10 esistono 10 miliardi di foglie.

È quindi evidente che la **strategia di esplorazione** dell'albero risulta di importanza fondamentale al fine di trovare una soluzione per un problema complesso in tempi ragionevolmente brevi (tecniche di consistenza).

Esempio:

- due variabili (X1 e X2)
- domini: X1, X2 :: [1..3]

Numero di foglie: 3^2



Diversi possibili approcci

- Algoritmi **Generativi** – uso dei vincoli a posteriori
  - Generate and Test
  - Standard Backtracking
- Algoritmi di **Propagazione** – uso dei vincoli a priori
  - Forward Checking
  - Partial Look Ahead
  - Full Look Ahead
- Algoritmi basati su **Tecniche di Consistenza** – uso dei vincoli a priori
  - Arc-Consistency

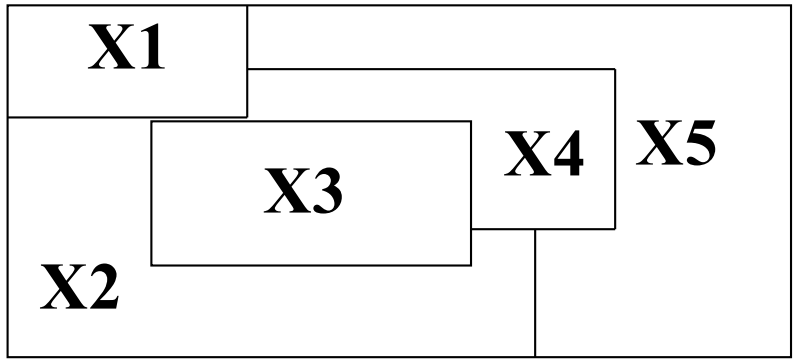
Generate and Test

Si sviluppa e visita un albero decisionale percorrendolo in profondità, assegnando valori alle variabili senza preoccuparsi di verificare la consistenza con gli altri vincoli (**fase di generate**). Solo giunti ad una foglia (assegnamento a tutte le variabili), si verifica se i vincoli sono soddisfatti (**fase di test**).

Esempio, istanza del problema di Map Coloring:

Vincoli di dominio: X1, X2, X3, X4, X5:: [rosso, giallo, verde, blu]

Vincoli topologici:
X1 ≠ X2 , X1 ≠ X3, X1 ≠ X4, X1 ≠ X5,
X2 ≠ X3, X2 ≠ X4, X2 ≠ X5,
X3 ≠ X4,
X4 ≠ X5



Albero decisionale con 4^5 foglie

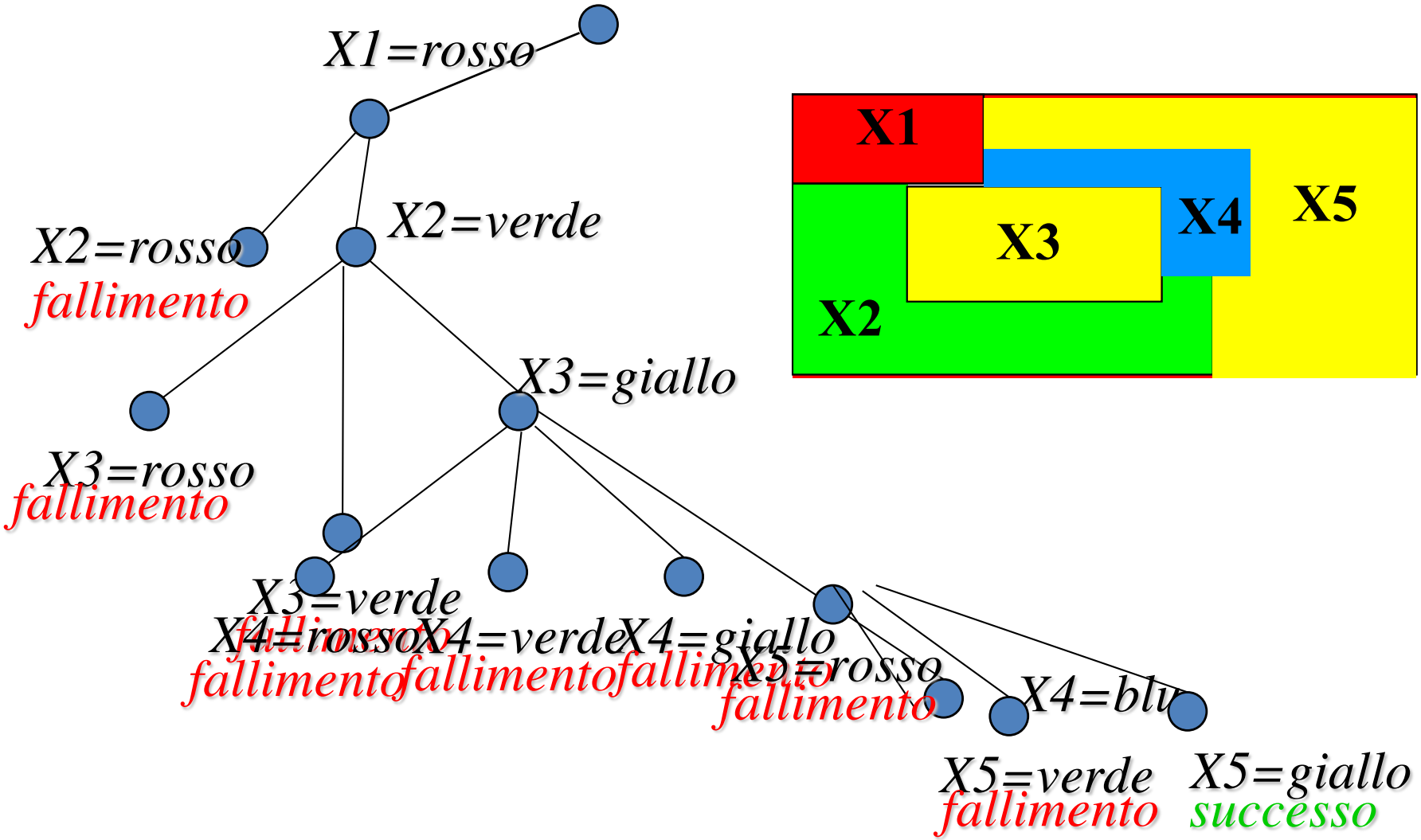
Standard Backtracking

Sebbene migliore della precedente anche questa tecnica prevede un utilizzo a posteriori dei vincoli

- A ogni istanziazione di una variabile si verifica la coerenza della variabile appena istanziata con quelle assegnate precedentemente.

Quindi l'utilizzo dei vincoli è più efficace del precedente perché non si prosegue la ricerca in rami che, ai primi livelli dell'albero, presentano già delle contraddizioni.

Standard Backtracking – Esempio Map Coloring



Standard Backtracking e Limiti dell'uso a posteriori dei vincoli

- A differenza della metodologia del Generate and Test, lo Standard Backtracking controlla la consistenza dei vincoli per ogni variabile istanziata.

- I vincoli sono utilizzati all'indietro (*backward*) e portano a una effettiva riduzione dello spazio di ricerca.
- Tuttavia questa riduzione viene fatta a posteriori (a *posteriori-pruning*) cioè dopo aver effettuato il tentativo. Questo è un difetto da attribuire a tutti i metodi che utilizzano i vincoli passivamente, cioè posteriormente ad un tentativo di istanziazione.

- Idea: utilizzare anche i vincoli che coinvolgono variabili ancora libere.

Uso dei vincoli a priori

- Utilizzo attivo dei vincoli nella guida della computazione e nel cosiddetto *pruning* a priori dell'albero decisionale associando, a ciascuna variabile, l'insieme di valori ammissibili, **PRIMA** o **DURANTE** la computazione.
- Questi insiemi (domini) vengono perciò **ridotti**, permettendo di scegliere per le variabili ancora libere valori ammissibili con le variabili già istanziate senza più considerare i vincoli che le legano.

Due approcci:

- **Algoritmi di Propagazione**
    - Basati sulla propagazione dei vincoli per eliminare a priori, durante la ricerca, porzioni dell'albero decisionale che porterebbero ad un sicuro fallimento (compatibilmente con le scelte già effettuate).
- **Tecniche di Consistenza**
    - Basati sulla propagazione dei vincoli per derivare un problema più semplice di quello (completo) originale.

Tipicamente, prima si applicano Tecniche di Consistenza e poi di Propagazione, oppure sono integrate durante la ricerca.

Algoritmi di Propagazione

Gli algoritmi di propagazione sono metodi di ricerca "più intelligenti" che tentano di prevenire i fallimenti anziché recuperare fallimenti già avvenuti.

- *Pruning a priori* dell'albero delle decisioni.
- Utilizzo delle relazioni tra le variabili del problema, i vincoli, per ridurre lo spazio di ricerca prima di arrivare al fallimento: la propagazione ha l'effetto di ridurre eventualmente i dominii delle variabili future (se un dominio risulta vuoto, fallimento)
- Come conseguenza, si eliminano rami dell'albero che porterebbero ad un sicuro insuccesso limitando inutili backtracking.

- **Forward Checking** (FC)
- **Partial Look Ahead** (PLA)
- **Full Look Ahead** (FLA)

Come lo SB, in più controllano i vincoli tra variabili già legate e variabili future (ancora da istanziare), e anche – in gradi diversi – tra coppie di variabili future.

Forward Checking

- Viene utilizzata, dopo ogni assegnamento, la **propagazione dei vincoli** che consiste nell'eliminazione dei valori incompatibili con quello appena istanziato dai domini delle variabili non ancora istanziate.
- Questo metodo si rivela molto efficace soprattutto quando le ultime variabili ancora libere sono associate ad un insieme di valori ammissibili ridotto e perciò risultano molto vincolate e facilmente assegnabili.
  - Se il dominio associato ad una variabile libera presenta un solo valore l'assegnamento può essere effettuato senza sforzo computazionale.
  - Se ad un certo punto della computazione ci si accorge che un dominio associato ad una variabile risulta vuoto il meccanismo del Forward Checking fallisce senza proseguire in tentativi e backtracking.
- L'assegnazione di un valore ad una variabile ha ripercussioni sull'insieme dei valori disponibili per le variabili ancora libere. In questo modo i vincoli agiscono in avanti (***forward***) e limitano lo spazio delle soluzioni prima che vengano effettuati tentativi su di esso.

Forward Checking – Esempio del map coloring

Variabili:

X1 X2 X3 X4

Supponiamo che X2 sia l'ultima variabile oggetto di *labeling* (assegnamento di un valore)

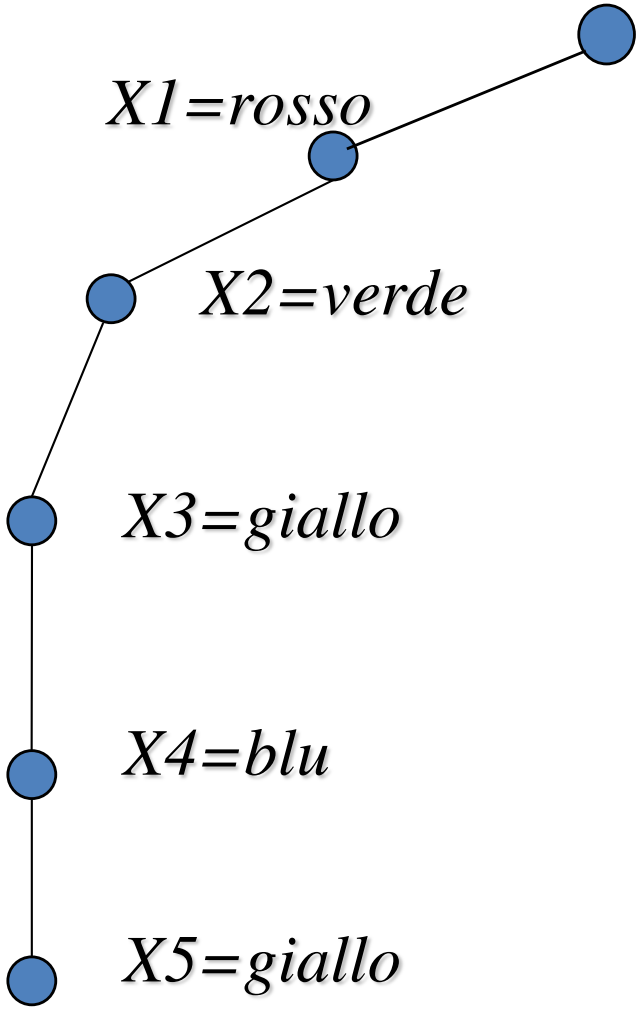
- Standard Backtracking: controlla c(X2,X1)

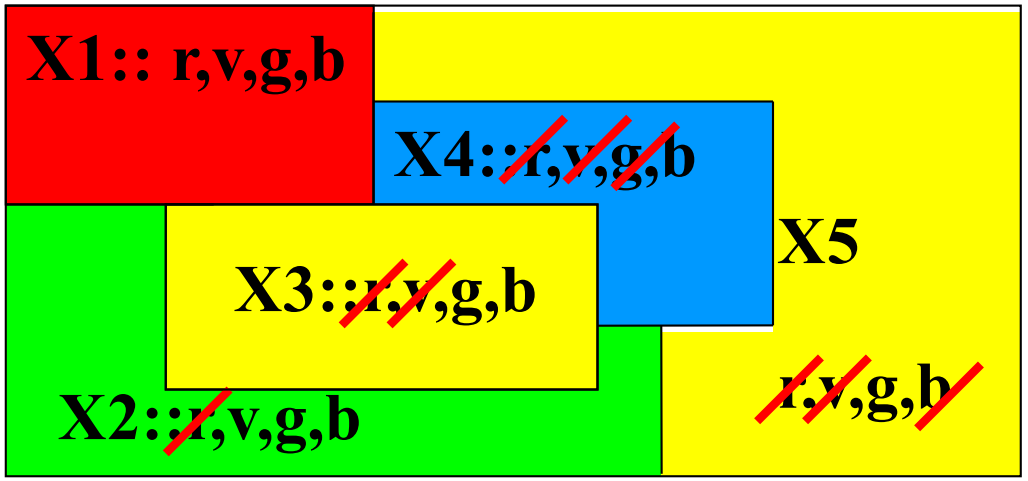
- Forward Checking:
  - controlla c(X2,X3) per ogni valore in D3
  - e c(X2,X4) per ogni valore in D4

Sono eliminati i valori da D3 e D4 incompatibili con l'assegnamento di X2

Se il dominio di una variabile libera diventa vuoto, *fallimento*

Forward Checking – Esempio del map coloring





(Partial and Full) Look Ahead

- La tecnica più completa per quel che riguarda il *pruning* a priori dell'albero decisionale.
- Ad ogni istanziazione viene controllata, come per il Forward Checking, la compatibilità dei vincoli contenenti la variabile appena assegnata con le precedenti (istanziate) e le successive (libere).
- In più viene sviluppato il ***look ahead*** **(sguardo in avanti)** che controlla l'esistenza, nei domini associati alle variabili ancora libere, di valori compatibili con i vincoli contenenti solo variabili non istanziate.
- I domini associati a ogni variabile vengono ridotti propagando anche le relazioni contenenti coppie di variabili non istanziate. Viene verificata quindi la possibilità di una futura assegnazione consistente fra (coppie del)le variabili libere.

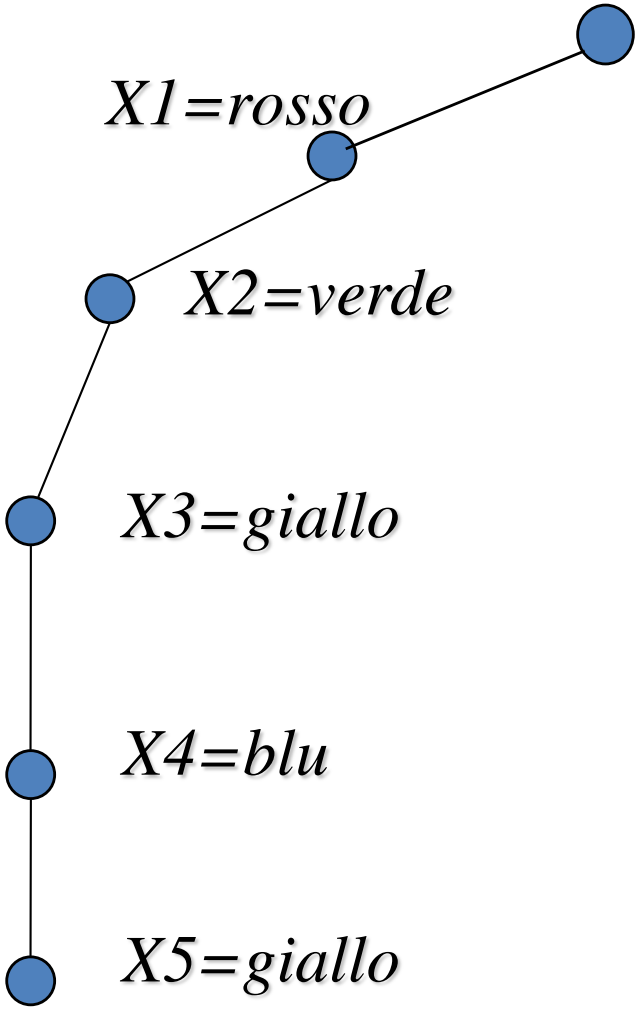
Partial and Full Look Ahead

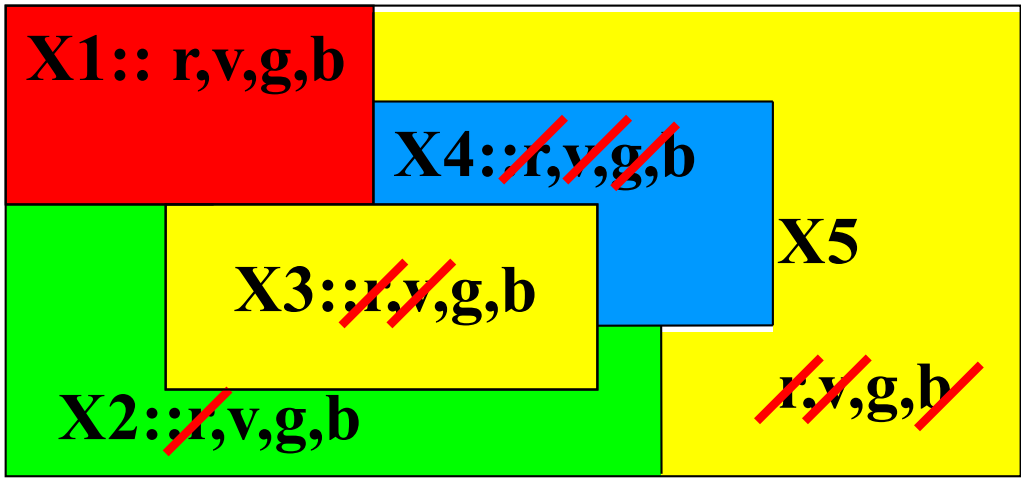
- **Partial Look Ahead**: si ha una propagazione dei vincoli contenenti la variabile Xh, non ancora istanziata e le variabili "future", ossia le variabili Xh+1,..., Xn
  - Per ogni variabile non ancora assegnata Xk+1,.....,Xn, deve esistere un valore per il quale sia possibile trovare, per tutte le altre variabili "successive" non ancora assegnate, almeno un valore compatibile con esso.

- **Full Look Ahead**: se Vk è il valore appena assegnato alla variabile Xk, si ha una propagazione dei vincoli contenenti la variabile Xh, non ancora istanziata, e tutte le variabili non ancora assegnate, ossia le variabili Xk+1,...Xh-1,Xh+1..,Xn.
  - per ogni variabile non ancora assegnata Xk+1,....,Xn deve esistere un valore per il quale sia possibile trovare, per tutte le variabili non ancora assegnate, almeno un valore compatibile con esso.

Esempio: FC vs. PLA in Map Coloring

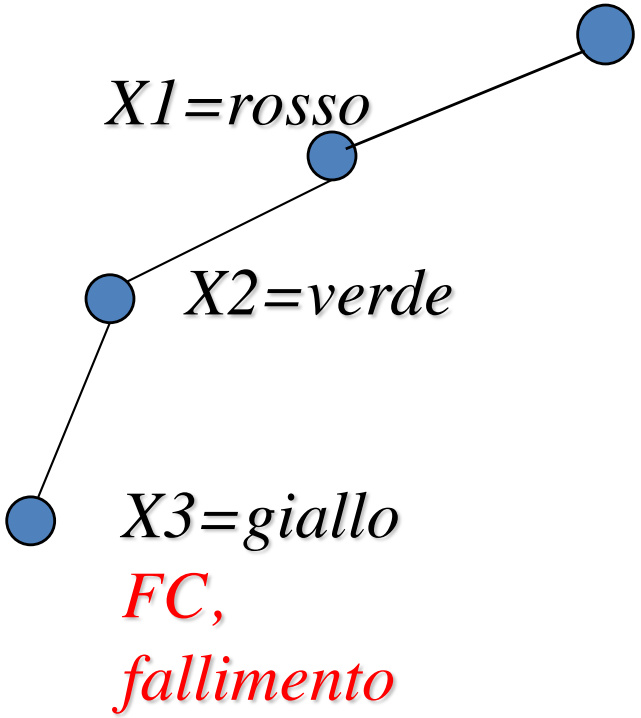
PLA rispetto a FC non aumenta il pruning dell' albero in questo caso

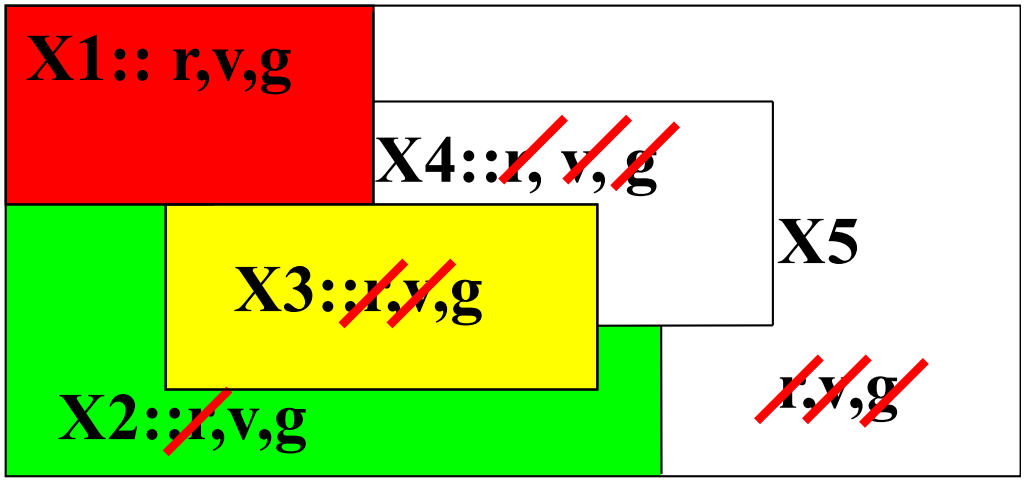




Esempio: FC vs. PLA in Map Coloring

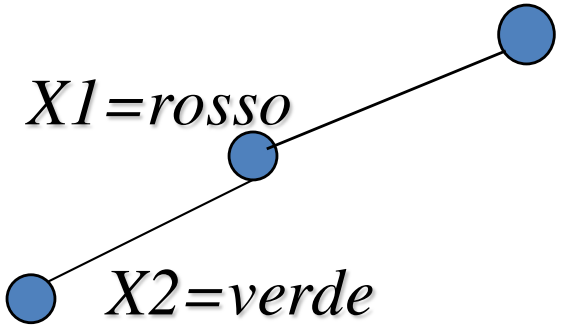
Ma per una istanza con tre soli valori di dominio (r,v,g), il FC:



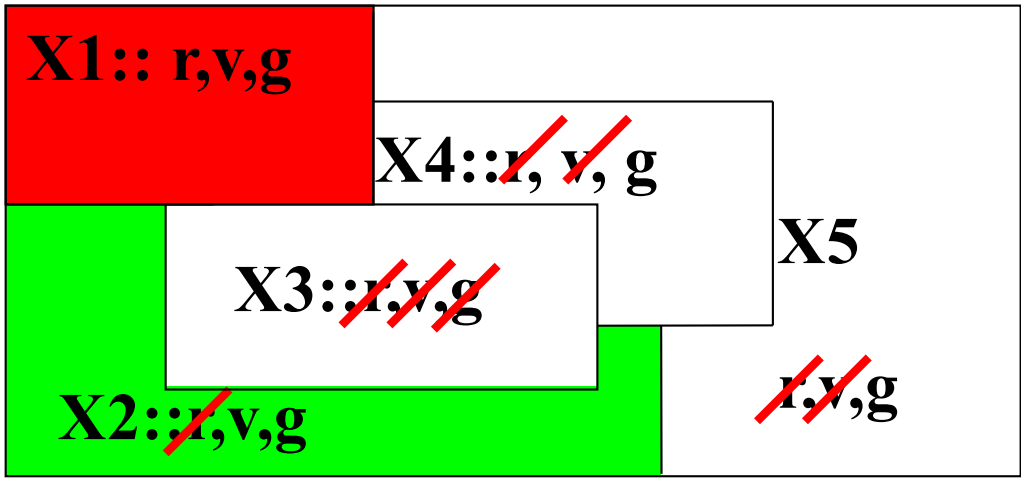


Esempio: FC vs. PLA in Map Coloring

Il PLA anticipa il fallimento (di un livello in questo caso):



PLA, fallimento



Il dominio di X3 diventa vuoto

(non esiste alcun valore nel dominio di X4 compatibile con il valore g per X3; analogamente nel dominio di X5)

Tecniche di Consistenza

Differenza fondamentale:

- gli algoritmi di propagazione propagano i vincoli in seguito a istanziazioni delle variabili coinvolte nel problema;
- le tecniche di consistenza riducono il problema originale eliminando dai domini delle variabili i valori che non possono comparire in una soluzione finale.

Possono essere applicate staticamente oppure ad ogni passo di assegnamento (labeling) come potenti tecniche di propagazione per le variabili non ancora istanziate.

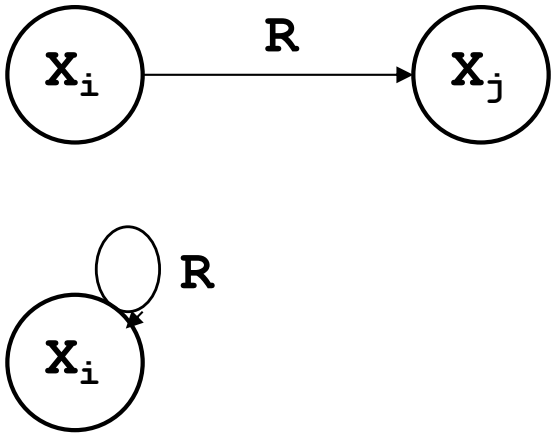
Tutte le tecniche di consistenza sono basate su una **rappresentazione del problema come una rete (grafo) di vincoli**.

- Gli archi possono essere orientati o non orientati: ad esempio il vincolo $>$ viene rappresentato da un arco orientato, mentre il vincolo $\neq$ da un arco semplice (non orientato o doppiamente orientato).

Constraint Graph

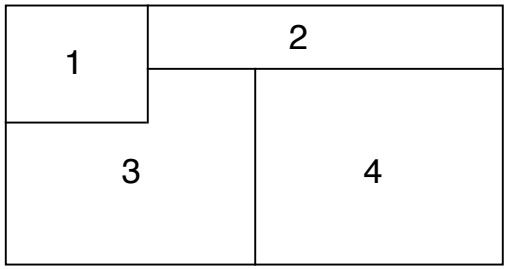
Per ogni CSP esiste un grafo (constraint graph) in cui i nodi rappresentano le variabili e gli archi i vincoli tra le variabili costituenti i nodi del grafo.

- I vincoli binari (R) collegano due nodi X_i e Xj:
- I vincoli unari sono rappresentati da archi che iniziano e terminano sullo stesso nodo X_i



Constraint Graph ed esempio Map Coloring

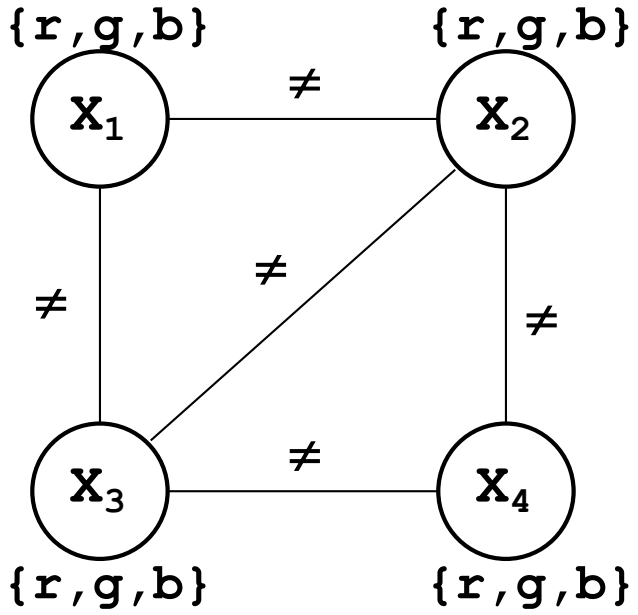
Supponiamo di dover colorare delle porzioni di un piano, caratterizzate da un numero, in modo tale che due regioni contigue siano colorate da colori diversi. Supponiamo anche di aver a disposizione i colori red (r), green (g) e blu (b).



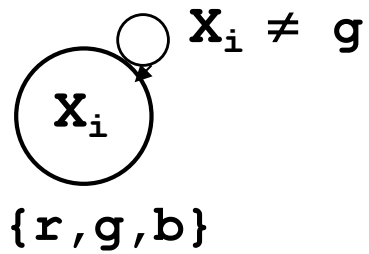
Il constraint-graph corrispondente è il seguente.

Tuttavia, esistono combinazioni di valori non compatibili tra loro (es: X1=r, X2=r, X3=r, X4=r).

Esistono diversi algoritmi che realizzano gradi diversi di consistenza.



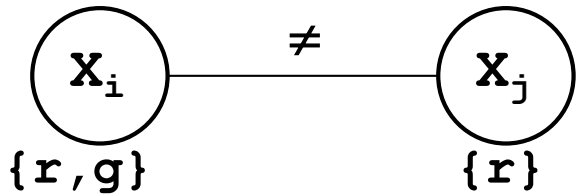
Node Consistency



- NODE-CONSISTENCY: consistenza di grado 1
  - Un nodo di un grafo di vincoli è consistente se per ogni valore $X_i \in D_i$ il vincolo unario su X_i è soddisfatto.

- Nell'esempio il nodo non e' node consistent perchè il valore $g \in D_i$ viola il vincolo unario P(i) su X_i.
- Per rendere il nodo consistente è necessario eliminare dal dominio di X_i il valore g.
- Un grafo è node consistente se tutti i suoi nodi sono consistenti.

Arc Consistency



La consistenza di grado 2 si ottiene partendo da un grafo node-consistente. Tale consistenza verifica se un arco A(i,j) è consistente.

ARC CONSISTENCY: un arco A(i,j) è consistente se per ogni valore $X \in D_i$ esiste almeno un valore $Y \in D_j$ tale che il vincolo tra i e j P(i,j) sia soddisfatto

L'arco in figura non è consistente perché, considerando il valore $r \in D_i$, non esiste un valore appartenente a D_j che soddisfi il vincolo P(i,j).

Per rendere consistente l'arco tra e X_j è necessario eliminare il valore r dal dominio di X_i : questo valore non comparirebbe in nessuna soluzione ammissibile.

CSP e ASIA-GiM – Take-home message

- Esistono molte tecniche per la risoluzione di un CSP
- Il difficile non è risolverlo (il computer ha potenza di calcolo)
- La parte DIFFICILE è rappresentare un problema in termini di un CSP

E' infatti necessario rispondere a tre domande:

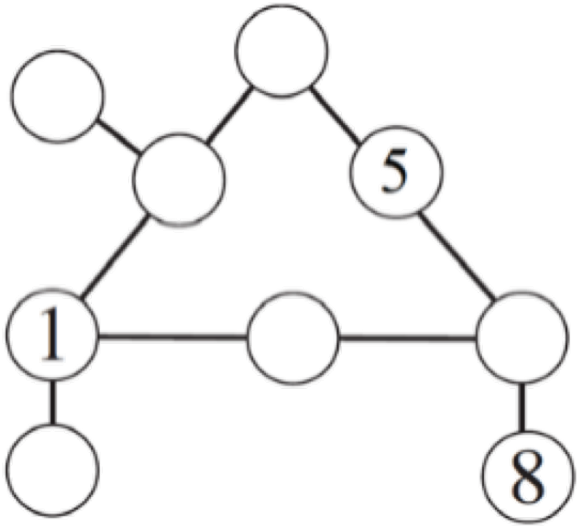
a) Quali e quante sono le variabili?
b) Quali sono i domini?
c) Quali e quanti sono i vincoli?

CSP e ASIA-GiM – Esempio 1

Giochi d'Autunno 2016 – Categoria CE –
Gioco 1

Dovete collocare tutte le cifre da 1 a 9 (incluse) nei cerchietti della figura (per aiutarvi 1, 5 e 8 sono state già scritte) in modo che le somme dei numeri collegati da un segmento siano sempre uguali a 10.

Quale numero in particolare avete scritto nel cerchietto in alto (in mezzo)?

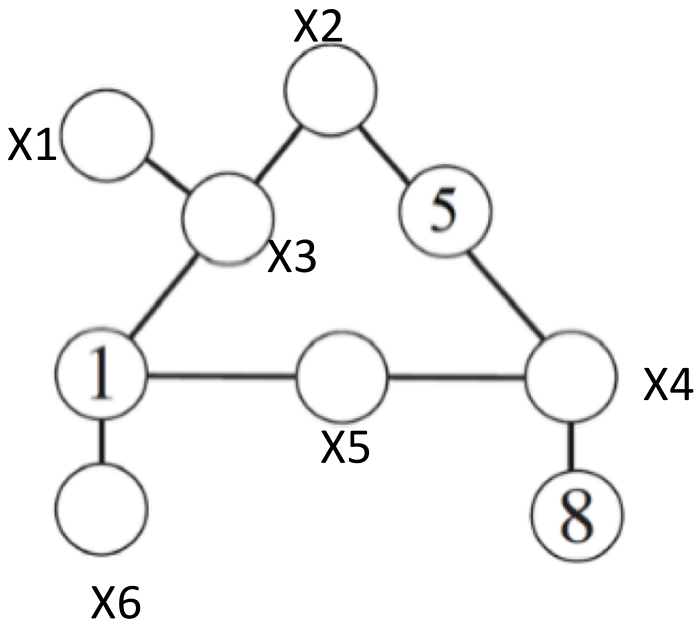


CSP e ASIA-GiM – Esempio 1 – Quali sono le variabili?

Giochi d'Autunno 2016 – Categoria CE – Gioco 1

Dovete collocare tutte le cifre da 1 a 9 (incluse) nei cerchietti della figura (per aiutarvi 1, 5 e 8 sono state già scritte) in modo che le somme dei numeri collegati da un segmento siano sempre uguali a 10.

Quale numero in particolare avete scritto nel cerchietto in alto (in mezzo)?



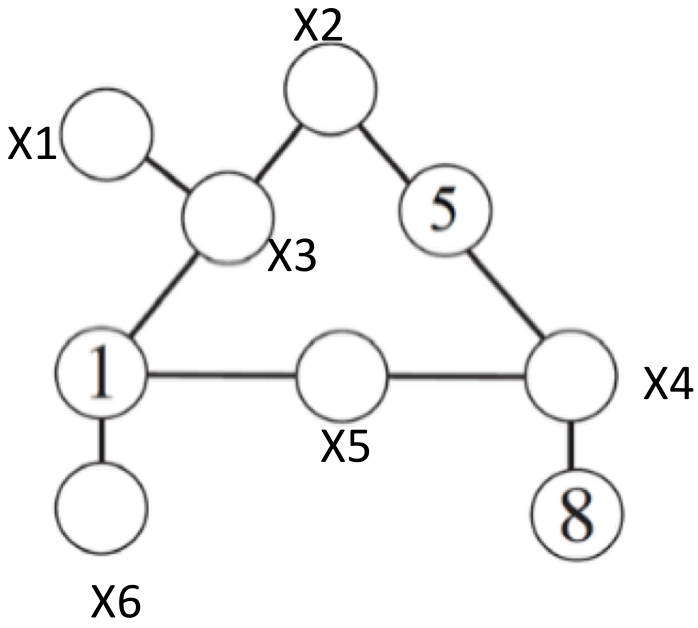
CSP e ASIA-GiM – Esempio 1 – Quali sono i domini?

Giochi d'Autunno 2016 – Categoria CE – Gioco 1

Dovete collocare tutte le cifre da 1 a 9 (incluse) nei cerchietti della figura (per aiutarvi 1, 5 e 8 sono state già scritte) in modo che le somme dei numeri collegati da un segmento siano sempre uguali a 10.

Quale numero in particolare avete scritto nel cerchietto in alto (in mezzo)?

X1, X2, ..., X6 :: [1..9]

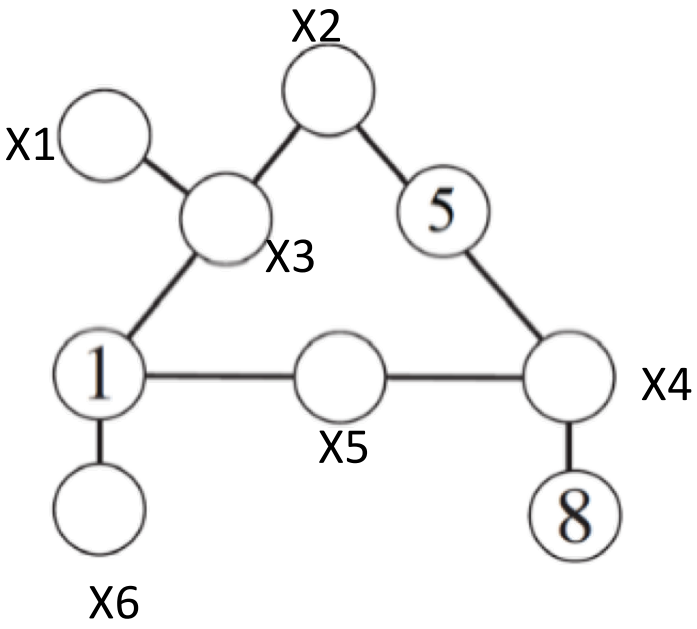


CSP e ASIA-GiM – Esempio 1 – Quali sono i vincoli?

Giochi d'Autunno 2016 – Categoria CE – Gioco 1

Dovete collocare tutte le cifre da 1 a 9 (incluse) nei cerchietti della figura (per aiutarvi 1, 5 e 8 sono state già scritte) in modo che le somme dei numeri collegati da un segmento siano sempre uguali a 10.

Quale numero in particolare avete scritto nel cerchietto in alto (in mezzo)?



X1, X2, ..., X6 :: [1..9]

all_different(X1, X2, ..., X6)

X1, X2, ..., X6 ≠ 1

X1, X2, ..., X6 ≠ 5

X1, X2, ..., X6 ≠ 8

X1+X3 = 10

X2+X3+1 = 10

X2+5+X4 = 10

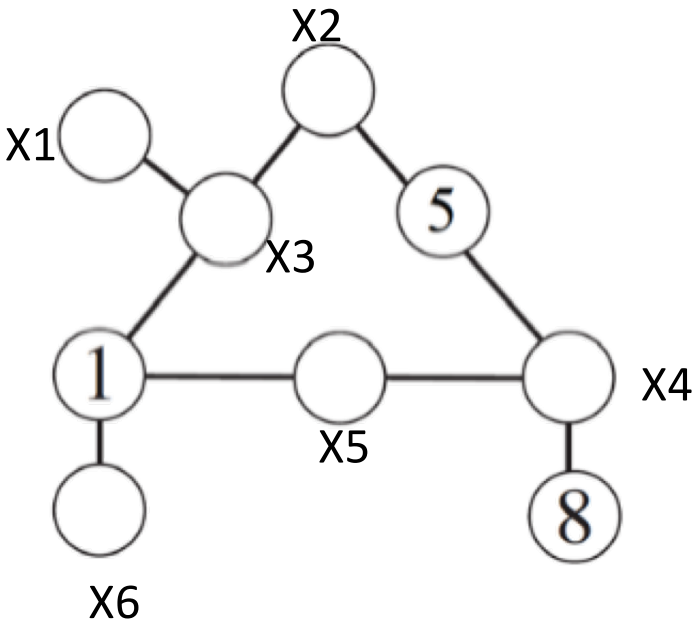
1 + X5 + X4 = 10

X4 + 8 = 10

1 + X6 = 10

CSP e ASIA-GiM – Esempio 1
Ma poi, fatto il modello?

Giochi d'Autunno 2016 – Categoria CE – Gioco 1



Dovete collocare tutte le cifre da 1 a 9 (incluse) nei cerchietti della figura (per aiutarvi 1, 5 e 8 sono state già scritte) in modo che le somme dei numeri collegati da un segmento siano sempre uguali a 10.

Quale numero in particolare avete scritto nel cerchietto in alto (in mezzo)?

X1, X2, ..., X6 :: [1..9]

all_different(X1, X2, ..., X6)

X1, X2, ..., X6 ≠ 1

X1, X2, ..., X6 ≠ 5

X1, X2, ..., X6 ≠ 8

X1 + X3 = 10

X2 + X3+1 = 10

X2 + 5 + X4 = 10

1 + X5 + X4 = 10

X4 + 8 = 10

1 + X6 = 10

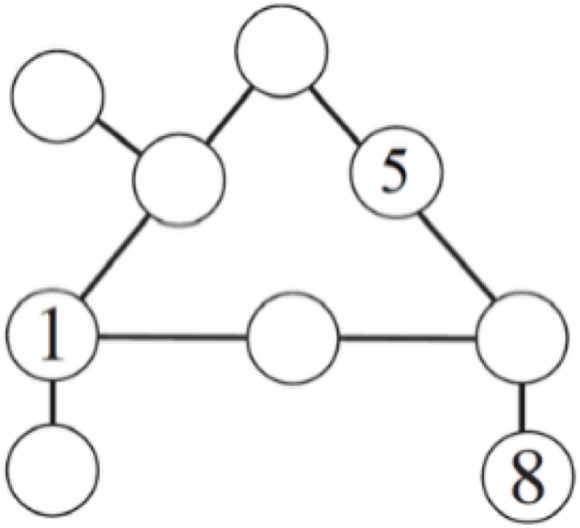
```
:- use_module(library(clpfd)).

test :-
[X1, X2, X3, X4, X5, X6] ins 1..9
, all_distinct([X1, X2, X3, X4, X5, X6])
, X1 #\= 1, X2 #\= 1, X3 #\= 1, X4 #\= 1, X5 #\= 1, X6 #\= 1
, X1 #\= 5, X2 #\= 5, X3 #\= 5, X4 #\= 5, X5 #\= 5, X6 #\= 5
, X1 #\= 8, X2 #\= 8, X3 #\= 8, X4 #\= 8, X5 #\= 8, X6 #\= 8
, X1 + X3 #= 10
, X2 + X3 + 1 #= 10
, X2 + 5 + X4 #= 10
, 1 + X5 + X4 #= 10
, X4 + 8 #= 10
, 1 + X6 #= 10
, label([X1, X2, X3, X4, X5, X6])
, write([X1, X2, X3, X4, X5, X6])
.
```

CSP e ASIA-GiM – Esempio 1 – Rivisto...

Giochi d'Autunno 2016 – Categoria CE –
Gioco 1

Dovete collocare tutte le cifre da 1 a 9 (incluse) nei cerchietti della figura (per aiutarvi 1, 5 e 8 sono state già scritte) in modo che le somme dei numeri collegati da un segmento siano sempre uguali a 10.

Quale numero in particolare avete scritto nel cerchietto in alto (in mezzo)?



Le variabili potrebbero essere le **posizioni** di ogni numero...

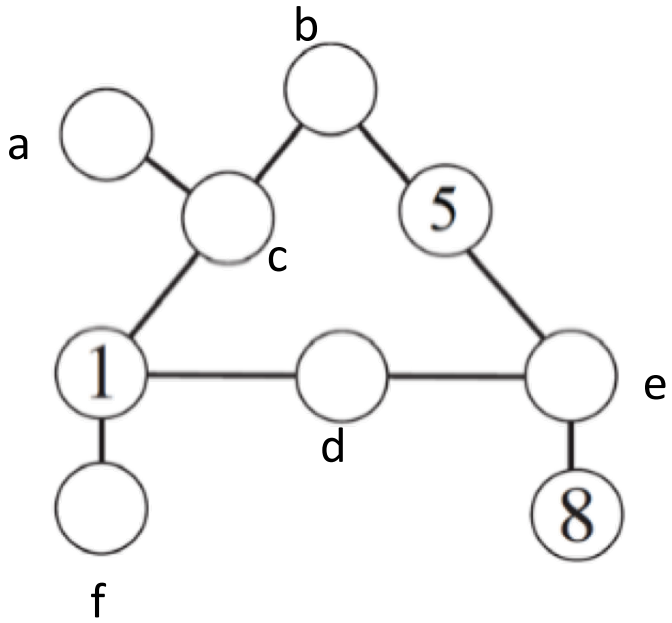
- una variabile X_i per ogni numero da 1 a 9
- denominiamo i cerchietti (posizioni) con una lettera
- si deve assegnare ad ogni numero una posizione

CSP e ASIA-GiM – Esempio 1 – Rivisto...
Quali sono i domini?

Giochi d'Autunno 2016 – Categoria CE – Gioco 1

Dovete collocare tutte le cifre da 1 a 9 (incluse) nei cerchietti della figura (per aiutarvi 1, 5 e 8 sono state già scritte) in modo che le somme dei numeri collegati da un segmento siano sempre uguali a 10.

Quale numero in particolare avete scritto nel cerchietto in alto (in mezzo)?



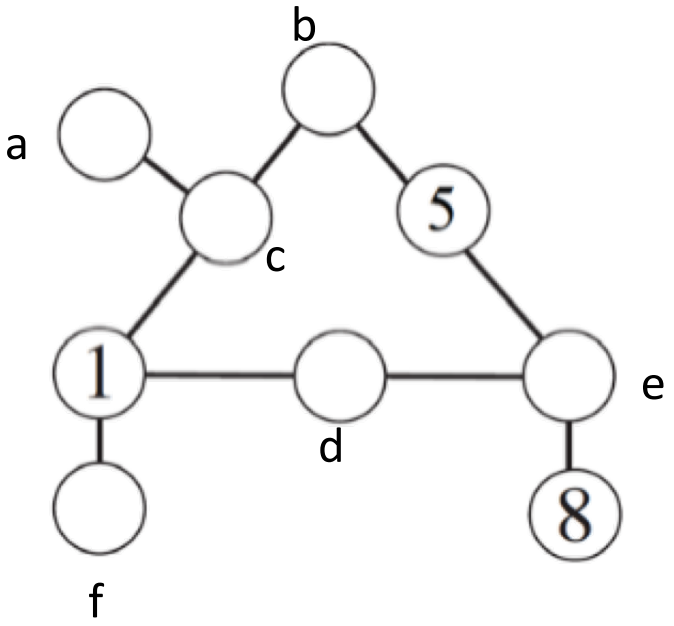
$X_2, X_3, X_4, X_6, X_7, X_9 :: [a, b, c, d, e, f]$

CSP e ASIA-GiM – Esempio 1 – Rivisto
Quali sono i vincoli?

Giochi d'Autunno 2016 – Categoria CE – Gioco 1

Dovete collocare tutte le cifre da 1 a 9 (incluse) nei cerchietti della figura (per aiutarvi 1, 5 e 8 sono state già scritte) in modo che le somme dei numeri collegati da un segmento siano sempre uguali a 10.

Quale numero in particolare avete scritto nel cerchietto in alto (in mezzo)?



$X_2, X_3, X_4, X_6, X_7, X_9 :: [a, b, c, d, e, f]$

Vincoli molto più difficili da esprimere... ma fattibile...

all_different(...)

$\forall\ (Xi==a \wedge Xj==c) \longrightarrow i + j = 10$

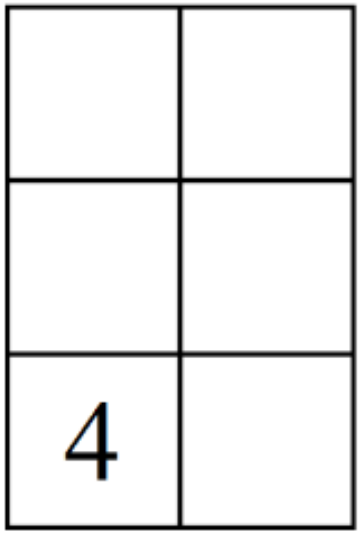
...

CSP e ASIA-GiM – Esempio 2

Giochi d'Autunno 2016 – Categoria CE –
Gioco 4

Dovete scrivere nelle tre righe della griglia tre numeri, ciascuno composto da due cifre (per aiutarvi, il 4 è stato già scritto). Le sei cifre devono essere tutte diverse e tali che il numero che si leggerà nella seconda riga risulti il doppio di quello della prima e che il numero della terza riga risulti il triplo di quello della prima.

Quale numero in particolare avete scritto nella seconda riga?



CSP e ASIA-GiM – Esempio 2 – Quali sono le variabili?

Giochi d'Autunno 2016 – Categoria CE –
Gioco 4

Dovete scrivere nelle tre righe della griglia tre numeri, ciascuno composto da due cifre (per aiutarvi, il 4 è stato già scritto). Le sei cifre devono essere tutte diverse e tali che il numero che si leggerà nella seconda riga risulti il doppio di quello della prima e che il numero della terza riga risulti il triplo di quello della prima.

Quale numero in particolare avete scritto nella seconda riga?

X1	X2
X3	X4
4	X5

CSP e ASIA-GiM – Esempio 2 – Quali sono i domini?

Giochi d'Autunno 2016 – Categoria CE – Gioco 4

Dovete scrivere nelle tre righe della griglia tre numeri, ciascuno composto da due cifre (per aiutarvi, il 4 è stato già scritto). Le sei cifre devono essere tutte diverse e tali che il numero che si leggerà nella seconda riga risulti il doppio di quello della prima e che il numero della terza riga risulti il triplo di quello della prima.

Quale numero in particolare avete scritto nella seconda riga?

X2, X4, X5 :: [0..9]

X1, X3 :: [1..9]

X1	X2
X3	X4
4	X5

CSP e ASIA-GiM – Esempio 2 – Quali sono i vincoli?

Giochi d'Autunno 2016 – Categoria CE – Gioco 4

Dovete scrivere nelle tre righe della griglia tre numeri, ciascuno composto da due cifre (per aiutarvi, il 4 è stato già scritto). Le sei cifre devono essere tutte diverse e tali che il numero che si leggerà nella seconda riga risulti il doppio di quello della prima e che il numero della terza riga risulti il triplo di quello della prima.

Quale numero in particolare avete scritto nella seconda riga?

X1	X2
X3	X4
4	X5

X2, X4, X5 :: [0..9]

X1, X3 :: [1..9]

all_different(X1, X2, ..., X5)

X3*10 + X4 = 2*(X1*10 + X2)

4*10+X5 = 3*(X1*10 + X2)

CSP e ASIA-GiM – Esempio 2
Ma poi, fatto il modello?

Giochi d'Autunno 2016 – Categoria CE – Gioco 4

Dovete scrivere nelle tre righe della griglia tre numeri, ciascuno composto da due cifre (per aiutarvi, il 4 è stato già scritto). Le sei cifre devono essere tutte diverse e tali che il numero che si leggerà nella seconda riga risulti il doppio di quello della prima e che il numero della terza riga risulti il triplo di quello della prima.

Quale numero in particolare avete scritto nella seconda riga?

X1	X2
X3	X4
4	X5

X2, X4, X5 :: [0..9]

X1, X3 :: [1..9]

all_different(X1, X2, ..., X5)

X3*10 + X4 = 2*(X1*10 + X2)

4*10+X5 = 3*(X1*10 + X2)

```
:- use_module(library(clpfd)).

test2 :-
    [X2, X4, X5] ins 0..9
    , [X1, X3] ins 1..9
    , all_distinct([X1, X2, X3, X4, X5])
    , X3*10 + X4 #=  2*(X1*10 +  X2)
    , 4*10 + X5 #=  3*(X1*10 +  X2)
    , Result #= X3*10+X4
    , label([X1, X2, X3, X4, X5, Result])
    , write(Result)
    .
```

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Paola Mello, Federico Chesani

DISI
Università di Bologna

paola.mello@unibo.it
federico.chesani@unibo.it

www.unibo.it