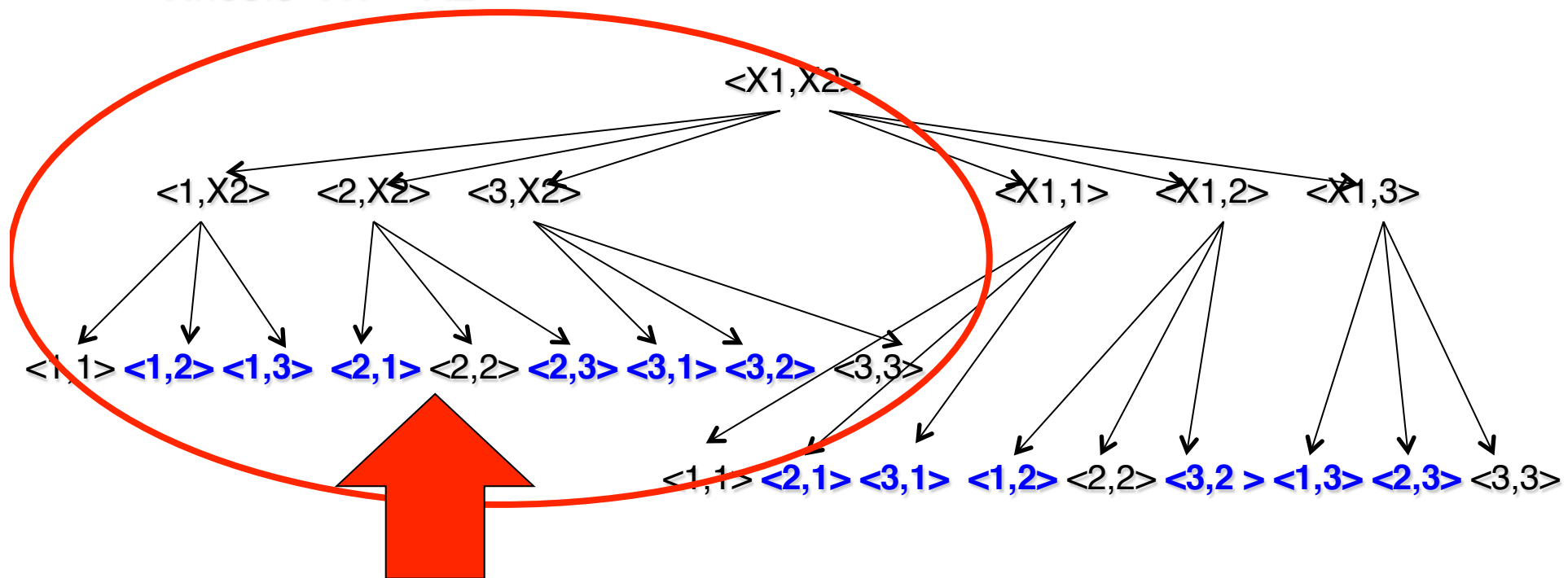


Constraint Satisfaction Problems (CSPs)

- Molti problemi di IA possono essere formulati come problemi per i quali la soluzione è l'assegnamento di valori (scelti da un insieme noto) a variabili (in numero noto)
- Le variabili sono legate tra loro da vincoli, specifici del problema
- Ricerca in uno spazio degli stati, dove lo stato è rappresentabile come n-pla di variabili $\langle X_1, \dots, X_n \rangle$
- Operatori, assegnamento di un valore ad una variabile della n-pla (**labeling**)
- Albero di ricerca, detto anche albero decisionale

ESEMPIO: 2 variabili

- X1 con possibile valore in {1,2,3}
- X2 con possibile valore in {1,2,3}
- Vincolo $X1 \neq X2$



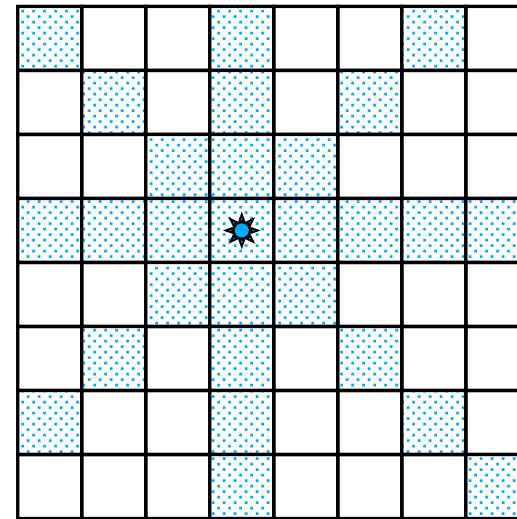
(3^2 foglie, solo alcune sono soluzione (in blu))

Constraint Satisfaction Problem (CSP)

- Si rappresenta la ricerca con un albero, detto albero decisionale
- Un passo nell'albero corrisponde alla scelta della prossima variabile da istanziare e al suo *labeling* (assegnamento di un valore)
- Nell'esempio precedente, se scelgo prima X1 poi X2, l'albero decisionale coincide con la porzione di albero cerchiata in rosso
- Per (cercare di) tagliare a-priori le foglie che non sono soluzione si possono utilizzare i vincoli del problema:
 - Staticamente , prima di effettuare alcun *labeling* (***tecniche di consistenza***)
 - Dinamicamente (frammischiati al *labeling*), durante la ricerca (metodi generativi e ***algoritmi di propagazione***)

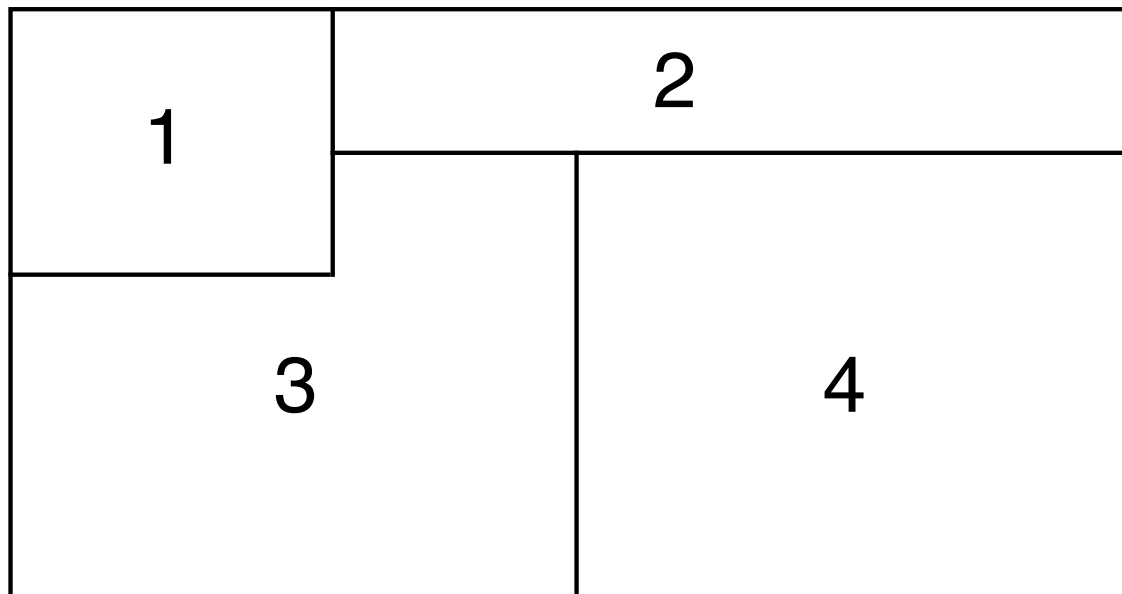
SODDISFACIMENTO DI VINCOLI

- 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.
- **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.



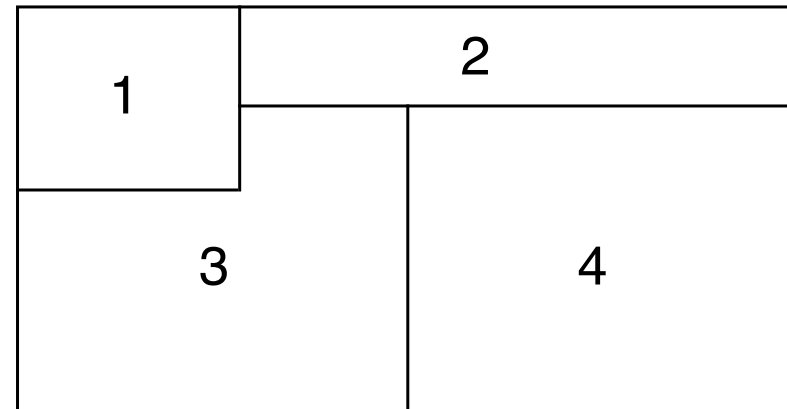
ESEMPIO: MAP COLORING

- 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)



MAP-COLORING come CSP

- Quattro colori sono sufficienti per ogni mappa (dimostrato solo nel 1976). Variante facile del problema del Graph Coloring– al massimo quattro regioni sono connesse con tutte.
- Variabili: regioni X_1, X_2, X_3, X_4
- Domini: colori permessi $[r, g, b]$
- Vincoli:
 - Regioni adiacenti devono avere colori diversi
 - $X_1 \neq X_2$ and $X_1 \neq X_3$ and $X_2 \neq X_3$ and $X_2 \neq X_4$ and $X_3 \neq X_4$



CRIPTOARITMETICA

•

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

- Due lettere diverse non possono avere lo stesso valore;
- Le somme delle cifre devono corrispondere a quanto illustrato dal problema.
- Obiettivo: determinare uno stato in cui ad ogni lettera è associata una cifra in modo che i vincoli iniziali siano soddisfatti.
- I problemi di progetto sono tipicamente problemi a vincoli: un progetto deve rispettare vincoli di costo, tempo, materiali ecc.

SUDOKU COME CSP

- 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
- inoltre, ogni riga orizzontale e ogni riga verticale dell'intera tavola non deve contenere ripetizioni di numeri
- Ogni casella è una variabile con dominio da 1 a 9;
- Vincoli di diversità (tanti...).
- Si illustreranno vincoli speciali (Alldifferent) in un corso successivo (→ **Constraint Processing**)

SUDOKU: griglia

		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		

PROBLEMA DELLE 8 REGINE: MODELLO 1

- Le caselle della scacchiera sono rappresentate da $N \times N$ variabili (8x8). Molto numerose.
- L'istanziamento di una variabile **N** al valore **1** indica che e' posizionata una regina, se il valore è **0** la posizione è 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:

MODELLO 2 (quello che adotteremo)

- Le otto regine (una per ciascuna colonna) vengono rappresentate con le variabili
 $x_1, x_2, \dots, x_8$
- Il pedice si riferisce alla colonna occupata dalla corrispondente regina.
- L'istanziamento di una variabile x_i al valore k appartenente all'insieme $[1..8]$ indica che la regina corrispondente viene posizionata sulla k -esima riga della i -esima colonna.
- Le variabili hanno come insieme di possibili valori gli interi compresi tra 1 e 8 che corrispondono alle righe occupate.

VINCOLI: DUE TIPI

- Quelli che vincolano i valori delle variabili al dominio suddetto e quelli che devono impedire un attacco reciproco e che impongono, quindi, relazioni tra i valori assunti dalle variabili.

$1 \leq x_i \leq 8$	per $1 \leq i \leq 8$
$x_i \neq x_j$	per $1 \leq i < j \leq 8$
$x_i \neq x_j + (j-i)$	per $1 \leq i < j \leq 8$
$x_i \neq x_j - (j-i)$	per $1 \leq i < j \leq 8$

- Il primo vincolo impone che i valori assunti dalle variabili del problema siano compresi tra i numeri interi 1 e 8: vincoli unari
- I tre vincoli successivi definiscono relazioni tra le variabili e, in particolare, tra due variabili alla volta: vincoli binari

Due tipi di vincoli (continua)

$$1 \leq x_i \leq 8$$

$$\text{per } 1 \leq i \leq 8$$

$$x_i \neq x_j$$

$$\text{per } 1 \leq i < j \leq 8$$

$$x_i \neq x_j + (j - i)$$

$$\text{per } 1 \leq i < j \leq 8$$

$$x_i \neq x_j - (j - i)$$

$$\text{per } 1 \leq i < j \leq 8$$

- Il secondo 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.

ESEMPIO: SCHEDULING COME CSP

- Scheduling:
 - assegnare azioni (con una certa durata) a risorse per un certo tempo (le risorse possono essere condivise).
- Variabili: start time delle attività
- Domini: possibili start time delle attività
- Vincoli:
 - Le attività che usano la stessa risorsa non possono essere eseguite in intervalli che si sovrappongono es:
$$\text{Start1} + d1 \leq \text{Start2} \text{ or } \text{Start2} + d2 \leq \text{Start1}$$
 - Eventualmente altri vincoli dipendenti dal problema, quali ordinamenti fra attività.

→ CSP PIÙ FORMALMENTE

- Formalmente un CSP (Constraints Satisfaction Problem) può essere definito su un insieme finito di variabili:
 $(x_1, x_2, \dots, x_n)$ i cui valori appartengono a domini finiti di definizione $(D_1, D_2, \dots, D_n)$, e su un insieme di vincoli.
- Un vincolo $c(x_{i1}, x_{i2}, \dots, x_{ik})$ tra k variabili è un sottoinsieme del prodotto cartesiano $D_{i1} \times D_{i2} \times \dots \times D_{ik}$ 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.
- Una soluzione ad un CSP prevede un assegnamento di tutte le variabili che soddisfi tutti i vincoli.

Differenti CSP

- Variabili Discrete
 - Domini Finiti:
 - n variabili, con domini di dimensione d (→ **quelli che vedremo**)
 - Interi, stringhe ecc,
 - e.g., job- scheduling, variabili rappresentano giorni di inizio-fine per ogni lavoro
 - *vincoli di durata* $StartJob_1 + 5 \leq StartJob_3$
- Variabili Continue
 - Programmazione lineare (→ **Ricerca Operativa**)

CSP come ricerca nello spazio degli stati

- CSP:
 - **state** è definito da **variabili** X_i con **valori** presi dai **domini** D_i
 - **goal test** è un insieme di **vincoli** che specificano le combinazioni di valori permessi (in modo intensionale).
 - **Operatori** sono gli assegnamenti di valori a variabili (*labeling*)
- Linguaggio di formalizzazione generale
- Algoritmi *general-purpose* per la soluzione.

CSP come possibile algoritmo di ricerca

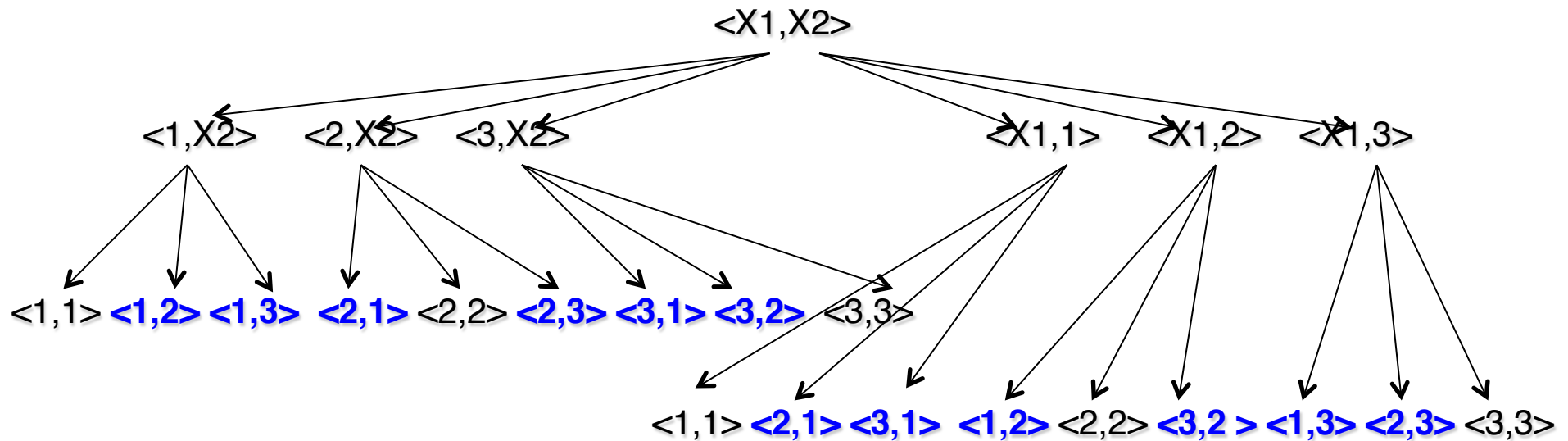
- **Stato iniziale:** assegnamento vuoto { }
 - **Funzione Successore:** assegna un valore ad una variabile non ancora legata (in modo che sia consistente – ovvero rispetti i vincoli - con gli assegnamenti già fatti).
 - fallisci se non esiste
 - **Goal test:** l'assegnamento è completo (tutte le variabili sono legate).
1. Schema identico per tutti i CSPs
 2. Profondità limitata a n se n sono le variabili.
 - usa depth-first search
 3. La strada è irrilevante.
 4. Problema commutativo con d^n foglie (se d è la cardinalità dei domini e n il numero di variabili)

ALBERO DECISIONALE

- Qualunque tecnica di soluzione di problemi NP-difficili fa uso, almeno concettualmente, di un **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

- $X1$ con possibile valore in $\{1,2,3\}$
 $X2$ con possibile valore in $\{1,2,3\}$
- Vincolo $X1 \neq X2$



(3^2 foglie, solo alcune sono soluzione, in blu in figura)

ALBERO DECISIONALE

- 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
- **Escludere rami che non portano a soluzione è fondamentale:**
 - Tecniche di consistenza
 - Algoritmi di propagazione
- Senza perdita di generalità ci riferiremo, nel seguito, a **CSP su vincoli binari** (vincoli cioè che coinvolgono due variabili)

SOLUZIONE DI CSP: DUE APPROCCI

- Dato un CSP esistono due possibili approcci per la sua risoluzione: uno basato sulle Tecniche di Consistenza e l'altro su Algoritmi di Propagazione.
- **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 queste sono integrate durante la ricerca.

ALGORITMI DI PROPAGAZIONE

USO DEI VINCOLI A PRIORI E A POSTERIORI

- Consideriamo una ricerca depth-first in un albero decisionale. Si tende a scendere di livello nell'albero fino a quando o si sono assegnate tutte le variabili, e quindi si è trovata una soluzione, oppure non è più possibile trovare un valore (la sequenza corrente non può portare a una soluzione ammissibile); quindi si esegue un'altra scelta sull'ultima variabile della sequenza stessa
- L'algoritmo ha tre gradi di libertà:
 - la scelta nell'ordinamento delle variabili;
 - la scelta nell'ordine di selezione del valore da attribuire alla variabile corrente;
 - la propagazione effettuata in ciascun nodo
- I primi due riguardano le euristiche sulla strategia di ricerca.

Propagazione dei vincoli:

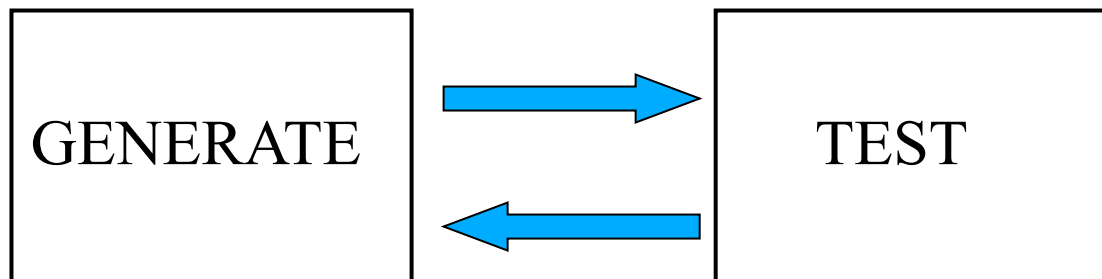
- Il terzo grado di libertà è ciò che differenzia le diverse strategie.
- La propagazione dei vincoli può avvenire **a posteriori**, considerando le sole variabili già istanziate:
- **Algoritmi (generativi)**
 - Generate and Test
 - Standard Backtracking

La propagazione dei vincoli può avvenire **a priori**, anche su variabili non ancora istanziate:

- **Algoritmi di Propagazione:**
 - Forward Checking
 - (Partial and Full) Look Ahead

ALGORITMI generativi

- Le due tecniche che usano i vincoli a posteriori sono:
 - Il Generate and Test (GT)
 - Lo Standard Backtracking (SB).



GENERATE AND TEST

- L'interprete del linguaggio sviluppa e visita un albero decisionale percorrendolo in profondità, assegnando valori alle variabili senza preoccuparsi di verificare la consistenza con gli altri vincoli.
- Esempio, istanza del problema di Map Coloring:

- Vincoli di dominio: $X1, X2, X3, X4, X5 :: [\text{rosso}, \text{giallo}, \text{verde}, \text{blu}]$

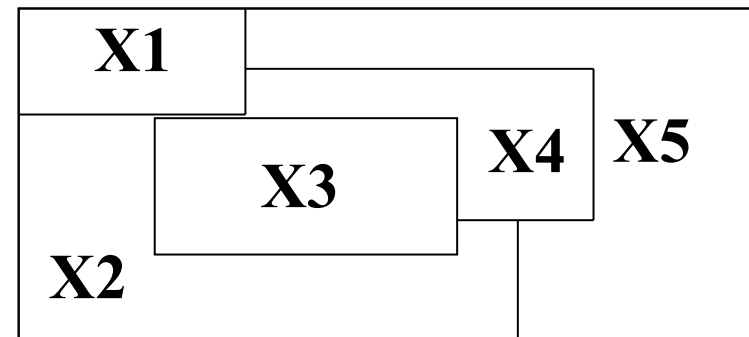
- Vincoli topologici:

$X1 \neq X2, X1 \neq X3, X1 \neq X4, X1 \neq X5,$

$X2 \neq X3, X2 \neq X4, X2 \neq X5,$

$X3 \neq X4,$

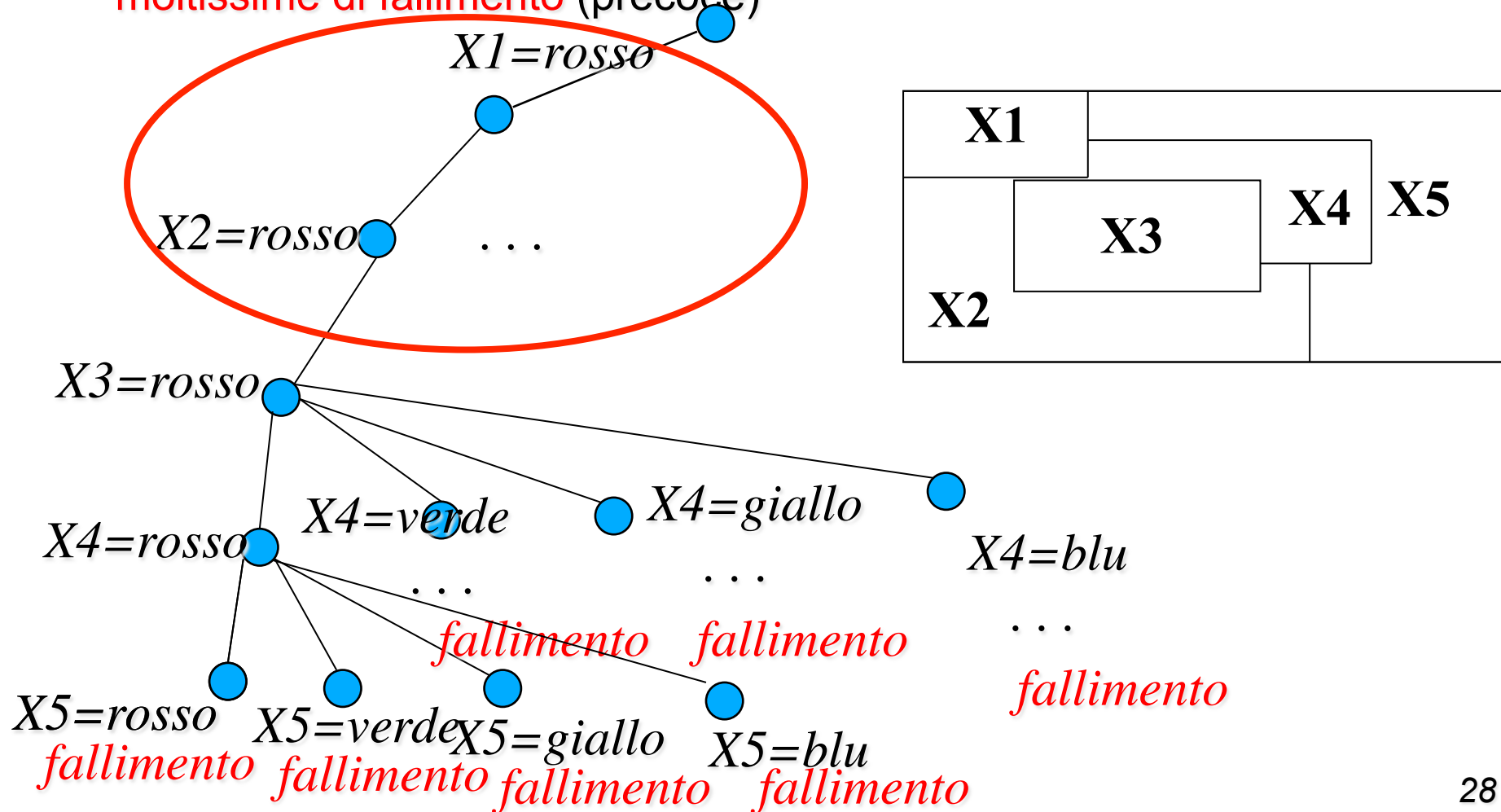
$X4 \neq X5$



- Generate&Test esplora depth-first l'albero decisionale, fino a una foglia, e verifica se è soluzione o meno
- **Albero decisionale con 4^5 foglie**

Generate and Test per Map coloring

- Algoritmo poco efficiente (albero decisionale con 4^5 foglie, 1024 foglie),
moltissime di fallimento (precoce)

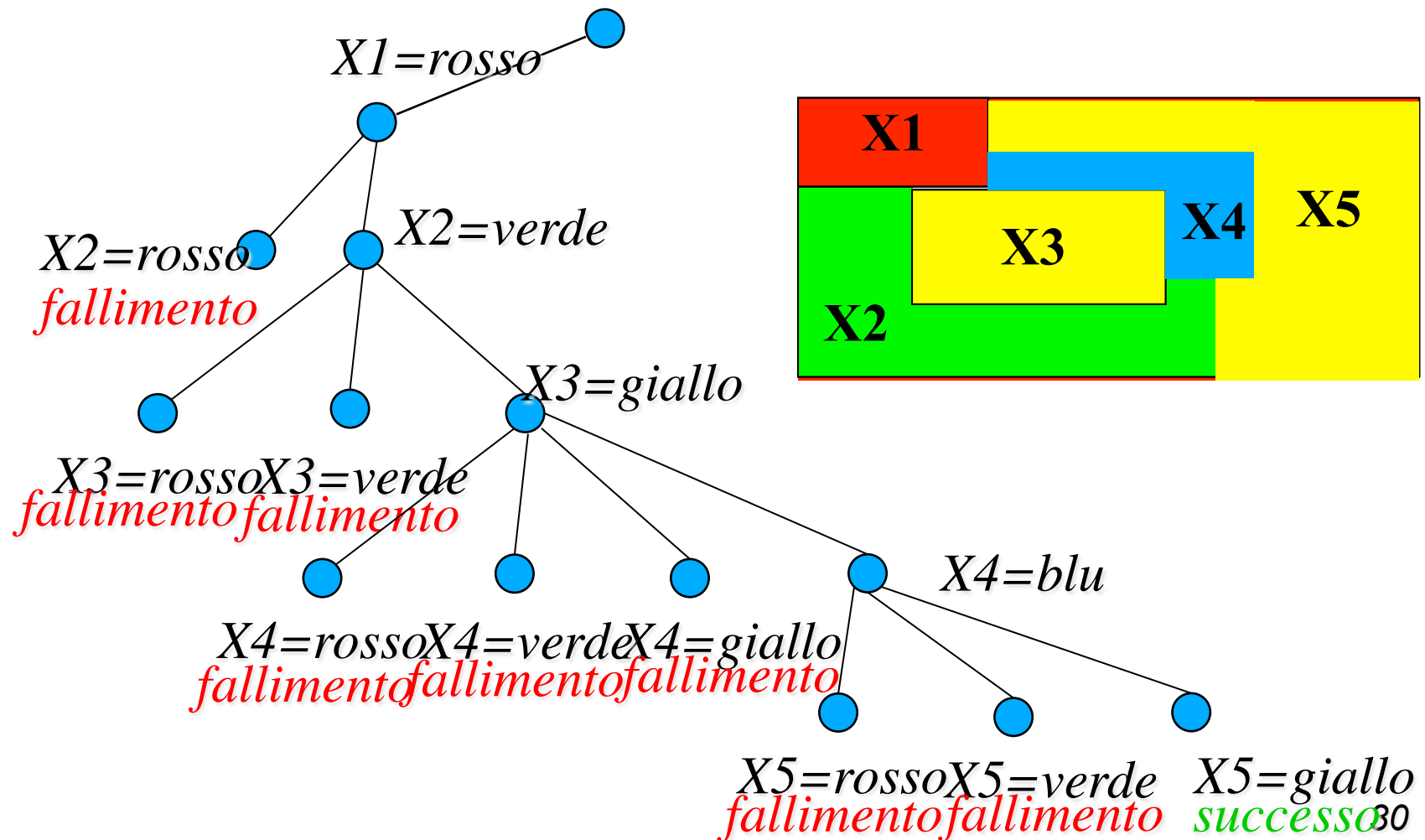


STANDARD BACKTRACKING

- Anche questa tecnica prevede un **utilizzo a posteriori dei vincoli**, ma su porzioni di soluzione - è migliore della precedente
 - A ogni istanziazione di una variabile si preoccupa di verificare la consistenza della variabile appena istanziata con quelle assegnate già 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 delle inconsistenze.

Standard Backtracking per Map coloring

- Algoritmo semplice



Standard Backtracking

- Depth-first search per CSPs con singolo assegnamento di variabili è chiamata **standard backtracking**
- **Standard Backtracking search** è l'algoritmo di ricerca non-informata basilare per CSP.

```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
  return RECURSIVE-BACKTRACKING({}, csp)

function RECURSIVE-BACKTRACKING(assignment, csp) returns a solution, or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(Variables[csp], assignment, csp)
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
    if value is consistent with assignment according to Constraints[csp] then
      add { var = value } to assignment
      result ← RECURSIVE-BACKTRACKING(assignment, csp)
      if result ≠ failure then return result
      remove { var = value } from assignment
  return failure
```

Propagazione dei vincoli:

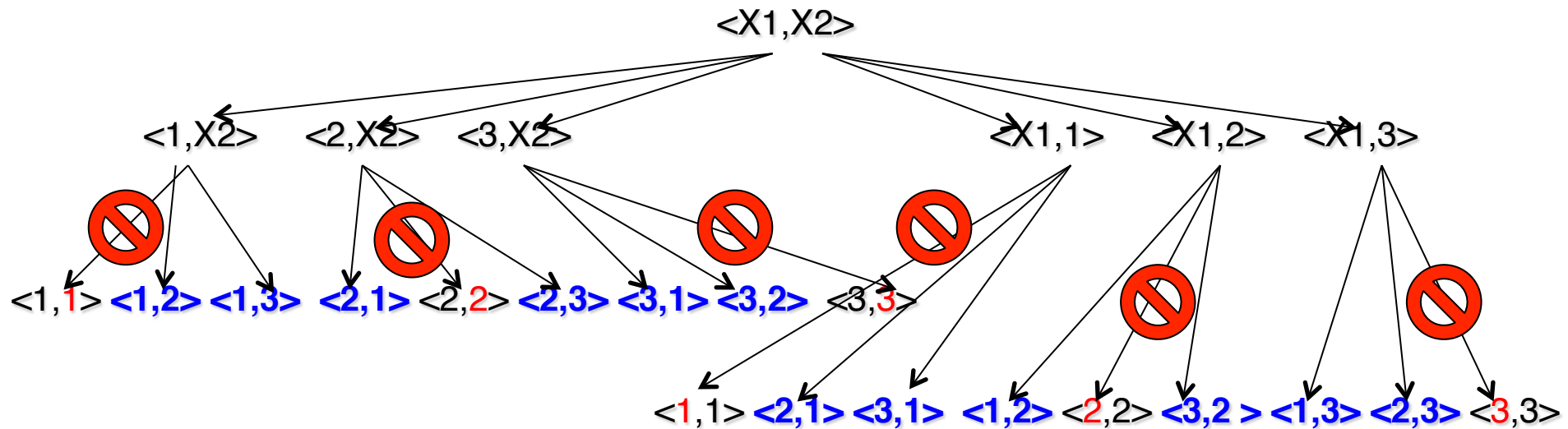
- Il terzo grado di libertà è ciò che differenzia le diverse strategie.
- La propagazione dei vincoli può avvenire **a posteriori**, considerando le sole variabili già istanziate:
- **Algoritmi (generativi)**
 - Generate and Test
 - Standard Backtracking

La propagazione dei vincoli può avvenire **a priori**, anche su variabili non ancora istanziate:

- **Algoritmi di Propagazione:**
 - Forward Checking
 - (Partial and Full) Look Ahead

ALBERO DECISIONALE: ESEMPIO

- X_1 con possibile valore in $\{1,2,3\}$
 X_2 con possibile valore in $\{1,2,3\}$
- Vincolo $X_1 \neq X_2$



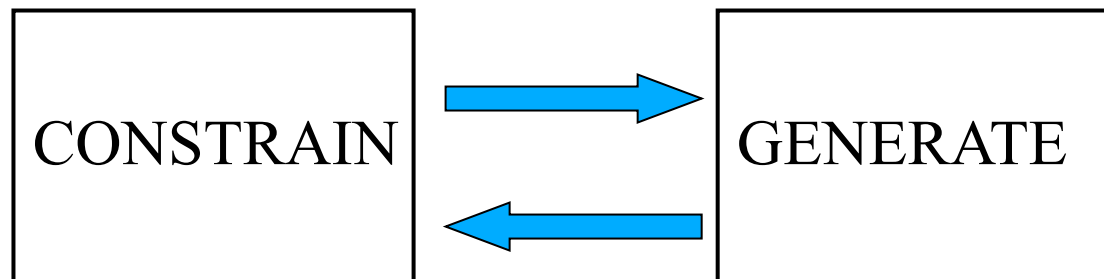
(3^2 foglie, solo alcune sono soluzione - in blu)

GLI ALGORITMI DI PROPAGAZIONE

- Gli algoritmi di propagazione sono metodi di ricerca più “intelligenti” (o meglio **più efficaci nel ridurre lo spazio di ricerca esplorato**) che tentano di ***prevenire i fallimenti*** anzichè recuperare fallimenti già avvenuti.
- *Pruning a priori* dell'albero delle decisioni.
- Utilizzare le relazioni tra le variabili del problema, i vincoli, per ridurre lo spazio di ricerca prima di arrivare al fallimento.
- Vengono così eliminati rami dell'albero che porterebbero ad un sicuro insuccesso limitando inutili backtracking.

ALGORITMI DI PROPAGAZIONE

- Gli algoritmi di propagazione si basano sul concetto inverso di quelli generativi
- Tecniche quali Forward Checking (FC), e il Looking Ahead (LA).
- Un modulo propaga i vincoli finché è possibile (*constrain*); alla fine della propagazione o si è giunti ad una soluzione (od a un fallimento) o sono necessarie nuove informazioni sulle variabili libere (*generate*, *fase di labeling di una variabile*).

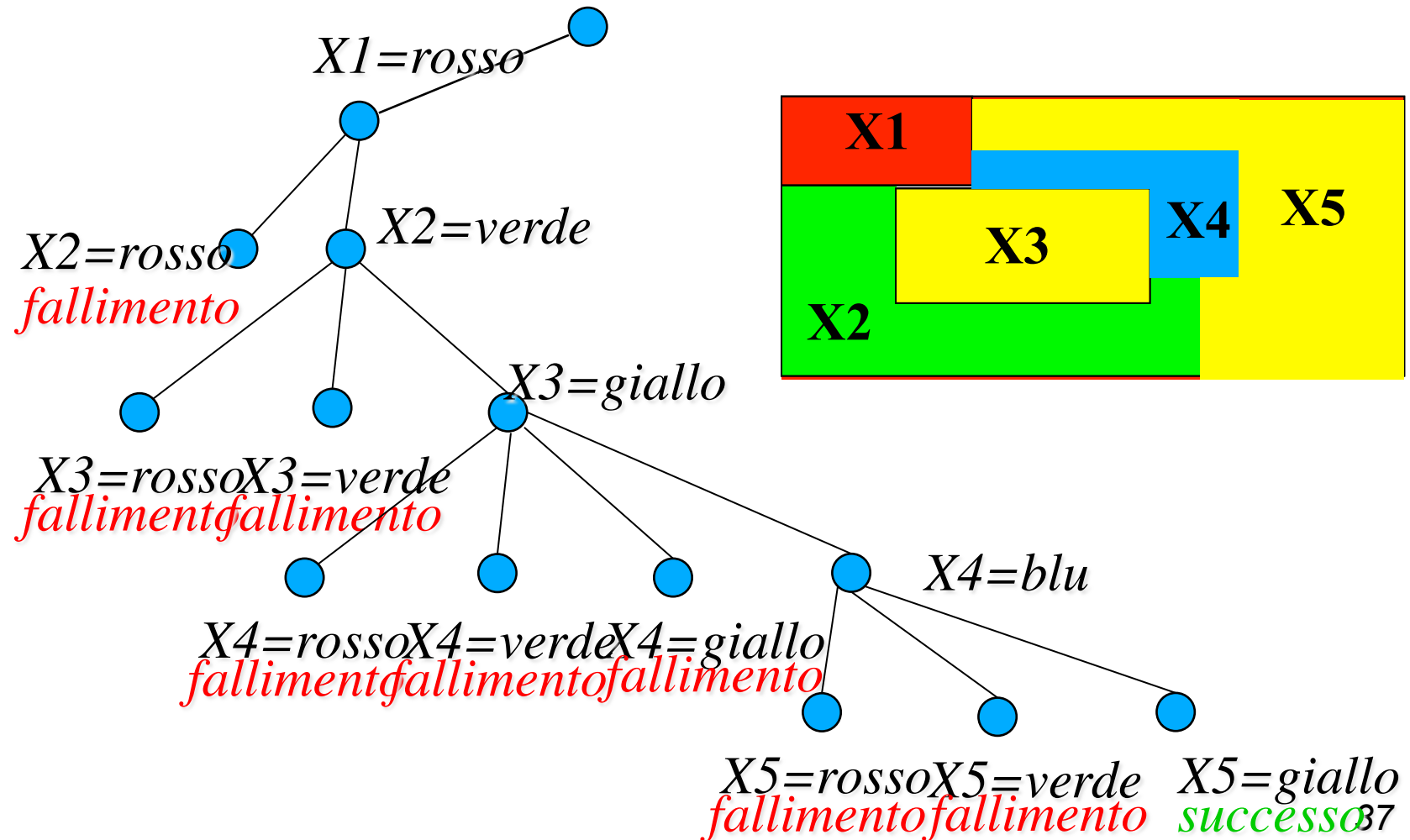


ALGORITMI DI PROPAGAZIONE

- 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
- La propagazione ha l'effetto di ridurre eventualmente i domini delle variabili future (appena un dominio risulta vuoto, fallimento)

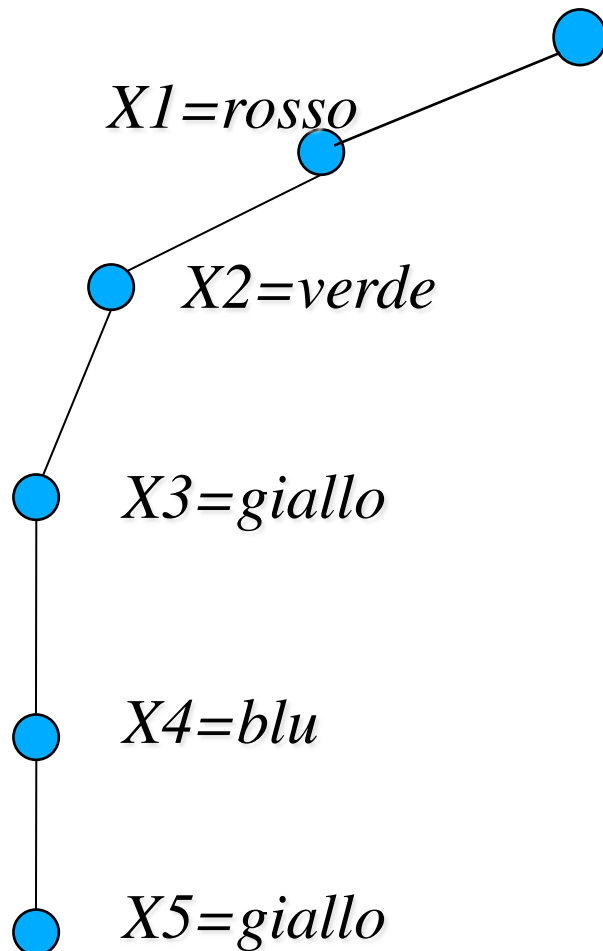
Standard Backtracking per Map coloring (*rivediamolo*)

- Algoritmo semplice

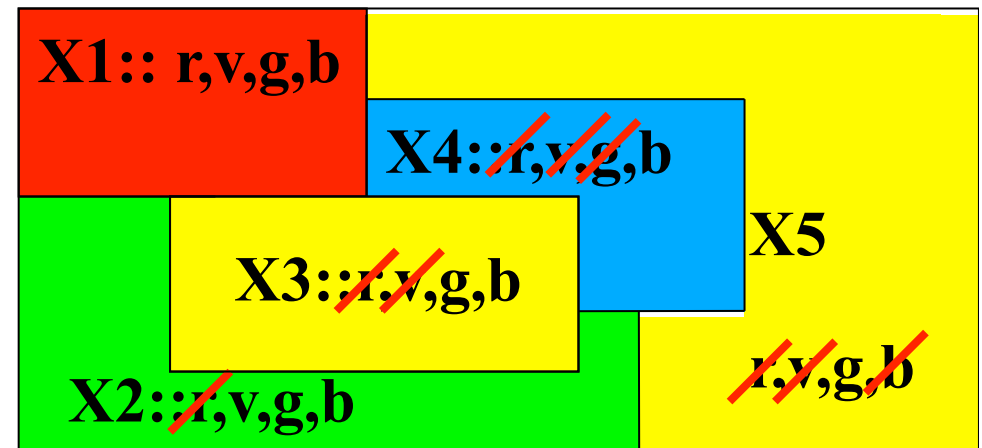


FORWARD CHECKING per Map Coloring

- Eliminazione a priori dei valori inconsistenti dai domini delle variabili future



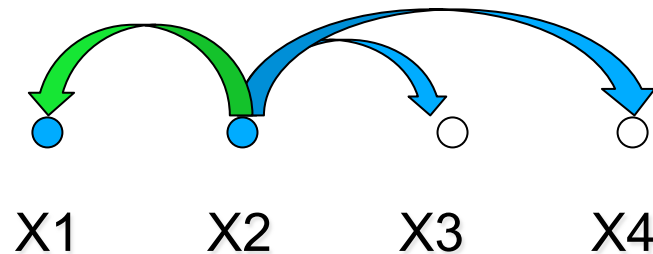
Il problema diventa addirittura deterministico ai livelli più profondi dell'albero



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 costo 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 si attua il **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



$c(x,y)$, vincolo tra x e y (“c” sta per constraint)

$X2$ ultima variabile oggetto di *labeling*

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*

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.

LOOK AHEAD

- Strategia di Partial Look Ahead (PLA) o Full Look Ahead (FLA).
- **PLA:** si ha una propagazione dei vincoli contenenti la variabile X_h , non ancora istanziata e le variabili "future", ossia le variabili $X_{h+1}, \dots, X_n$
 - Per ogni variabile non ancora assegnata $X_{k+1}, \dots, X_n$, 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.
- **FLA:** se V_k è il valore appena assegnato alla variabile X_k , si ha una propagazione dei vincoli contenenti la variabile X_h , non ancora istanziata, e tutte le variabili non ancora assegnate, ossia le variabili $X_{k+1}, \dots, X_{h-1}, X_{h+1}, \dots, X_n$.
 - per ogni variabile non ancora assegnata $X_{k+1}, \dots, X_n$ deve esistere un valore per il quale sia possibile trovare, per tutte le variabili non ancora assegnate, almeno un valore compatibile con esso.

ESEMPIO: PLA vs FLA

 $X_0=0$  X_1  X_2  X_3

- $X_0 < X_1 < X_2 < X_3$ con domini per $X_1, X_2, X_3::[1,2,3]$

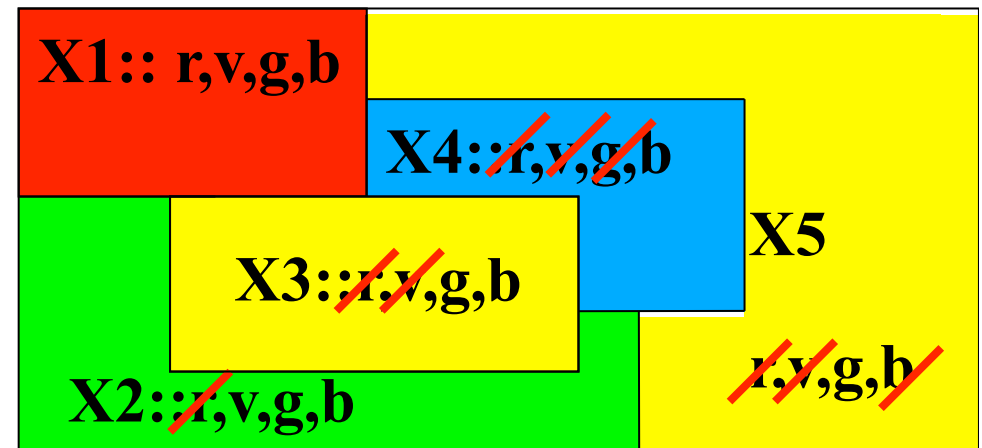
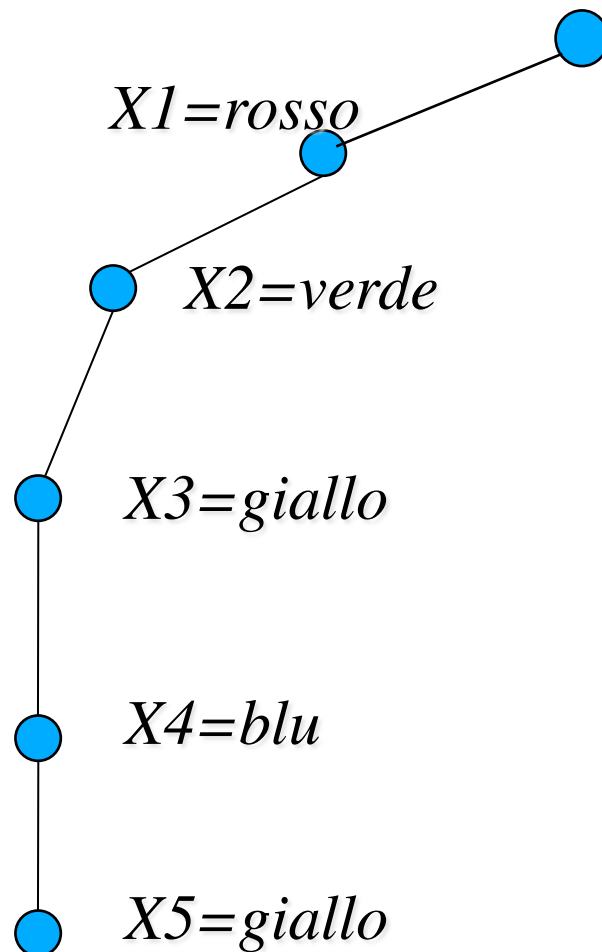
Stessi controlli di SB, propagazioni di FC, e in più PLA, verifica:

- per ogni valore in D1 se esiste almeno un valore in D2 e almeno un valore in D3 compatibili (sono eliminati i valori di D1 per i quali non esiste alcun valore compatibile in D2 o in D3),
- per ogni valore in D2 se esiste almeno un valore in D3 compatibile (sono eliminati i valori di D2 per i quali non esiste alcun valore compatibile in D3)

- | | | |
|----------------|-----------|------------|
| • $D1=[1,2,3]$ | dopo PLA: | $D1=[1,2]$ |
| • $D2=[1,2,3]$ | dopo PLA: | $D2=[1,2]$ |
| • $D3=[1,2,3]$ | | |

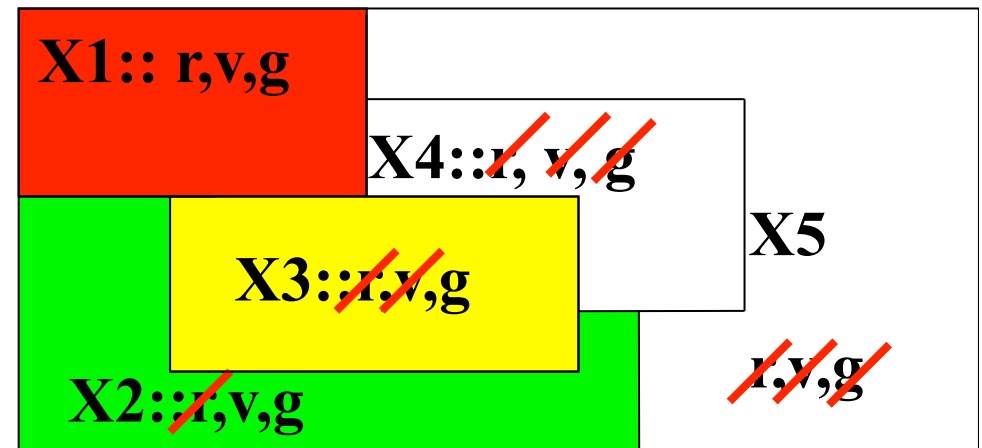
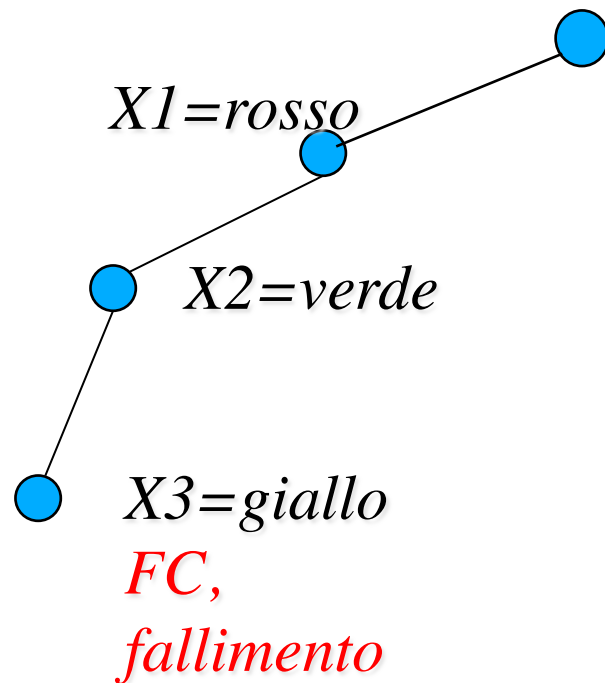
FC vs PLA per Map Coloring

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



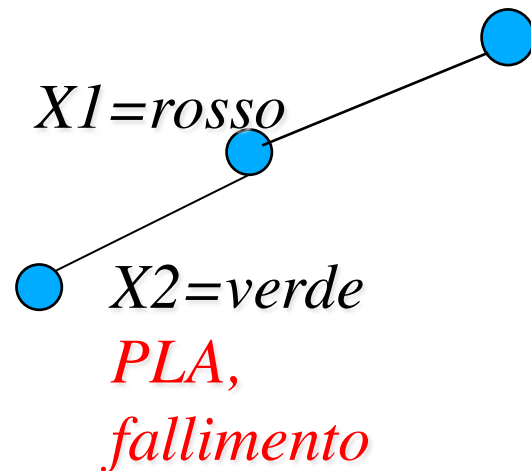
FC vs PLA per Map Coloring (*cont.*)

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



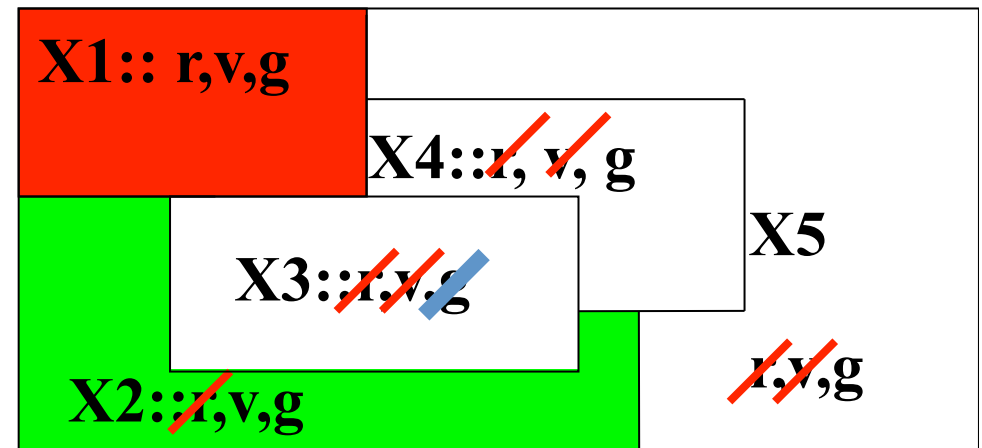
FC vs PLA per Map Coloring (*cont.*)

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



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)



ESEMPIO: PLA vs FLA

 $X_0=0$  X_1  X_2  X_3

- $X_0 < X_1 < X_2 < X_3$ con domini per $X_1, X_2, X_3::[1,2,3]$

PLA, verifica:

- per ogni valore in D1 se esiste almeno un valore in D2 e almeno un valore in D3 compatibili (sono eliminati i valori di D1 per i quali non esiste alcun valore compatibile in D2 o in D3),
- per ogni valore in D2 se esiste almeno un valore in D3 compatibile (sono eliminati i valori di D2 per i quali non esiste alcun valore compatibile in D3)

FLA, come PLA e inoltre controlla:

- per ogni valore in D2 se esiste almeno un valore in D1 compatibile (sono eliminati i valori di D2 per i quali non esiste alcun valore compatibile in D1),
- per ogni valore in D3 se esiste almeno un valore in D2 e almeno un valore in D1 compatibili (sono eliminati i valori di D3 per i quali non esiste alcun valore compatibile in D1 o in D2)

LOOK AHEAD

- Strategia di Partial Look Ahead (PLA) o Full Look Ahead (FLA).
- PLA: si ha una propagazione dei vincoli contenenti la variabile X_h , non ancora istanziata e le variabili "future", ossia le variabili $X_{h+1}, \dots, X_n$
 - Per ogni variabile non ancora assegnata $X_{k+1}, \dots, X_n$, 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.
- FLA: se V_k è il valore appena assegnato alla variabile X_k , si ha una propagazione dei vincoli contenenti la variabile X_h , non ancora istanziata, e tutte le variabili non ancora assegnate, ossia le variabili $X_{k+1}, \dots, X_{h-1}, X_{h+1}, \dots, X_n$.
 - per ogni variabile non ancora assegnata $X_{k+1}, \dots, X_n$ deve esistere un valore per il quale sia possibile trovare, per tutte le variabili non ancora assegnate, almeno un valore compatibile con esso.

ESEMPIO: PLA vs FLA (cont.)

● X0=0 ○ X1 ○ X2 ○ X3

- $X0 < X1 < X2 < X3$ con domini $X1, X2, X3 :: [1, 2, 3]$
- PLA $X1 :: [1, 2, \cancel{3}]$
 $X2 :: [1, 2, \cancel{3}]$
 $X3 :: [1, 2, 3]$
- FLA $X1 :: [1, 2]$ controlla le variabili future nei due sensi
 $X2 :: [\cancel{1}, 2]$
 $X3 :: [\cancel{1}, \cancel{2}, 3]$

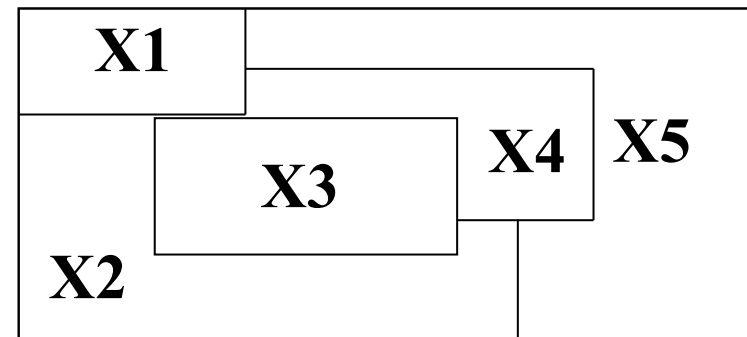
Riduce ulteriormente i domini delle variabili. In particolare, elimina i valori 1 e 2 dal dominio di X3 (non c'è supporto in D1 e D2) e il valore 1 dal dominio di X2 (non c'è supporto in D1) → diventa deterministico per X2 e X3

Esercizio Map Coloring

- Vincoli di dominio: $X1, X2, X3, X4, X5 :: [\text{rosso}, \text{giallo}, \text{verde}, \text{blu}]$
- Vincoli topologici:
 $X1 \neq X2, X1 \neq X3, X1 \neq X4, X1 \neq X5,$
 $X2 \neq X3, X2 \neq X4, X2 \neq X5,$
 $X3 \neq X4,$
 $X4 \neq X5$

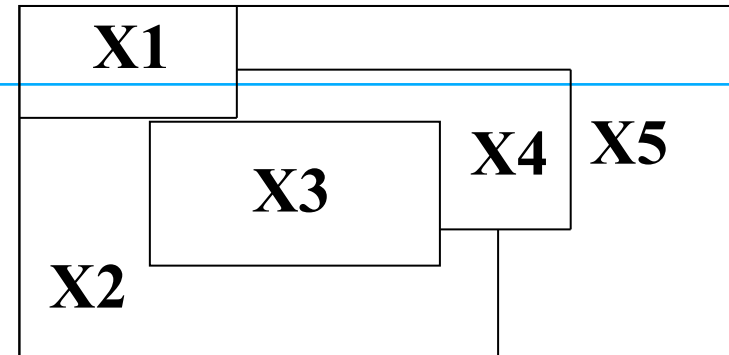
Vincolo unario:
 $X4 \neq b$

- Applicare
 - **FC**
 - **PLA**
 - **FLA**



✓ Forward checking

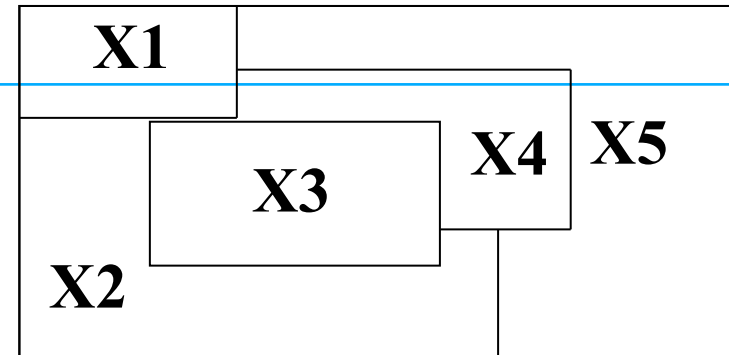
$X1 \neq X2$, $X1 \neq X3$, $X1 \neq X4$,
 $X1 \neq X5$, $X2 \neq X3$, $X2 \neq X4$,
 $X2 \neq X5$, $X3 \neq X4$, $X4 \neq X5$
 $X4 \neq b$ (*tolgo b dal dominio*)



X1	X2	X3	X4	X5
r,g,v,b	r,g,v,b	r,g,v,b	r,g,v	r,g,v,b
r	g,v,b	g,v,b	g,v	g,v,b
	g	v,b	v	v,b
		v	fail	

✓ Forward checking

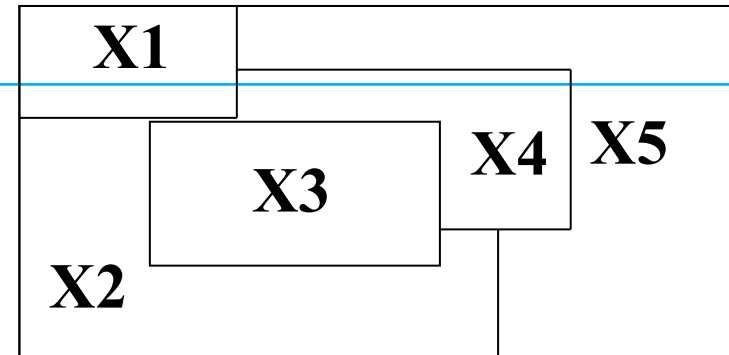
$X1 \neq X2$, $X1 \neq X3$, $X1 \neq X4$,
 $X1 \neq X5$, $X2 \neq X3$, $X2 \neq X4$,
 $X2 \neq X5$, $X3 \neq X4$, $X4 \neq X5$
 $X4 \neq b$ (*tolgo b dal dominio*)



X1	X2	X3	X4	X5
r,g,v,b	r,g,v,b	r,g,v,b	r,g,v	r,g,v,b
r	g,v,b	g,v,b	g,v	g,v,b
	g	v,b	v	v,b
		v	fail	

✓ Forward checking

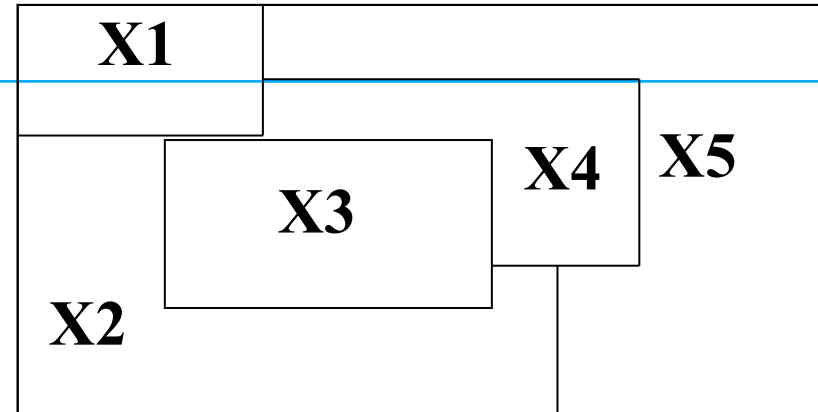
$X1 \neq X2$, $X1 \neq X3$, $X1 \neq X4$,
 $X1 \neq X5$, $X2 \neq X3$, $X2 \neq X4$,
 $X2 \neq X5$, $X3 \neq X4$, $X4 \neq X5$
 $X4 \neq b$ (*tolgo b dal dominio*)



X1	X2	X3	X4	X5
r,g,v,b	r,g,v,b	r,g,v,b	r,g,v	r,g,v,b
r	g,v,b	g,v,b	g,v	g,v,b
	g	v,b	v	v,b
		v	fail	
		b	v	b

✓ Partial Look Ahead

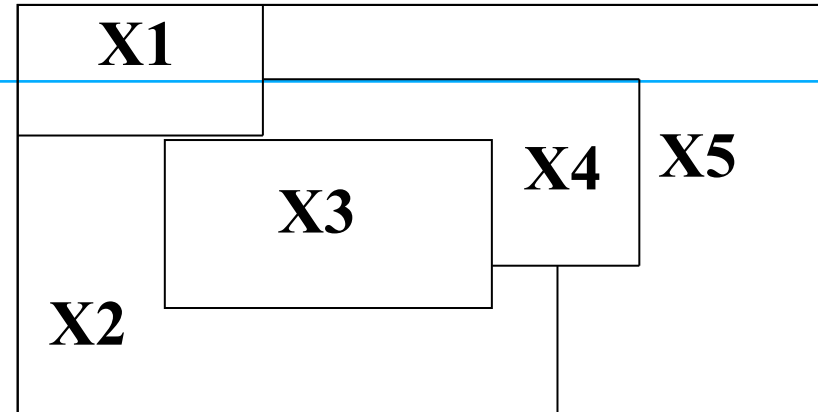
$X1 \neq X2$, $X1 \neq X3$, $X1 \neq X4$,
 $X1 \neq X5$, $X2 \neq X3$, $X2 \neq X4$,
 $X2 \neq X5$, $X3 \neq X4$, $X4 \neq X5$
 $X4 \neq b$



X1	X2	X3	X4	X5
r,g,v,b	r,g,v,b	r,g,v,b	r,g,v	r,g,v,b
r	g,v,b	g,v,b	g,v	g,v,b
	g	v ,b	v	v,b

✓ Partial Look Ahead

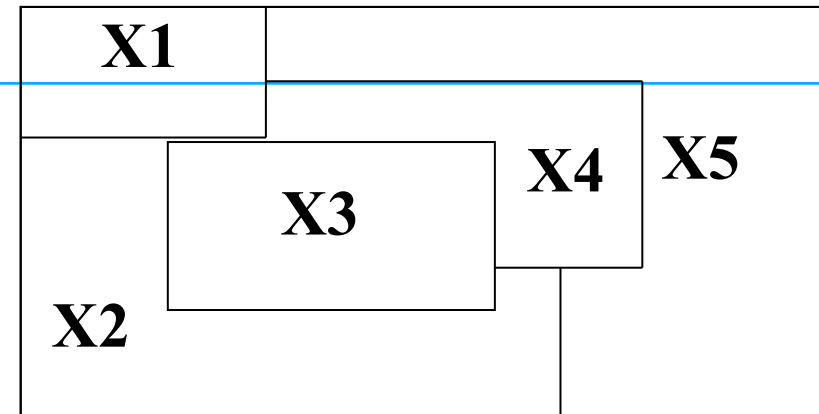
$X1 \neq X2$, $X1 \neq X3$, $X1 \neq X4$,
 $X1 \neq X5$, $X2 \neq X3$, $X2 \neq X4$,
 $X2 \neq X5$, $X3 \neq X4$, $X4 \neq X5$
 $X4 \neq b$



X1	X2	X3	X4	X5
r,g,v,b	r,g,v,b	r,g,v,b	r,g,v	r,g,v,b
r	g,v,b	g,v,b	g,v	g,v,b
	g	v ,b	v	v,b
		b	v	v ,b

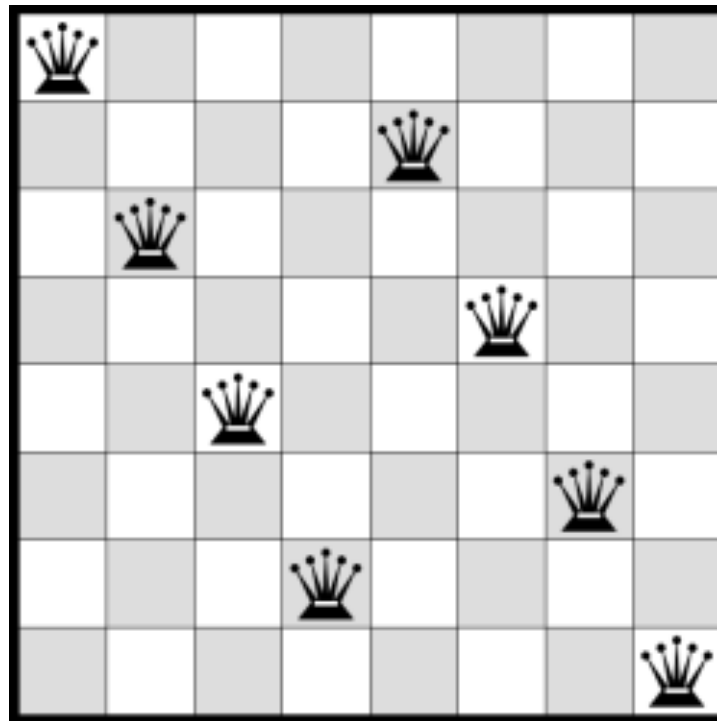
✓ Full Look Ahead

$X1 \neq X2$, $X1 \neq X3$, $X1 \neq X4$,
 $X1 \neq X5$, $X2 \neq X3$, $X2 \neq X4$,
 $X2 \neq X5$, $X3 \neq X4$, $X4 \neq X5$
 $X4 \neq b$



X1	X2	X3	X4	X5
r,g,v,b	r,g,v,b	r,g,v,b	r,g,v	r,g,v,b
r	g,v,b	g,v,b	g,v	g,v,b
	g	v ,b	v	v ,b

UN ESEMPIO TORMENTONE ...



LE 8 REGINE

- Consideriamo il problema delle otto regine: le variabili coinvolte nel problema prevedono, come dominio di definizione, i numeri interi compresi tra 1 e 8.

$$\begin{array}{ll} 1 \leq x_i \leq 8 & \text{per } 1 \leq i \leq 8 \\ x_i \neq x_j & \text{per } 1 \leq i < j \leq 8 \\ x_i \neq x_j + (j-i) & \text{per } 1 \leq i < j \leq 8 \\ x_i \neq x_j - (j-i) & \text{per } 1 \leq i < j \leq 8 \end{array}$$

- Il primo vincolo impone che i valori assunti dalle variabili del problema siano compresi tra i numeri interi 1 e 8: vincoli unari
- I tre vincoli successivi definiscono relazioni tra le variabili e, in particolare, tra due variabili alla volta: vincoli binari

LE 8 REGINE: GENERATE AND TEST

- Ciascuna regina deve essere su una riga diversa, possiamo generare le configurazioni finali limitandoci alla generazione delle permutazioni della sequenza:

(1,2,3,4,5,6,7,8)

- Unici vincoli considerati nella fase Generate:
 - $1 \leq X_i \leq 8$ per $1 \leq i \leq 8$
 - $X_i \neq X_j$ per $1 \leq i < j \leq 8$

Il secondo è dovuto al fatto che ogni tentativo consiste in una permutazione dei valori appartenenti ai domini e quindi ciascun valore assegnato alle variabili è diverso dagli altri
- Il Generate and Test avanza, nella ricerca di una soluzione, in modo 'miope' assegnando all'insieme delle variabili una permutazione degli interi che compongono il dominio

LE 8 REGINE: GENERATE AND TEST

- Così facendo una soluzione al problema può essere la seguente:
 $(X_1, X_2, \dots, X_8) = (1, 2, 3, 4, 5, 6, 7, 8)$. **Assegnamento di ogni regina a una casella appartenente alla diagonale principale. Sbagliato !**
- Solo in un secondo tempo questa tecnica considera gli altri vincoli rifiutando la soluzione trovata perché incompatibile con i vincoli del problema.
- A questo punto inizia la procedura di backtracking tentando con la seconda permutazione e così via finché non si trova una soluzione.
- Inefficienza di base
 - I vincoli sono utilizzati per limitare lo spazio delle soluzioni dopo che la ricerca è stata effettuata, quindi a posteriori.
 - Il numero delle possibili permutazioni aumenta con il fattoriale del numero di termini da permutare. Nel caso di $n=8$ **abbiamo un numero di permutazioni pari a $8! = 40320$** , se $n=10$ allora $n! = 3628800$ arrivando per $n=20$ a ordini di grandezza di 10^{18} e quindi dimensioni inaccettabili dallo spazio di ricerca.

LE 8 REGINE: STANDARD BACKTRACKING

- Il GT, produrrebbe la configurazione $(X_1, X_2, \dots, X_8) = (1, 2, 3, 4, 5, 6, 7, 8)$ sulla quale sarebbero poi verificati i vincoli di diagonale
- Con lo Standard Backtracking, invece, il tentativo effettuato dal Generate and Test per risolvere il problema delle otto regine assegnandole ad una diagonale della scacchiera verrebbe bloccato alla seconda istanziazione:
 - Quando $X_1=1$, assegnando a $X_2=2$, SB fallisce
 - Infatti il vincolo $X_i \neq X_j - (j-i)$ per $1 \leq i < j \leq 8$ è violato dalle prime due variabili: sostituendo $X_1 = 1$ e $X_2 = 2$ nella relazione si ottiene $1 \neq 2 - (2-1)$ che porta al fallimento
- Questa linea di ricerca viene quindi fermata e, con un backtracking, si tenta di assegnare a X_2 il valore 3 con successo, e così via.

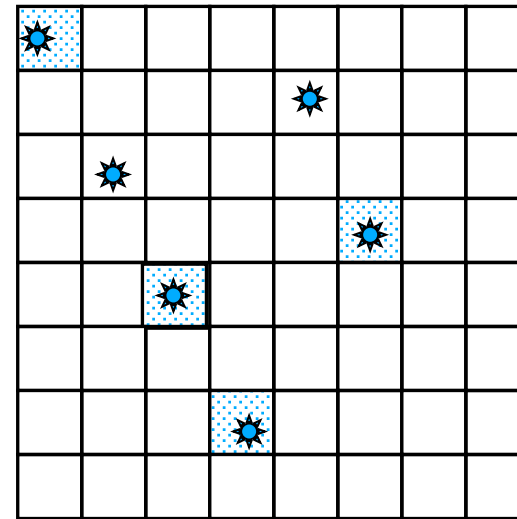
STANDARD BACKTRACKING

- 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 (sulle variabili già oggetto di labeling).
- A differenza della metodologia del Generate and Test, che tralascia il terzo e il quarto vincolo del problema delle otto regine (i vincoli sulle diagonali) durante la ricerca, per poi considerarli al termine dell'istanziamento di tutte le variabili, lo Standard Backtracking controlla la loro consistenza per ogni variabile istanziata.
- Quindi è prevista la verifica dei vincoli:
$$V_{k+1} \neq V_i + (k+1-i) \quad \text{per } 1 \leq i \leq k ; \quad \text{(diagonali a -45 gradi)}$$
$$V_{k+1} \neq V_i - (k+1-i) \quad \text{per } 1 \leq i \leq k ; \quad \text{(diagonali a 45 gradi)}$$
per ogni valore di k (sono gli stessi di prima, scritti in modo un po' diverso ...)

ESEMPIO SB PER LE 8 REGINE

Nella risoluzione del problema delle otto regine, vediamo come SB arriva ad istanziare le prime sei variabili ai valori:

$$(X_1, X_2, X_3, X_4, X_5, X_6) = (1, 3, 5, 7, 2, 4)$$



ESEMPIO SB PER LE 8 REGINE

- L'assegnazione $X_1=1$ è la prima scelta fatta.
- Se ora si assegnasse alla seconda variabile il valore $X_2=2$ verrebbe violato il vincolo
 - $V_{k+1} \neq V_i - (k+1-i)$ per $1 \leq i \leq k$.
- Infatti la seconda regina verrebbe posizionata sulla stessa diagonale della prima.
- In backtracking, viene tentato l'assegnamento $X_2=3$ che verifica con successo i vincoli:
 - $3 \neq 1$;
 - $1 \leq 3 \leq 8$;
 - $3 \neq 1 + (2 - 1)$;
 - $3 \neq 1 - (2 - 1)$.
- Per la terza variabile X_3 è ancora disponibile il valore 2 che però viola il vincolo sulla diagonale con la seconda variabile, infatti: $2 \neq 3 - (3 - 2)$ non è soddisfatto. Si innesca quindi di nuovo il backtracking.

ESEMPIO SB PER LE 8 REGINE

- Allora si procede all'assegnazione del valore 4 alla variabile X_3 che risulta però incompatibile con il valore assegnato alla variabile X_2 a causa del vincolo sulla diagonale. Backtracking su X_3 .
- Si tenta allora l'istanziatura $X_3=5$ che ha successo verificando i vincoli:

$$\begin{aligned}5 &\neq 1 ; 5 \neq 3; & 1 \leq 5 \leq 8 ; \\5 &\neq 1 + (3 - 1) ; 5 \neq 3 + (3 - 2) ; \\5 &\neq 1 - (3 - 1) ; 5 \neq 3 - (3 - 2) .\end{aligned}$$

- Procediamo ora all'istanziatura della quarta variabile al valore $X_4=7$ che soddisfa i vincoli :

$$\begin{aligned}7 &\neq 1 ; 7 \neq 3 ; 7 \neq 5 ; 1 \leq 7 \leq 8 ; \\7 &\neq 1 + (4 - 1) ; 7 \neq 3 + (4 - 2) ; 7 \neq 5 + (4 - 3) \\7 &\neq 1 - (4 - 1) ; 7 \neq 3 - (4 - 2) ; 7 \neq 5 - (4 - 3)\end{aligned}$$

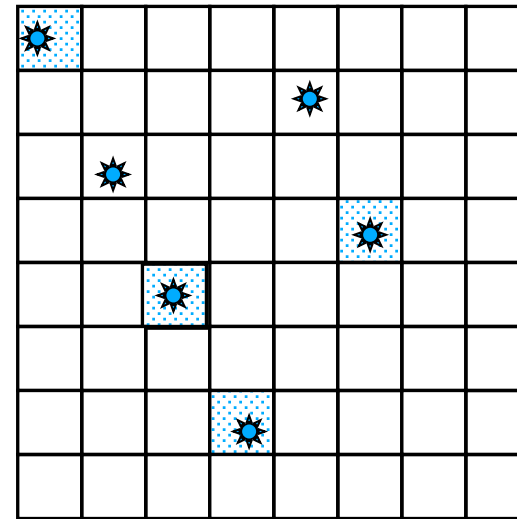
- Procedendo in questo modo si può facilmente verificare che anche le istanziazioni $X_5=2$ e $X_6=4$ soddisfano tutti i vincoli e quindi vengono accettate dallo Standard Backtracking.

ESEMPIO SB PER LE 8 REGINE

A questo punto:

$$(X_1, X_2, X_3, X_4, X_5, X_6) = (1, 3, 5, 7, 2, 4)$$

lo Standard Backtracking procede all'assegnazione della settima variabile all'ultimo valore ancora disponibile $X_7=6$, ne controlla la compatibilità con i valori già assegnati alle variabili $X_1, X_2, \dots, X_6$ e procede poi all'istanziamento dell'ultima regina.



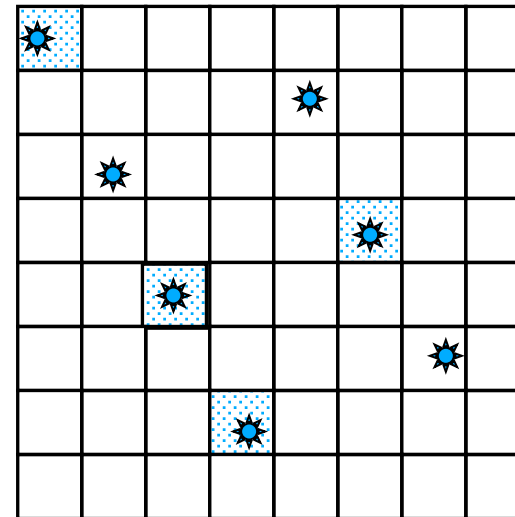
ESEMPIO SB PER LE 8 REGINE

Si assegna $X_7=6$

Si noti che l'ultima colonna, corrispondente alla variabile X_8 , non ha più posizioni disponibili assegnabili alla regina.

L'algoritmo SB solo dopo aver istanziato anche X_8 si 'accorge' di non avere più caselle disponibili e quindi fallisce.

La ricerca procede anche nel caso in cui una variabile ancora libera, nel nostro caso l'ultima (X_8) non presenti più posizioni disponibili.



LIMITI DELL'USO A POSTERIORI DEI VINCOLI

- Come abbiamo detto, questo è un difetto da attribuire a tutti i metodi che utilizzano i vincoli passivamente cioè posteriormente ad un tentativo di istanziamento.
- Utilizzando anche i vincoli che coinvolgono variabili ancora libere il problema sarebbe stato rilevato in anticipo evitando così costosi backtracking.
- L'idea che sta alla base degli **algoritmi di propagazione a priori** consiste in un 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 rimanenti dopo ogni assegnazione.
- Questi insiemi (domini) vengono perciò ridotti nel corso della computazione permettendo di scegliere per le variabili ancora libere valori ammissibili con le variabili già istanziate senza più considerare i vincoli che le legano.

LE 8 REGINE: FORWARD CHECKING

- Vediamo come con FC si arriva all'assegnazione dei valori (1,3,5,7,2,4) alla sequenza di variabili ($X_1, \dots, X_6$).
- Inizialmente l'insieme dei domini di tutte le variabili contiene gli interi compresi tra 1 e 8.
- Dopo l'istanziamento di X_1 al valore 1 vengono eliminati, dagli insiemi relativi a $X_2, \dots, X_8$, tutti i valori incompatibili con questo. Ovviamente il valore 1 viene tolto da tutti i domini a causa del vincolo

$$X_1 \neq X_i \text{ per } i = 2, \dots, 8.$$

- Dopo l'istanziamento del valore 1 alla variabile X_1 , vengono ridotti gli insiemi dei valori ammissibili per le variabili ancora libere. Infatti X_2 ha, come valori ammissibili (3,4,5,6,7,8) in quanto il valore 2 risulta incompatibile con l'assegnazione già effettuata e pertanto viene eliminato.

LE 8 REGINE: FC (*cont.*)

- Procedendo con questo ragionamento elenchiamo gli insiemi associati a ciascuna variabile:
 - X_2 è associata al dominio $D_2=(3,4,5,6,7,8)$,
 - X_3 è associata al dominio $D_3=(2,4,5,6,7,8)$,
 - X_4 è associato al dominio $D_4=(2,3,5,6,7,8)$,
 - X_5 è associato al dominio $D_5=(2,3,4,6,7,8)$,
 - X_6 è associato al dominio $D_6=(2,3,4,5,7,8)$,
 - X_7 è associato al dominio $D_7=(2,3,4,5,6,8)$,
 - X_8 è associato al dominio $D_8=(2,3,4,5,6,7)$.
- Ora si procede istanziando X_2 al valore 3. I domini delle variabili libere diventano:
 - X_3 è associata al dominio $D_3=(5,6,7,8)$,
 - X_4 è associato al dominio $D_4=(2,6,7,8)$,
 - X_5 è associato al dominio $D_5=(2,4,7,8)$,
 - X_6 è associato al dominio $D_6=(2,4,5,8)$,
 - X_7 è associato al dominio $D_7=(2,4,5,6)$,
 - X_8 è associato al dominio $D_8=(2,4,5,6,7)$.

LE 8 REGINE: FC (*cont.*)

- Ora si prosegue nella ricerca assegnando un valore a X_3 (il valore 5) e si propagano i vincoli ottenendo:
 - X_4 è associato al dominio $D_4=(2,7,8)$,
 - X_5 è associato al dominio $D_5=(2,4,8)$,
 - X_6 è associato al dominio $D_6=(4)$,
 - X_7 è associato al dominio $D_7=(2,4,6)$,
 - X_8 è associato al dominio $D_8=(2,4,6,7)$.
- Per la variabile X_6 il dominio contiene ora un solo valore (*vedi euristiche sulla scelta della variabile, nel seguito*). Si procede all'istanziamento della variabile X_4 al valore 2, il primo appartenente al suo dominio.
- Gli insiemi di valori rimanenti:
 - X_5 è associato al dominio $D_5=(4,8)$,
 - X_6 ha dominio D_6 vuoto,
 - X_7 è associato al dominio $D_7=(4,6)$,
 - X_8 è associato al dominio $D_8=(4,7)$.
- Si ha quindi il **fallimento** in seguito all'assenza di valori ammissibili per la variabile X_6 .

LE 8 REGINE: FC (*cont.*)

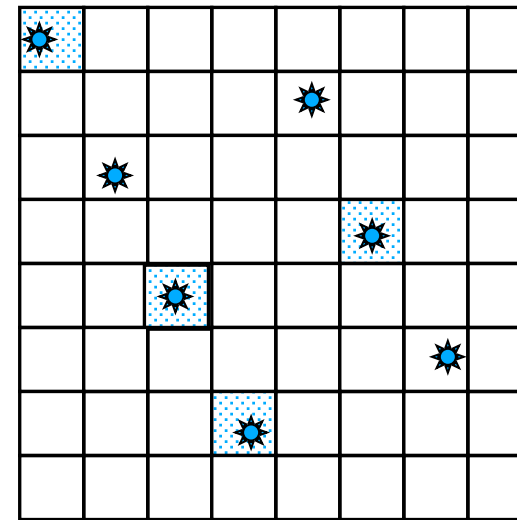
- Il backtracking porta alla ritrattazione del valore assegnato all'ultima variabile istanziata, della propagazione causata da quest'ultimo e al successivo tentativo con il valore $X_4=7$.
- L'algoritmo restringe i domini nel modo seguente:
 - X_5 è associato al dominio $D_5=(2,4)$,
 - X_6 è associato al dominio $D_6=(4)$,
 - X_7 è associato al dominio $D_7=(2,6)$,
 - X_8 è associato al dominio $D_8=(2,4,6)$.
- Poi si procede all'istanziamento della variabile X_5 al valore 2 e X_6 al valore 4 eliminando ogni possibile istanziazione per la variabile X_8 :
 - X_8 è associato al dominio $D_8=()$
- **Il FC si accorge che la sequenza di valori (1,3,5,7,2,4) per la sequenza di variabili $(X_1, \dots, X_6)$ non porta ad alcuna soluzione.**

ESEMPIO: SB vs FC

Assegnamenti effettuati: (1,3,5,7,2,4) per la sequenza di variabili ($X_1, \dots, X_6$)

SB assegna $X_7=6$ e solo dopo aver istanziato anche X_8 si 'accorge' di non avere più caselle disponibili e quindi fallisce

Il FC si accorge che la sequenza di valori (1,3,5,7,2,4) per la sequenza di variabili ($X_1, \dots, X_6$) non porta ad alcuna soluzione (il dominio D_8 è vuoto)

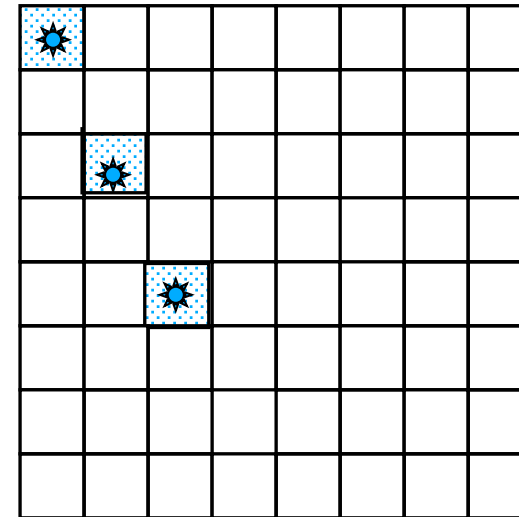


Si può fare meglio?

FC anticipa il fallimento di 2 livelli rispetto a SB per questo esempio.

LE 8 REGINE: PARTIAL LOOK AHEAD

- Supponiamo di avere assegnato alle prime tre variabili X_1, X_2, X_3 rispettivamente i valori 1,3,5.
- Applicando il Forward Checking, i domini delle variabili future si riducono a:
 - X_4 è associato al dominio $D_4=(2,7,8)$,
 - X_5 è associato al dominio $D_5=(2,4,8)$,
 - X_6 è associato al dominio $D_6=(4)$,
 - X_7 è associato al dominio $D_7=(2,4,6)$,
 - X_8 è associato al dominio $D_8=(2,4,6,7)$.
- PLA controlla anche alcune coppie di variabili non istanziate e può ridurre ulteriormente i domini ...



- PLA: Per ciascuna variabile non assegnata, si elimina ogni valore dal suo dominio per il quale non esiste, nei domini associati alle variabili non istanziate "successive" almeno un posizionamento compatibile con il valore considerato.

LE 8 REGINE: PLA (cont.)

X4 è associato al dominio $D_4 = (\cancel{2}, 7, 8)$,
X5 è associato al dominio $D_5 = (2, 4, 8)$,
X6 è associato al dominio $D_6 = (4)$,
X7 è associato al dominio $D_7 = (2, 4, 6)$,
X8 è associato al dominio $D_8 = (2, 4, 6, 7)$.

- PLA considera X_4 e i vincoli binari con le variabili “successive”:
 - Il valore 2 del suo dominio D_4 ha almeno un valore compatibile (4) in D_5 , (6) in D_7 e (4) in D_8 ma non ha alcun valore compatibile in D_6 (il valore 4, unico assegnabile a X_6 , non è compatibile). **Dal dominio D_4 viene quindi eliminato il valore 2 (con FC questo valore veniva provato, ma era cmq una strada di fallimento con conseguente backtracking ...)**
 - Il valore 7 del suo dominio D_4 ha almeno un valore compatibile (2) in D_5 , (4) in D_6 , (6) in D_7 e (6) in D_8 . Pertanto 7 non va eliminato dal dominio come, del resto, il valore 8.
 - Il dominio associato a X_4 diventa pertanto: $D_4 = (7, 8)$.

LE 8 REGINE: PLA (cont.)

X4 è associato al dominio $D4=(\cancel{2},7,8)$,
X5 è associato al dominio $D5=(2,\cancel{4},8)$,
X6 è associato al dominio $D6=(4)$,
X7 è associato al dominio $D7=(2,4,6)$,
X8 è associato al dominio $D8=(2,4,6,7)$.

- PLA considera X_5 e i vincoli binari con le variabili “successive”:
 - Il valore 2 del suo dominio D_5 ha almeno un valore compatibile (4) in D_6 , (6) in D_7 e (8) in D_8 . Pertanto 2 non va eliminato dal dominio.
 - Per il valore 4 del dominio D_5 si vede subito che l'unico valore associato a X_6 non ha più alcuna possibilità di soddisfare i vincoli quindi da D_5 si può eliminare il valore 4.
 - Non dà problemi invece il valore 8 in D_5 per la presenza dei valori 4 in D_6 , 2 in D_7 e 2 in D_8 che rispettano tutti i vincoli.
 - Il dominio associato a X_5 diventa pertanto:
 $D_5=(2,8)$

LE 8 REGINE: PLA (cont.)

X4 è associato al dominio $D4=(\cancel{2},7,8)$,

X5 è associato al dominio $D5=(2,\cancel{4},8)$,

X6 è associato al dominio $D6=(4)$,

X7 è associato al dominio $D7=(2,4,6)$,

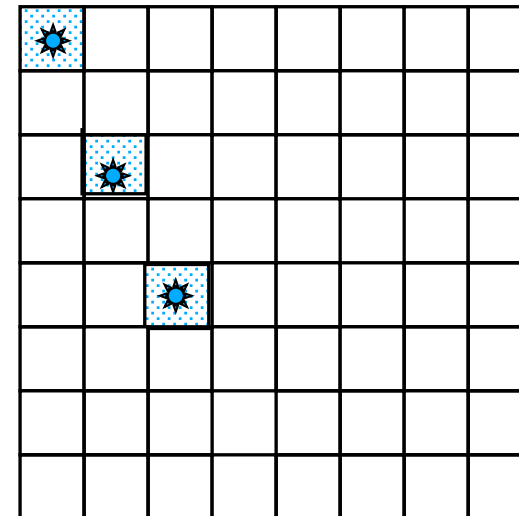
X8 è associato al dominio $D8=(2,4,6,7)$.

- PLA considera X_6 e i vincoli binari con le variabili “future”, e questa volta l'insieme D_6 non viene modificato.
- PLA considera X_7 e i vincoli binari con l'unica variabile “futura”: anche questa volta il dominio D_7 non viene modificato.

ESEMPIO: FC vs PLA

- Avendo già assegnato a X_1, X_2, X_3 rispettivamente i valori 1,3,5

- Con FC:
 - $D4=(2,7,8)$,
 - $D5=(2,4,8)$,
 - $D6=(4)$,
 - $D7=(2,4,6)$,
 - $D8=(2,4,6,7)$.
- Con PLA:
 - $D4=(7,8)$,
 - $D5=(2,8)$,
 - $D6=(4)$,
 - $D7=(2,4,6)$,
 - $D8=(2,4,6,7)$.

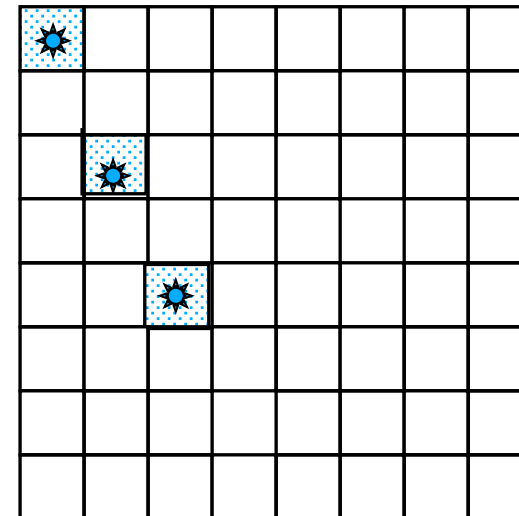


- **Con PLA, riusciamo ad evitare l'assegnazione del valore 2 alla variabile X_4 , che portava immediatamente al fallimento (la successiva propagazione FC, portava D_6 a diventare vuoto)**
- **Maggior *pruning* dell'albero con PLA vs FC, ma ...**

Si può fare ancora meglio?

ESEMPIO: FC vs PLA

- Avendo già assegnato a X_1, X_2, X_3 rispettivamente i valori 1,3,5
 - Con FC:
 - $D_4=(2,7,8)$,
 - $D_5=(2,4,8)$,
 - $D_6=(4)$,
 - $D_7=(2,4,6)$,
 - $D_8=(2,4,6,7)$.
 - Con PLA:
 - $D_4=(7,8)$,
 - $D_5=(2,8)$,
 - $D_6=(4)$,
 - $D_7=(2,4,6)$,
 - $D_8=(2,4,6,7)$.



Si noti che per il valore 8 in D_5 , non esiste alcun valore appartenente a D_4 che soddisfi i vincoli imposti dal problema.

Il PLA non si "accorge" di questa inconsistenza perché verifica la consistenza dei valori appartenenti ad un dominio D_i con i valori appartenenti ai domini D_j solo se $j > i$ → applichiamo FLA!

LE 8 REGINE: FULL LOOK AHEAD

- Riprendiamo l'esempio precedente, supponendo di avere già propagato i domini con la tecnica del PLA:
 - X_4 è associato al dominio $D_4=(7,8)$,
 - X_5 è associato al dominio $D_5=(2,8)$,
 - X_6 è associato al dominio $D_6=(4)$,
 - X_7 è associato al dominio $D_7=(2,4,6)$,
 - X_8 è associato al dominio $D_8=(2,4,6,7)$.
- **Applichiamo FLA:**
- FLA considera X_5 e i vincoli binari con le variabili non istanziate “precedenti”:
 - Per il valore 8 in D_5 , non esiste alcun valore appartenente a D_4 che soddisfi i vincoli imposti dal problema. Lo si cancella quindi.
 - Il valore 2 del dominio D_5 , ha almeno un compatibile (valore 4) in D_6 e (valori 7,8) in D_4 .
 - X_4 è associato al dominio $D_4=(7,8)$,
 - X_5 è associato al dominio $D_5=(2)$,
 - X_6 è associato al dominio $D_6=(4)$,

LE 8 REGINE: FLA (cont.)

X_4 è associato al dominio $D_4=(7,8)$,

X_5 è associato al dominio $D_5=(2)$,

X_6 è associato al dominio $D_6=(4)$,

X_7 è associato al dominio $D_7=(\cancel{2}, \cancel{4}, 6)$,

X_8 è associato al dominio $D_8=(\cancel{2}, \cancel{4}, \cancel{6}, \cancel{7})$.

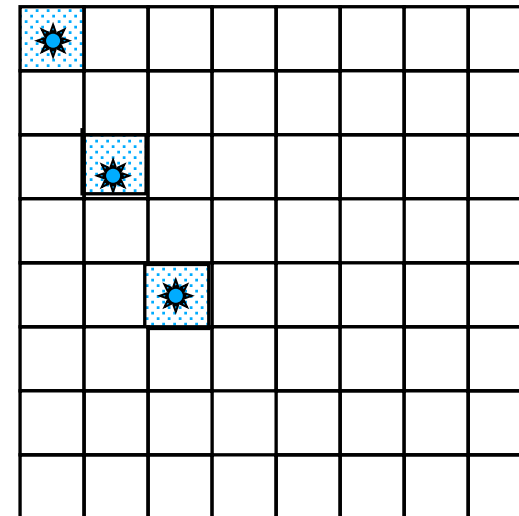
- FLA analizza il dominio D_7 : il valore 2 in D_7 non è compatibile con 2 in D_5 (unico valore di D_5) quindi viene eliminato. La stessa cosa avviene per 4 in D_7 incompatibile con 2 in D_5 :

X_7 è associato al dominio $D_7=(6)$,

- FLA analizza il dominio D_8 che, già a questo punto della computazione, non contiene più valori compatibili con quelli dei domini precedenti. Infatti il valore 2 in D_8 risulta incompatibile con 2 in D_5 , 4 in D_8 risulta incompatibile con 4 in D_6 , 6 e 7 in D_8 risultano incompatibili con 6 in D_7 .
- FLA porta la computazione a fallire senza tentare ulteriori assegnazioni sulle variabili future, già quando ho fatto i primi tre assegnamenti!**

ESEMPIO: PLA vs FLA

- Avendo già assegnato a X_1, X_2, X_3 rispettivamente i valori 1,3,5
 - Con PLA:
 - $D4=(7,8)$,
 - $D5=(2,8)$,
 - $D6=(4)$,
 - $D7=(2,4,6)$,
 - $D8=(2,4,6,7)$.
 - Con FLA:
 - $D4=(7,8)$,
 - $D5=(2)$,
 - $D6=(4)$,
 - $D7=(6)$,
 - **$D8=()$.**



Perché non usare allora sempre e solo FLA?

Il **carico computazionale** dovuto alle continue verifiche della consistenza dei vincoli, e quindi alla propagazione piuttosto pesante, non porta al raggiungimento di vantaggi quando le dimensioni del problema diventano considerevoli ai primi livelli dell'albero

– Nell'esempio delle otto regine infatti i domini ridotti, dopo le prime due istanziazioni, dalle tecniche in esame sono “identici”, ma, mentre il Look Ahead verifica la consistenza dei vincoli su tutte le coppie di variabili ancora libere (la maggioranza), **il Forward Checking esegue molte meno verifiche guadagnando in efficienza.**

USO DEI VINCOLI A PRIORI E A POSTERIORI (*ripetiamo*)

- Consideriamo una ricerca depth-first in un albero decisionale. Si tende a scendere di livello nell'albero fino a quando o si sono assegnate tutte le variabili, e quindi si è trovata una soluzione, oppure non è più possibile trovare un valore (la sequenza corrente non può portare a una soluzione ammissibile); quindi si esegue un'altra scelta sull'ultima variabile della sequenza stessa.
- L'algoritmo ha tre gradi di libertà:
 - la scelta nell'ordinamento delle variabili;
 - la scelta nell'ordine di selezione del valore da attribuire alla variabile corrente;
 - la propagazione effettuata in ciascun nodo.
- I primi due riguardano le euristiche sulla strategia di ricerca.

USO DEI VINCOLI A PRIORI E A POSTERIORI (*ripetiamo*)

- Il terzo grado di libertà è ciò che differenzia le diverse strategie:
- Algoritmi senza propagazione (generativi):
 - Generate and Test
 - Standard Backtracking
- Algoritmi di Propagazione
 - Forward Checking
 - (Partial and Full) Look Ahead

CLASSIFICAZIONE DELLE EURISTICHE

- La scelta dell'ordinamento delle variabili e la scelta dell'ordine di selezione dei valori rimangono a disposizione del programmatore
- Le euristiche potranno agire quindi su questi due gradi di libertà per cercare di garantire il raggiungimento di una buona soluzione in tempi ragionevoli anche per i problemi più complessi

CLASSIFICAZIONE DELLE EURISTICHE

Una prima classificazione delle euristiche è la seguente:

- euristiche statiche:
 - determinano l'ordine in cui le variabili (o i valori) vengono scelti prima di iniziare la ricerca; tale ordine rimane invariato durante tutta la ricerca.
- euristiche dinamiche:
 - scelgono la prossima selezione da effettuare ogni volta che una nuova selezione viene richiesta (quindi ad ogni passo di labeling).
- Le euristiche dinamiche sono potenzialmente migliori (meno backtracking). La determinazione dell'euristica perfetta (che non richiede backtracking) è un problema che ha, in genere, la stessa complessità del problema originale. Bisognerà quindi trovare un compromesso.

EURISTICHE DINAMICHE

Le euristiche possono riguardare:

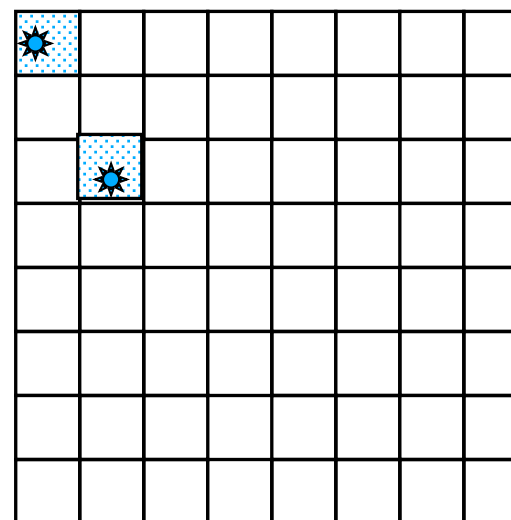
- euristiche per la selezione della variabile:
 - determinano quale deve essere la prossima variabile da istanziare. Le due euristiche più comunemente usate sono il **first-fail** (o **MRV**: Minimum Remaining Values) che sceglie la variabile con il dominio di cardinalità minore, e il **most-constrained principle** che sceglie la variabile legata a più vincoli. Entrambe queste euristiche decidono di istanziare prima le variabili più difficili da assegnare.
- euristiche per la selezione del valore:
 - determinano quale valore assegnare alla variabile selezionata. Si segue in genere il principio di scegliere prima il valore che si ritiene abbia più probabilità di successo (**least constraining principle**).

FORWARD CHECKING per 8 REGINE

(First Fail)

Consideriamo un punto della computazione in cui le prime due variabili sono assegnate: $X_1=1$, $X_2=3$

X_3 è associata al dominio $D_3=(5,6,7,8)$,
 X_4 è associato al dominio $D_4=(2,6,7,8)$,
 X_5 è associato al dominio $D_5=(2,4,7,8)$,
 X_6 è associato al dominio $D_6=(2,4,5,8)$,
 X_7 è associato al dominio $D_7=(2,4,5,6)$,
 X_8 è associato al dominio $D_8=(2,4,5,6,7)$.



FORWARD CHECKING per 8 REGINE

- Ora si prosegue nella ricerca assegnando un valore a X_3 (il valore 5) e si propagano i vincoli ottenendo:
 - X_4 è associato al dominio $D_4=(2,7,8)$,
 - X_5 è associato al dominio $D_5=(2,4,8)$,
 - X_6 è associato al dominio $D_6=(4)$,
 - X_7 è associato al dominio $D_7=(2,4,6)$,
 - X_8 è associato al dominio $D_8=(2,4,6,7)$.
- Per la variabile X_6 il dominio contiene ora un solo valore. Con **euristica statica**, scegliendo le variabili secondo l'ordine del loro pedice, si procede all'istanziamento della variabile X_4 al valore 2, il primo appartenente al suo dominio.
- Gli insiemi di valori rimanenti:
 - X_5 è associato al dominio $D_5=(4,8)$,
 - X_6 ha dominio D_6 vuoto,
 - X_7 è associato al dominio $D_7=(4,6)$,
 - X_8 è associato al dominio $D_8=(4,7)$.
- Si ha quindi il fallimento in seguito all'assenza di valori ammissibili per la variabile X_6 .

Euristica FIRST FAIL: 8 regine

- Nell'esempio precedente delle 8 Regine per l'algoritmo Forward Checking, dopo le istanziazioni: $X_1=1$, $X_2=3$, $X_3=5$
 - X_4 è associato al dominio $D_4=(2,7,8)$,
 - X_5 è associato al dominio $D_5=(2,4,8)$,
 - X_6 è associato al dominio $D_6=(4)$,
 - X_7 è associato al dominio $D_7=(2,4,6)$,
 - X_8 è associato al dominio $D_8=(2,4,6,7)$.
- il First fail proseguirebbe con la scelta per la variabile X_6 perché il suo dominio è il più ristretto (contiene il solo valore 4). FC riduce i domini:
 - X_4 è associato al dominio $D_4=(7,8)$,
 - X_5 è associato al dominio $D_5=(2,8)$,
 - X_7 è associato al dominio $D_7=(2,6)$,
 - X_8 è associato al dominio $D_8=(7)$.
- il First fail proseguirebbe con la scelta per la variabile X_8 perché il suo dominio è il più ristretto (contiene il solo valore 7).

Euristica FIRST FAIL: 8 regine

- Dopo le istanziazioni: $X_1=1$, $X_2=3$, $X_3=5$, $X_6=4$, $X_8=7$, il FC riduce i domini delle variabili libere a:
 - X_4 è associato al dominio $D_4=(8)$,
 - X_5 è associato al dominio $D_5=(2,8)$,
 - X_7 è associato al dominio $D_7=(2)$,
- il First fail proseguirebbe con la scelta per la variabile X_4 (o X_7)
- Al successivo *labeling* si arriva comunque al fallimento

RICERCA E PROPAGAZIONE

(riassumiamo)

- Ricerca depth-first, con scelta della variabile e del valore (euristiche) per il *labeling* ed eventuale propagazione:
- Algoritmi senza propagazione:
 - Generate and Test
 - Standard Backtracking
- Algoritmi di Propagazione
 - Forward Checking
 - (Partial and Full) Look Ahead

RICERCA E PROPAGAZIONE

(riassumiamo)

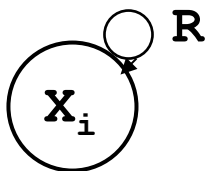
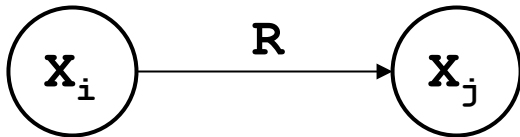
- Ricerca depth-first, con scelta della variabile e del valore (euristiche) per il *labeling* ed eventuale propagazione:
- Algoritmi senza propagazione:
 - Generate and Test
 - Standard Backtracking
- Algoritmi di Propagazione
 - Forward Checking
 - (Partial and Full) Look Ahead **ma non solo**
- Tecniche di Consistenza
 - Basati sulla propagazione dei vincoli per derivare un problema più semplice di quello (completo) originale (**prima della ricerca**)

TECNICHE DI CONSISTENZA

- Differenza fondamentale: al contrario degli algoritmi di propagazione che 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 (**Algoritmo AC-3**)
- 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).

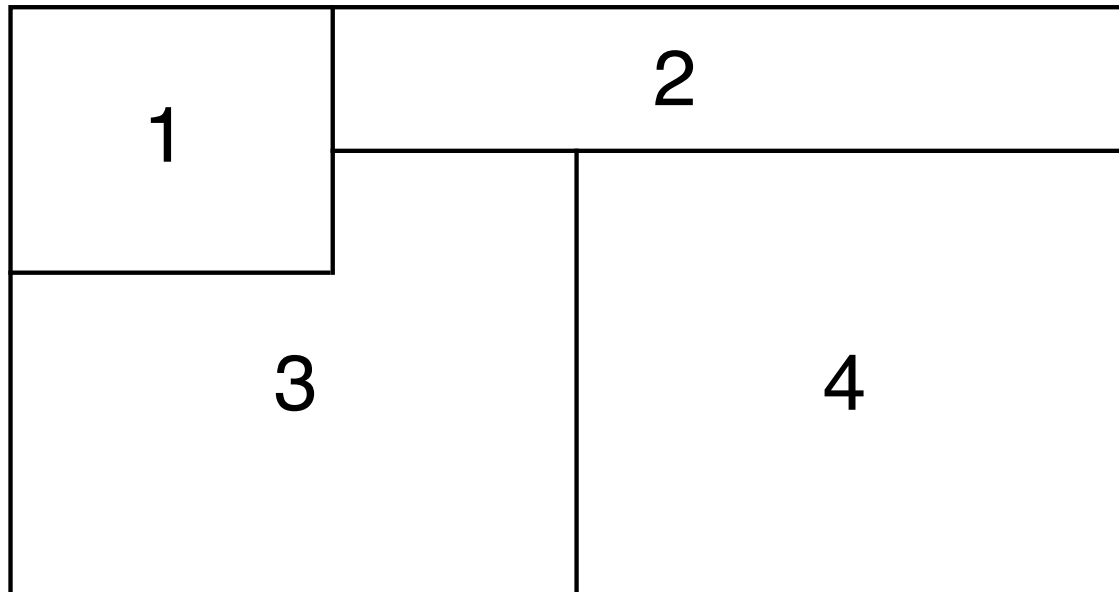
CONSTRAINT GRAPH

- 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 X_j :
 - I vincoli unari sono rappresentati da archi che iniziano e terminano sullo stesso nodo X_i



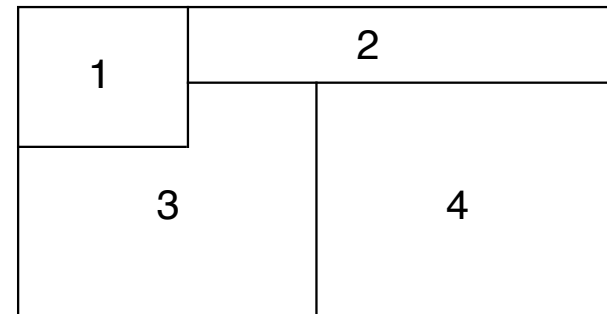
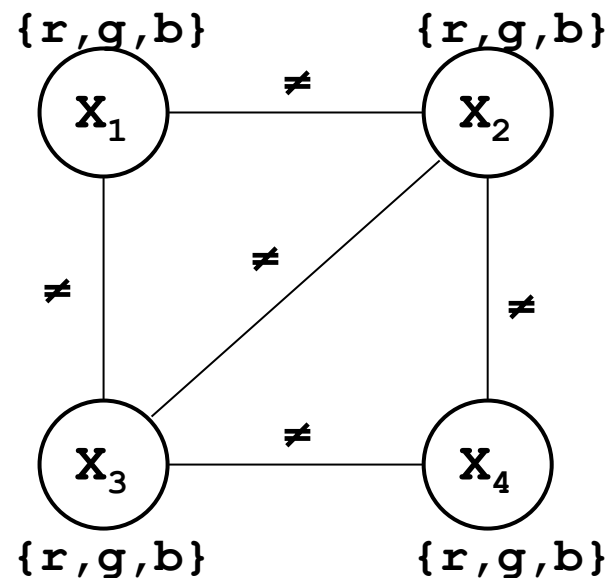
ESEMPIO: MAP COLORING PROBLEM

- 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)

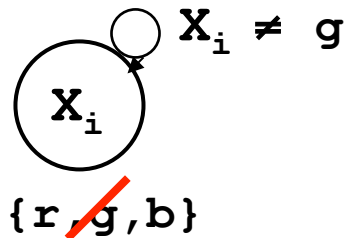


ESEMPIO: MAP COLORING PROBLEM

- Il constraint-graph corrispondente è il seguente. Tuttavia, esistono combinazioni di valori non compatibili tra loro (es: $X_1=r$, $X_2=r$, $X_3=r$, $X_4=r$).
- Esistono diversi algoritmi che realizzano gradi diversi di consistenza.

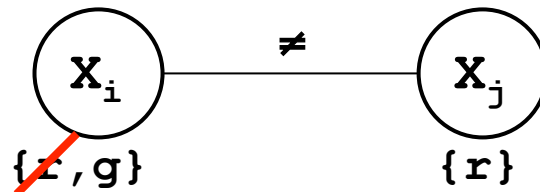


NODE CONSISTENCY



- **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 è 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.

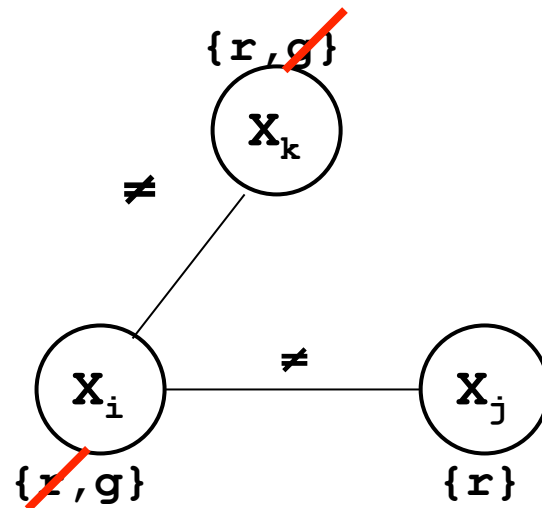
ARC CONSISTENCY



- 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 $r \in D_j$, non esiste un valore appartenente a D_i che soddisfi il vincolo $P(i,j)$.
- Per rendere consistente l'arco tra X_i e X_j è necessario eliminare il valore r dal dominio di X_i : questo valore non comparirebbe in nessuna soluzione ammissibile.

PROCEDIMENTO ITERATIVO

→ vedi esempio famiglie e case (dopo)



- La rimozione di alcuni valori dal dominio di una variabile rende necessarie ulteriori verifiche che coinvolgono i vincoli contenenti la variabile stessa.
- Quindi questo procedimento deve essere ripetuto finché la rete non raggiunge una configurazione stabile QUIESCENZA
- La rimozione del valore $X_i = r$ a causa del vincolo $\neq$ tra X_i e X_j comporta la rimozione del valore $X_k = g$ a causa del vincolo tra X_k e X_i
- **Quindi l'arc consistency è un procedimento iterativo che converge ad una rete stabile e arc-consistente.**

ESEMPIO: PLA vs FLA (*già visto prima.*)

● X0=0 ○ X1 ○ X2 ○ X3

- $X0 < X1 < X2 < X3$ con domini $X1, X2, X3 :: [1, 2, 3]$

- PLA $X1 :: [1, 2, \cancel{3}]$
 $X2 :: [1, 2, \cancel{3}]$
 $X3 :: [1, 2, 3]$

- FLA $X1 :: [1, 2]$
 $X2 :: [1, 2]$
 $X3 :: [\cancel{1}, \cancel{2}, 3]$

Riduce ulteriormente i domini delle variabili. In particolare, elimina i valori 1 e 2 dal dominio di X3 (non c'è supporto in D1 e D2) e il valore 1 dal dominio di X2 (non c'è supporto in D1) → diventa deterministico per X2 e X3

ESEMPIO: PLA vs FLA (*già visto prima.*)

● X0=0 ○ X1 ○ X2 ○ X3

- $X0 < X1 < X2 < X3$ con domini $X1, X2, X3 :: [1, 2, 3]$

- PLA $X1 :: [1, 2, \cancel{3}]$
 $X2 :: [1, 2, \cancel{3}]$
 $X3 :: [1, 2, 3]$

- FLA $X1 :: [1, 2]$
 $X2 :: [\cancel{1}, 2]$
 $X3 :: [\cancel{1}, \cancel{2}, 3]$

Riduce ulteriormente i domini delle variabili. In particolare, elimina i valori 1 e 2 dal dominio di X3 (non c'è supporto in D1 e D2) e il valore 1 dal dominio di X2 (non c'è supporto in D1) → diventa deterministico per X2 e X3

ARC CONSISTENCY vs FLA: ESEMPIO (*cont.*)

- $X_1 < X_2 < X_3$ con domini $X_1, X_2, X_3 :: [1, 2, 3]$

- **Prima** iterazione:
- **Seconda** iterazione:
- Terza iterazione: quiescenza

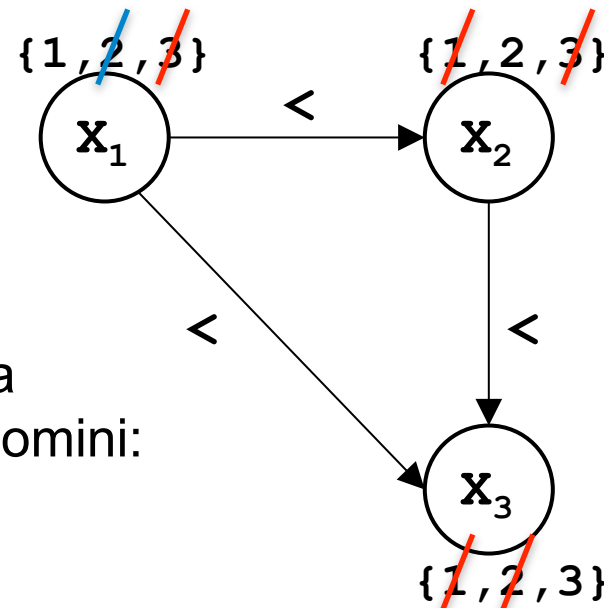
Il problema diventa deterministico (caso felice). Si noti che per il problema analogo visto prima, il FLA portava ai domini:

$X_1 :: [1, 2]$

$X_2 :: [2]$

$X_3 :: [3]$

Minor *pruning* vs AC, ma minor costo computazionale (FLA è anche chiamato AC1/2)



ARC CONSISTENCY

- Il controllo della consistenza dell' arco può essere applicato sia prima della ricerca, come *preprocessing* per produrre un problema *semplificato* oppure come passo di propagazione (in analogia al *look ahead*) dopo ogni assegnamento di variabile: è spesso chiamato *Maintaining Arc Consistency* (MAC) in questo caso
- L' algoritmo completo per il controllo di consistenza di un arco è chiamato **AC-3** (Mackworth, 1977)
- Utilizza una coda di archi (*queue*), iterando finché non è vuota
- Non appena il dominio di una variabile X_i è ridotto per la cancellazione di un valore, si riaggiungono a *queue* gli archi (X_k, X_i) per ciascuna variabile X_k collegata da un arco incidente su X_i .

Algoritmo AC-3 (Mackworth, 1977)

(→ vedi esempio mappa Australia)

```
function AC-3(csp) returns the CSP, possibly with reduced domains
  inputs: csp, a binary CSP with variables  $\{X_1, X_2, \dots, X_n\}$ 
  local variables: queue, a queue of arcs, initially all the arcs in csp

  while queue is not empty do
     $(X_i, X_j) \leftarrow \text{REMOVE-FIRST}(\textit{queue})$ 
    if RM-INCONSISTENT-VALUES( $X_i, X_j$ ) then
      for each  $X_k$  in NEIGHBORS[ $X_i$ ] do
        add  $(X_k, X_i)$  to queue



---

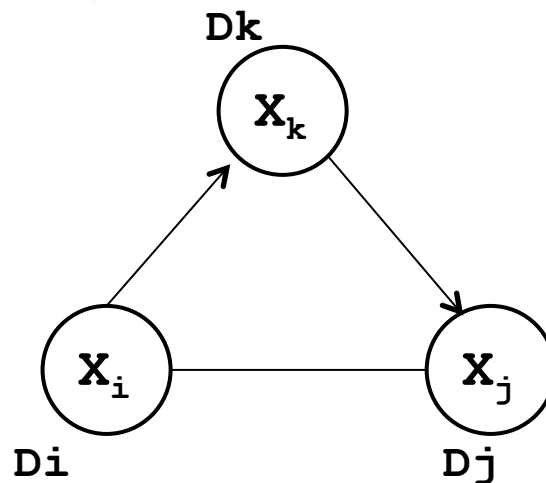

function RM-INCONSISTENT-VALUES( $X_i, X_j$ ) returns true iff remove a value
  removed  $\leftarrow$  false
  for each  $x$  in DOMAIN[ $X_i$ ] do
    if no value  $y$  in DOMAIN[ $X_j$ ] allows  $(x, y)$  to satisfy constraint( $X_i, X_j$ )
      then delete  $x$  from DOMAIN[ $X_i$ ]; removed  $\leftarrow$  true
  return removed
```

PATH-CONSISTENCY: grado 3

- La consistenza di grado 3 si ottiene partendo da un grafo arc-consistente.

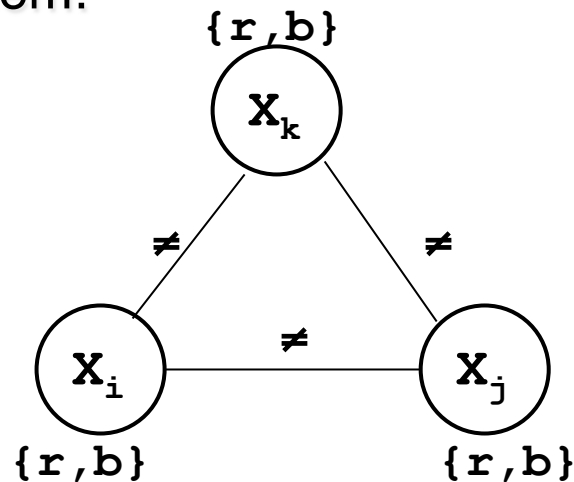
- PATH CONSISTENCY: Un cammino tra i nodi (i,j,k) è path consistente se, per ogni valore $x \in D_i$, e $y \in D_j$ (che rispettano la node e la arc-consistenza) esiste un valore $z \in D_k$ che soddisfa i vincoli $P(i,k)$ e $P(k,j)$.

(la consistenza del vincolo unario $P(k)$ è garantita dalla node consistenza della rete)



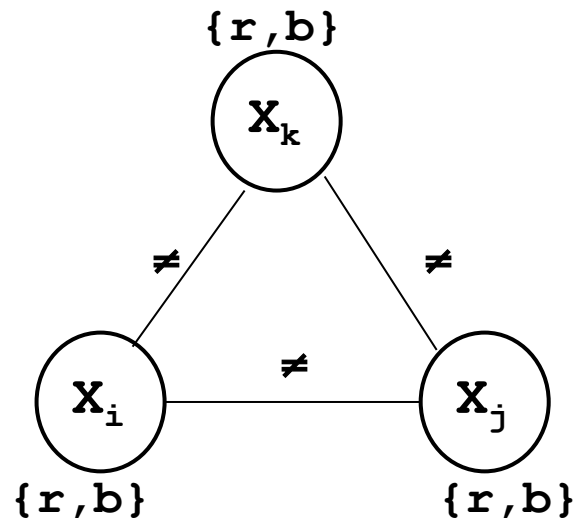
PATH CONSISTENCY:ESEMPIO1

- Supponiamo di considerare la rete di vincoli relativa a una istanza del map coloring problem:



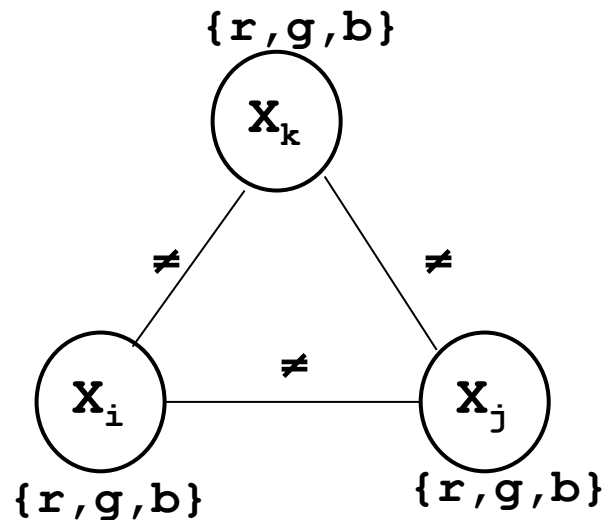
- Questa rete è arc-consistente: infatti, per ogni valore di ciascun dominio, esiste almeno un valore in ogni altro dominio che soddisfa il vincolo esistente tra i due nodi.
- Tuttavia, è immediato verificare che la rete non presenta soluzioni -> non è path-consistente: presi i valori $r \in D_i$ e $b \in D_k$, non esiste nessun valore appartenente a D_j che soddisfi contemporaneamente i vincoli $P(i,j)$ e $P(j,k)$.

PATH CONSISTENCY:ESEMPIO1 (*cont.*)



- Strutture dati aggiuntive, per ciascuna coppia di variabili (per ciascun arco) si memorizzano le coppie di valori compatibili per i quali esiste nel dominio della terza variabile un valore compatibile (esempio: per l'arco da X_i a X_j , nessuna delle coppie $\langle r, b \rangle$ e $\langle b, r \rangle$ ha un valore a supporto nel dominio di X_k)
- Verificare, per questo esempio, che la rete non è PC equivale alla situazione di *fallimento* nella ricerca

PATH CONSISTENCY:ESEMPIO1 (*cont.*)



- Aggiungendo un colore alla palette disponibile: r,g,b
- La rete è PC
- Verificare la consistenza di grado $n=3$ per questa rete che ha esattamente $n=3$ variabili equivale a verificare che *esiste una soluzione* (Nota bene: poiché il grado di consistenza è identico al numero di variabili)

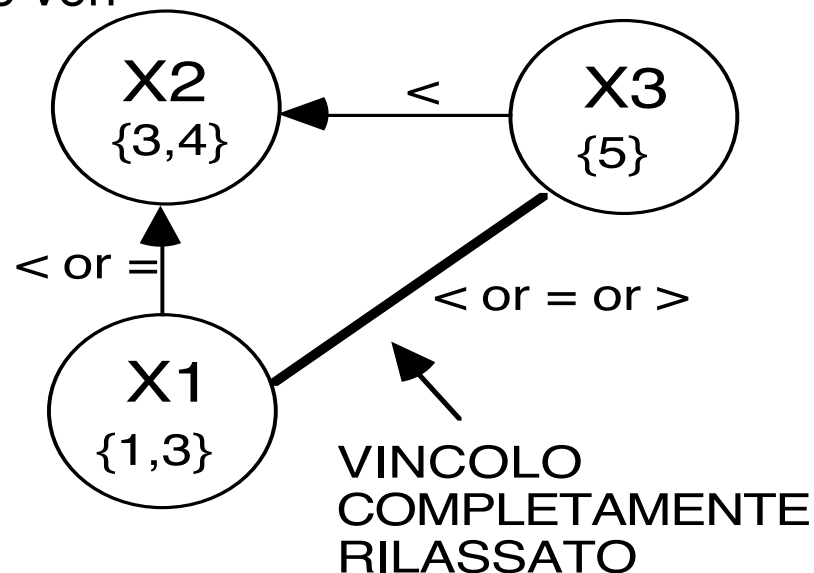
PATH-CONSISTENCY

- A network is path consistent iff any pair allowed by any direct relation R_{ij} is also allowed by all paths from X_i to X_j . A pair is allowed by a path from X_i to X_j if at every intermediate vertex (X_k) values can be found that satisfy the unary and binary predicates along the path.

PATH-CONSISTENCY

- La definizione originale di path-consistenza si riferisce **a cammini di lunghezza arbitraria m.**
- Tuttavia, esiste il seguente *teorema*:
- *Se ogni cammino di lunghezza 2 di un grafo completo è path-consistente, allora l'intera rete è path-consistente.*

Un constraint-graph non completo può essere reso completo aggiungendo, tra le variabili non vincolate, vincoli completamente rilassati: sempre veri



Esempio case

- Slide 8-9-10

K-CONSISTENZA

- Scelti valori per ogni $k-1$ -pla di variabili consistenti con i vincoli imposti dal problema, si cerca un valore per ogni k -esima variabile che soddisfa i vincoli tra tutte le k variabili. Se tale valore esiste allora le k variabili sono k consistenti.
- In generale, se un grafo contenente n variabili è k - consistente con $k < n$, allora per trovare una soluzione è necessaria una ricerca nello spazio restante.
- Freuder nel 1978 ha definito un algoritmo generale per rendere una rete k -consistente con k qualunque.
- Se un grafo contenente n variabili è n consistente, allora si può trovare una soluzione senza ricerca.
- Tuttavia, rendere una rete di vincoli contenente n variabili n -consistente ha una complessità esponenziale in n (*costa quanto effettuare la ricerca sul problema originale*)

CONSTRAINT SOLVER in pratica

- I più diffusi *constraint solver* fanno uso delle tecniche viste fino ad ora. In Constraint Programming tipicamente si usa un algoritmo di *labeling* con le euristiche viste ed *arc-consistency* per la propagazione di vincoli su variabili non istanziate.
- I vincoli sono visti come componenti software che incapsulano un algoritmo di *filtering*. Molto spesso l'algoritmo di *filtering* che effettua la propagazione non è *general purpose* come quelli visti finora, ma si basa sulla semantica del vincolo per motivi di efficienza.
- Esempio: $X::[1..10]$, $Y::[1..10]$, $X > Y$. Non importa controllare tutti i valori nei due domini, ma basta controllare i *bound*. In particolare questo vincolo è AC se $\min(X) > \min(Y)$ and $\max(X) > \max(Y)$
- Esempio $X::[1..10]$, $Y::[1..10]$, $X \neq Y$. Questo vincolo è SEMPRE AC se i due domini contengono entrambi più di un valore. L'AC si controlla quando una delle due variabili viene istanziata a un valore. Questo valore viene rimosso dal dominio dell'altra.

CONSTRAINT SOLVER in pratica

- Una caratteristica fondamentale dei constraint solver è la presenza di vincoli n-ari, anche detti vincoli *globali*
- Anche essi hanno un algoritmo di *filtering* al loro interno
- Chiaramente per raggiungere la consistenza di un vincolo n-ario (Generalized Arc Consistency), in linea di principio, si dovrebbe applicare la n-consistenza, che ha complessità esponenziale.
 - Tuttavia esistono vincoli particolari per cui raggiungere la n-consistenza ha costo polinomiale (ad esempio **alldifferent**)
 - Per gli altri ci si accontenta di un' approssimazione della Generalized Arc Consistency
- Si presentano alcuni vincoli globali nel corso di
→ **Constraint Processing**

CSP e Problemi di Ottimizzazione

- Abbiamo considerato solo CSP in cui le variabili hanno domini discreti. La maggior parte di questi problemi sono *NP-difficili*, cioè sono problemi per i quali *non è ancora stato trovato*, e probabilmente non esiste (ma non lo si è dimostrato), *un algoritmo in grado di trovare la soluzione in un tempo polinomiale* nella dimensione del problema.
- Un Constraint Optimization Problem (COP) è un Problema di Soddisfacimento di Vincoli in cui viene aggiunto un **obiettivo**. Un COP è quindi formalmente descrivibile come un CSP il cui scopo non è solo trovare una soluzione ammissibile, ma la soluzione **ottima** secondo un certo criterio di valutazione.

CSP e Problemi di Ottimizzazione

- Dato un algoritmo generale in grado di risolvere qualsiasi CSP si può allora utilizzare tale algoritmo per risolvere anche qualsiasi COP. Infatti, dopo aver descritto il problema in termini di variabili, domini e vincoli, basta aggiungere una variabile ulteriore che rappresenta la *funzione obiettivo*.
- Ogni volta che si trova una soluzione al CSP viene aggiunto un nuovo vincolo che garantisce che ogni soluzione futura avrà un valore della funzione obiettivo migliore. Questo procedimento continua finché non sarà più possibile trovare alcuna soluzione.
- L'ultima soluzione trovata è la soluzione ottima.

Tecniche Alternative

- Programmazione Lineare

- Tecniche numeriche per risolvere sistemi di equazioni (o disequazioni) lineari + ottimizzazione

- Es.: algoritmo del simplesso

→ *Ricerca Operativa*

- Lavora MOLTO bene per vincoli lineari

$$4*X - 3*Y > Z + 2$$

YES !

$$X^3 - 3*Y > Z^2 + 2$$

NO !

- Lavora , non molto bene, per problemi discreti
- In questi casi : **Constraint Processing** è una scelta migliore
- Anche: per problemi di vincoli non su dati numerici (domini finiti)