ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Problemi come soddisfacimento di vincoli

Federico Chesani

DISI

Department of Informatics – Science and Engineering

Disclaimer & Further Reading

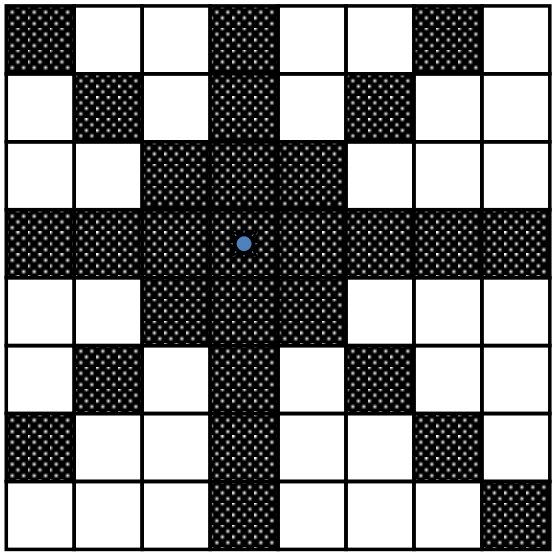
- These slides are largely based on previous work by Prof. Paola Mello

- Russell Norvig, AIMA, vol. 1 ed. italiana:
  - Cap. 6.1, 6.2, 6.3

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.

CSP su domini finiti

Formalmente un **CSP (Constraints Satisfaction Problem)** può essere definito su un insieme finito di variabili:

- Un insieme di variabili $(X_1,X_2,...,X_n)$
- i cui valori appartengono a domini finiti di definizione $(D_1,D_2,...,D_n)$
- e un insieme di vincoli.

Un vincolo $c(X_{i1},X_{i2},...X_{ik})$ tra k variabili è un sottoinsieme del prodotto cartesiano $D_{i1} \times D_{i2} \times ... \times D_{ik}$ che specifica quali valori delle variabili sono compatibili con le altre.

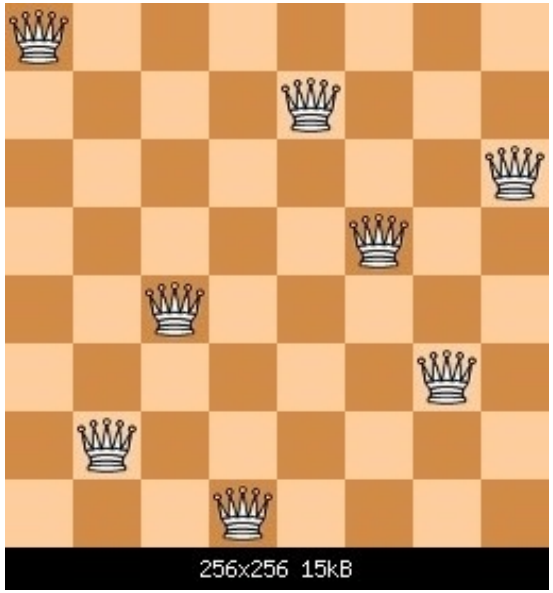
- Tale sottoinsieme non deve essere definito esplicitamente ma è rappresentato in termini di relazioni, che per semplicità considereremo unarie o binarie.

Una soluzione ad un CSP prevede un assegnamento di tutte le variabili che soddisfi tutti i vincoli.

Rompicapo 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: 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 è posizionata una regina, se il valore è 0 la posizione è libera. Dominio di possibili valori: [1,0].

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

Problema delle 8 Regine: Modello 2 (quello che adotteremo)

- Le otto regine vengono rappresentate con le variabili

  $X_1, X_2, ..., X_8$

  Il pedice si riferisce alla colonna occupata dalla corrispondente regina.

- L'istanziazione 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.

Modello 2 – Vincoli: Due Tipi

1. Vincoli riguardo i valori delle variabili e il dominio suddetto
2. Vincoli 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

Vincoli: Due Tipi (continua)

- $1 \leq Xi \leq 8$ per $1 \leq i \leq 8$
- $Xi \neq Xj$ per $1 \leq i < j \leq 8$
- $Xi \neq Xj + (j-i)$ per $1 \leq i < j \leq 8$
- $Xi \neq Xj - (j-i)$ 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

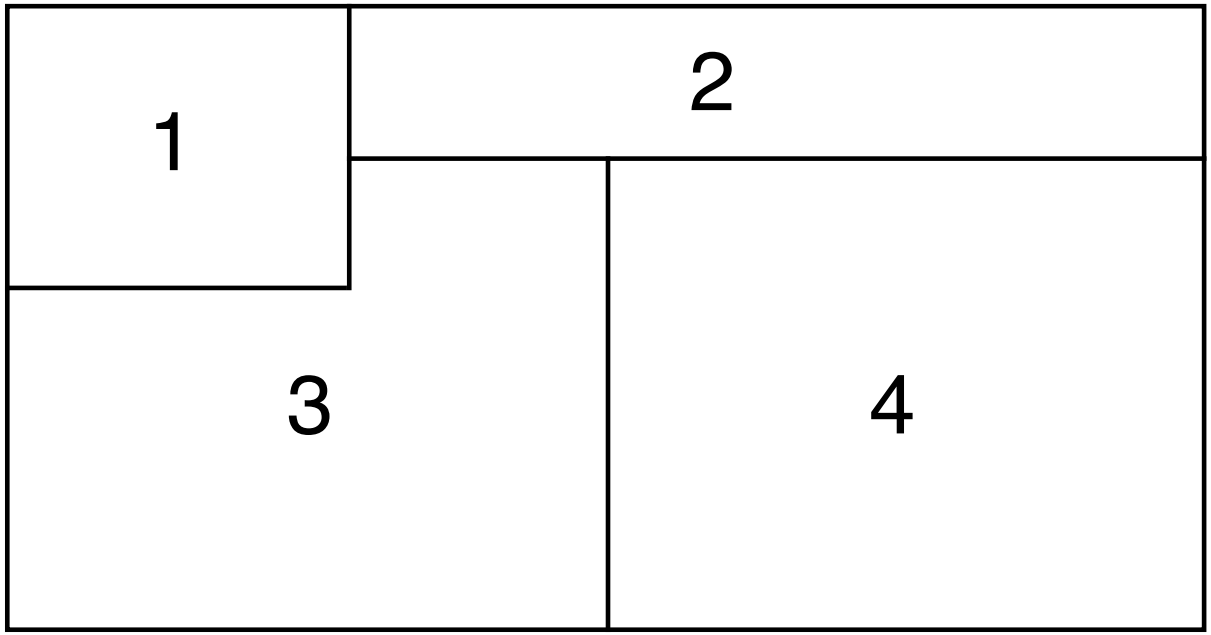
Scheduling: assegnare azioni (con una certa durata) a risorse per un certo tempo (le risorse possono essere condivise).

- 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:
  - Start1 + d1 ≤ Start2 **or** Start2 + d2 ≤ Start1
- Eventualmente altri dipendenti dal problema, quali ordinamenti fra attività.

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
- Domini: colori permessi
- Vincoli: Regioni adiacenti devono avere colori diversi.

Esempio: 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.

Esempio: Sudoku come CSP

- La tavola è suddivisa in 9 quadranti di 3x3 caselle
- Alcune caselle sono già fissate, le altre vanno riempite con numeri dall'1 al 9
- 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…).
- Vedrete vincoli speciali (da es Alldifferent) nel corso di Intelligent Systems.

Esempio: Sudoku come CSP

		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		

Risolvere un CSP come Ricerca nello Spazio degli Stati

CSP:

- stato 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 possono essere assegnamenti di valori a variabili (labeling)

- Linguaggio di formalizzazione generale
- Algoritmi general-purpose per la soluzione.

Possibile Algoritmo di Ricerca per CSP

- Stato iniziale: assegnamento vuoto { }
- Funzione Successore: assegna un valore ad una variabile non ancora legata (in modo che sia legale con gli assegnamenti già fatti).
  → fallisci se non esiste
- Goal test: l'assegnamento è completo (tutte le variabili sono legate).

Nota:

- Schema identico per tutti i CSPs
- Profondità limitata a n se n sono le variabili.
  → usa depth-first search
- La strada è irrilevante.
- Problema commutativo con d^n foglie (se d è la cardinalità dei dominii)

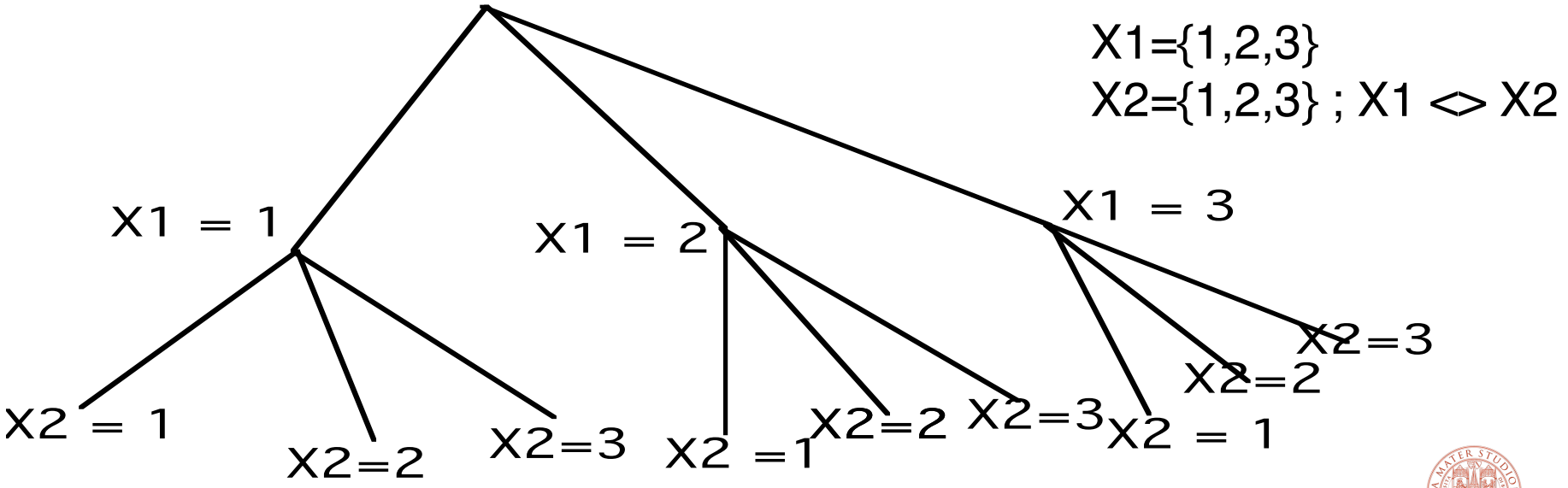
Albero Decisionale

- 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
  - ad ogni nodo corrisponde 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.

Esempio

- Per 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). Qui 3^2



Algoritmo di Soluzione di un Csp

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

Il terzo grado di libertà è ciò che differenzia le diverse strategie:

Algoritmi senza propagazione:

- **Generate and Test**
- **Standard Backtracking**

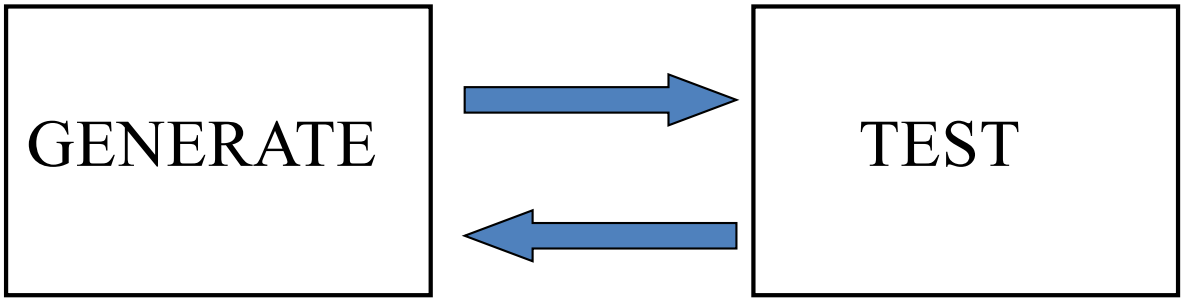
Algoritmi di Propagazione:

- **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).

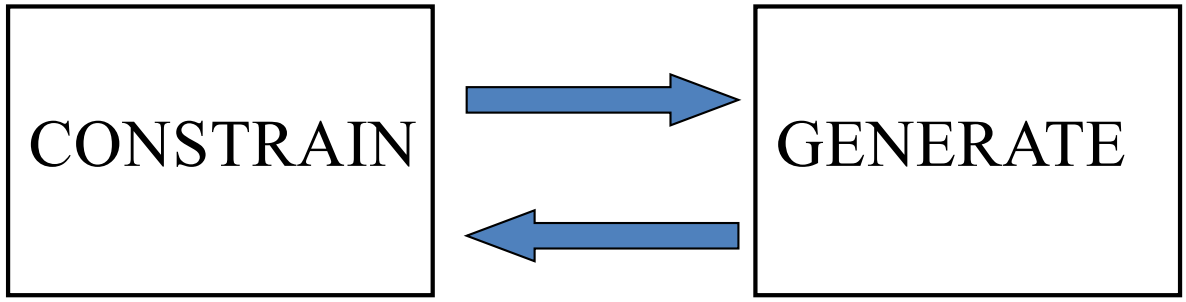


Algoritmi di Propagazione

Gli algoritmi di propagazione si basano sul concetto inverso.

- Forward Checking (FC)
- 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).

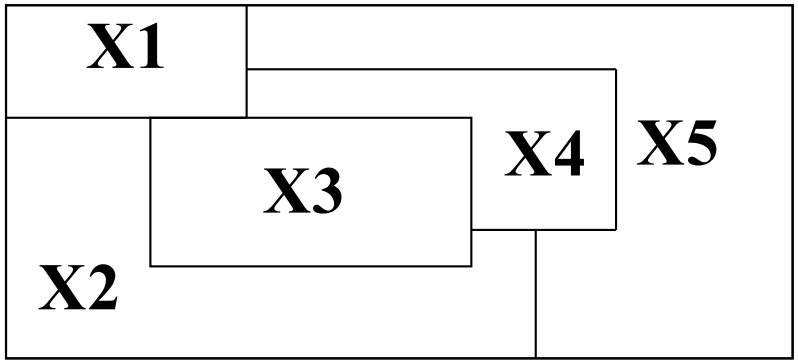


Generate and Test

- 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:: [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 45 foglie

Generate and Test per 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.

- 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.

Unici vincoli considerati nella fase Generate:

- $1 \leq Xi \leq 8$ per $1 \leq i \leq 8$
- $Xi \neq Xj$ 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.

Generate and Test

Così facendo un tentativo di soluzione al problema può essere la seguente:

- $(X_1, X_2, ..., X_8) = (1,2,3,4,5,6,7,8)$. 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 1018 e quindi dimensioni inaccettabili dallo spazio di ricerca.

Standard Backtracking

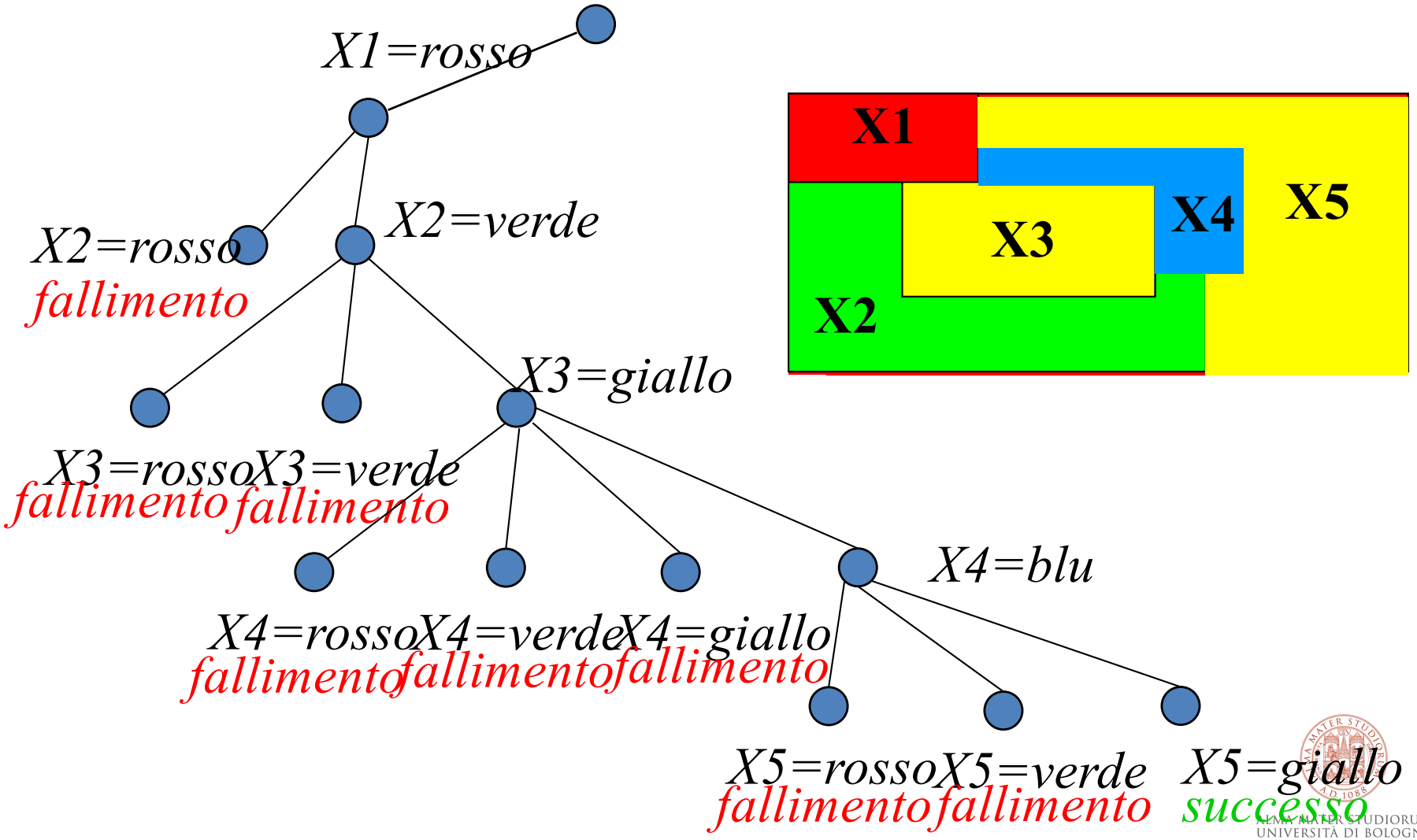
Sebbene migliore della precedente anche questa tecnica prevede un utilizzo a posteriori dei vincoli

- A ogni istanziazione di una variabile si preoccupa di verificare 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 delle contraddizioni.

Standard Backtracking per Map Coloring

Algoritmo semplice:



Standard Backtracking per 8 Regine

- 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.

- Infatti il vincolo $X_i \neq X_j - (j-i)$ per $1 \leq i < j \leq 8$ verrebbe violato dalle prime due variabili: sostituendo X1 = 1 e X2 = 2 nella relazione si ottiene $1 \neq 2 - (2-1)$
  che porta a una contraddizione.

- L'algoritmo viene quindi fermato e, con un backtracking, si tenta di assegnare a X2 il valore 3 con successo e così via.

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 ≠ failue then return result
            remove { var = value } from assignment
    return failure
```

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.

- A differenza della metodologia del Generate and Test, che considera il controllo dei vincoli del problema solo al termine dell'istanziazione di tutte le variabili, lo Standard Backtracking controlla la loro consistenza ad ogni istanzazione di variabile.

Esempio Standard Backtracking: le 8 Regine

Nella risoluzione del problema delle otto regine supponiamo di avere già istanziato sei variabili ai valori:

- $(X_1, X_2, X_3, X_4, X_5, X_6) = (1,3,5,7,2,4)$.
- L'assegnazione $X_1=1$ è la prima scelta fatta.
- Assegnare $X_2= 1$ viola il vincolo sulla riga (Backtracking)
- $X_2 = 2$ viola i vincolo di diagonale (Backtracking)
  - $V_{k+1} \neq V_i - (k + 1-i)$ to $1 \leq i \leq k$.
- Quindi viene assegnato $X_2= 3$ che soddisfa tutti i vincoli:
  - $3 \neq 1$;
  - $1 \leq 3 \leq 8$;
  - $3 \neq 1 + (2 - 1)$;
  - $3 \neq 1 - (2-1)$.
- Per la terza variabile il valore 1 viola il vincolo di riga, il valore 2 il vincolo in diagonale con la seconda variabile e infatti $2 \neq 3 - (3-2)$ non è soddisfatto. Il valore 3 viola invece il vincolo di riga con la seconda variabile.

Esempio: 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.
- Si tenta allora l'istanziazione $X_3=5$ che ha successo verificando i vincoli:

  $5 \neq 1 ; 5 \neq 3; \qquad 1 \leq 5 \leq 8 ;$

  $5 \neq 1 + (3 - 1) ; 5 \neq 3 + (3 - 2) ;$

  $5 \neq 1 - (3 - 1) ; 5 \neq 3 - (3 - 2) .$
- Procediamo ora all'istanziazione della quarta variabile al valore $X_4=7$ che soddisfa i vincoli:

  $7 \neq 1 ; 7 \neq 3 ; 7 \neq 5 ; \ 1 \leq 7 \leq 8 ;$

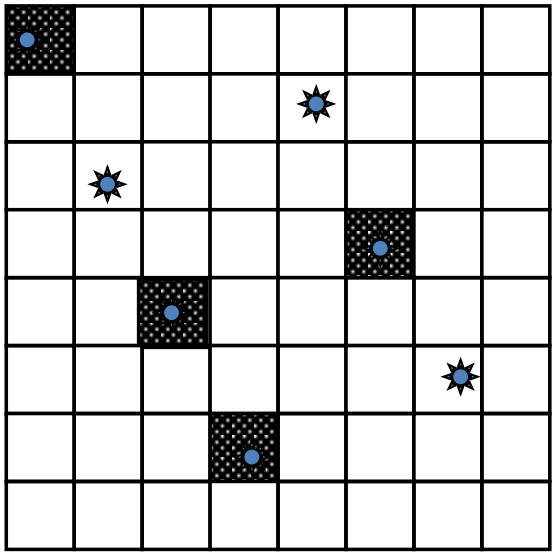
  $7 \neq 1 + (4 - 1) ; 7 \neq 3 + (4 - 2) ; \ 7 \neq 5 + (4 - 3)$

  $7 \neq 1 - (4 - 1) ; 7 \neq 3 - (4 - 2) ; \ 7 \neq 5 - (4 - 3)$
- 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 8 Regine

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 $X_1, X_2, \ldots\ldots, X_6$
- procede all'istanziazione dell'ultima regina.



Problema

- L'ultima colonna, corrispondente alla variabile X_8, non ha più posizioni disponibili assegnabili alla regina.

- L'algoritmo solo dopo aver istanziato anche X_7 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, non presenta più posizioni disponibili.

Limiti dell'uso A Posteriori dei Vincoli

- Questo è un difetto da attribuire a tutti i metodi che utilizzano i vincoli passivamente cioè posteriormente ad un tentativo di istanziazione.

- 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.

Gli Algoritmi di Propagazione

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

- **Pruning a priori dell'albero delle decisioni**.

- Idea: 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

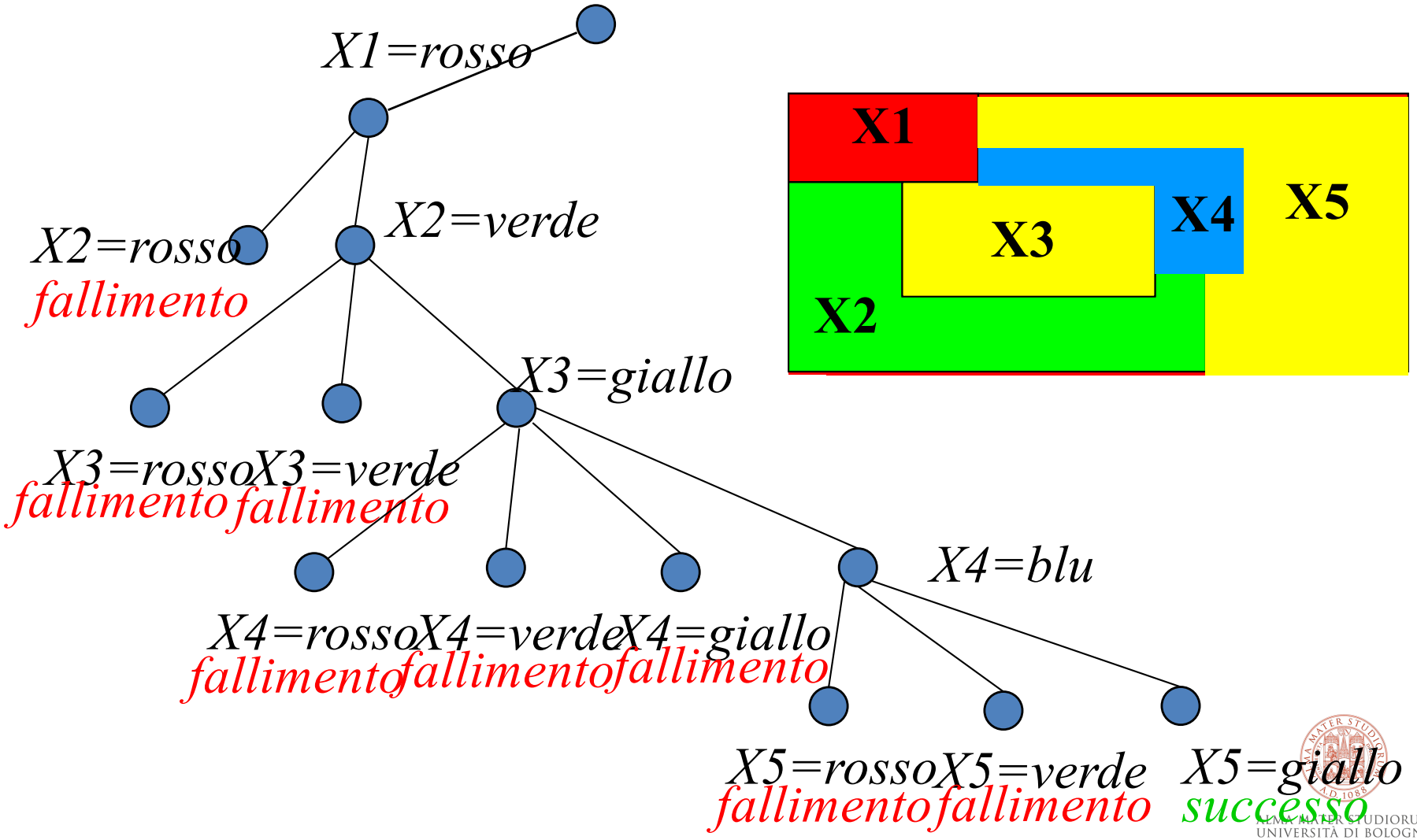
- **Forward Checking (FC)**
- **Partial Look Ahead (PLA)**
- **Full Look Ahead (FLA)**

- 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 dominii delle variabili future (**se un dominio risulta vuoto, fallimento**)

Standard Backtracking per Map Coloring (rivediamolo)

Algoritmo semplice:



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

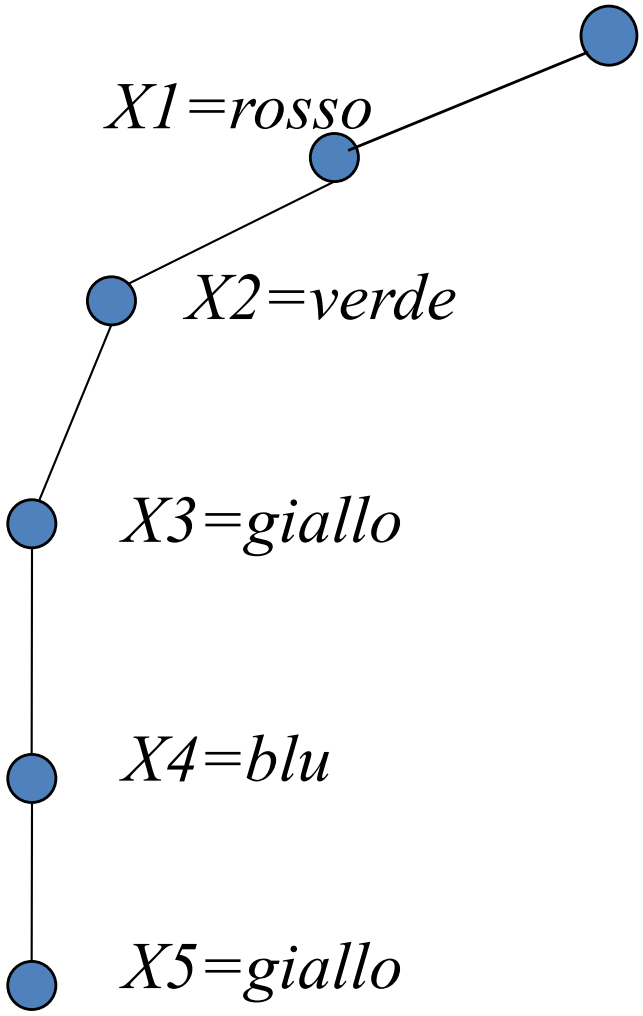
Supponiamo X2 ultima variabile oggetto di *labeling:*

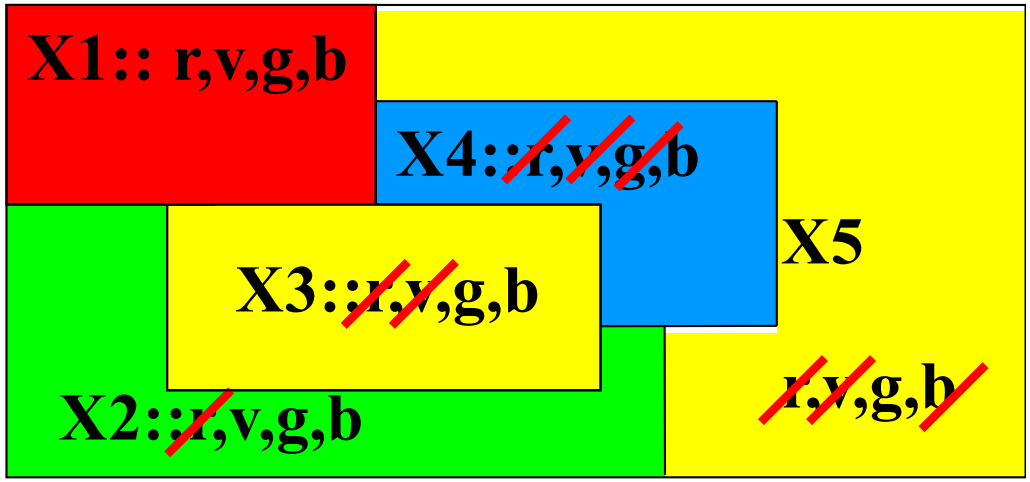
- Standard Backtracking: controlla c(X2,X1)
- Forward Checking:
  - controlla c(X2,X3) per ogni valore in D3
  - controlla c(X2,X4) per ogni valore in D4
- Sono eliminati i valori da D3 e D4 incompatibili con l' assegnamento di X2

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

Forward Checking per Map Coloring

Eliminazione a priori dei valori inconsistenti dai dominii delle variabili future





Forward Checking: 8 Regine

- L'esempio che porta all'assegnazione dei valori (1,3,5,7,2,4) alla sequenza di variabili (X_1,.....,X_6).
- Inizialmente l'insieme dei domini di tutte le variabili contiene gli interi compresi tra 1 e 8.

- Dopo l'istanziazione di X_1 al valore 1 vengono eliminati, dagli insiemi relativi a X_2,.....X_8, tutti i valori incompatibili con questo. Ovviamente il valore 1 viene tolto da tutti i domini a causa del vincolo

  $X_1 \neq X_i$ per i =2,...,8.

- Dopo l'istanziazione del valore 1 alla variabile X_1, vengono ridotti gli insiemi dei valori ammissibili per le variabili ancora libere. Infatti X2 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.

Forward Checking: 8 Regine (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).

Forward Checking: 8 Regine (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.

- Si procede all'istanziazione 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.

Forward Checking: 8 Regine (cont.)

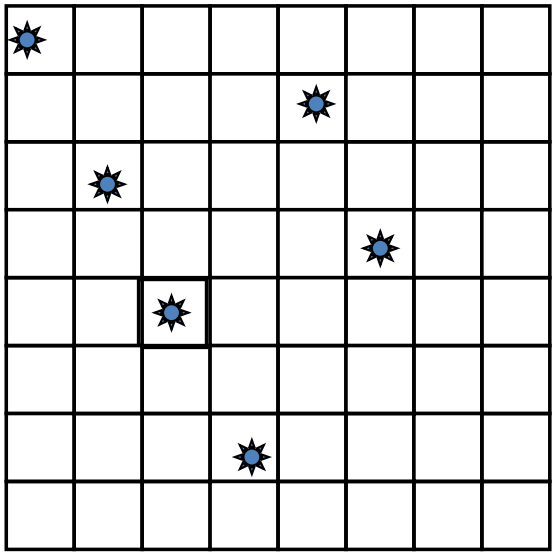
- Il backtracking porta alla ritrattazione del valore assegnato all'ultima variabile istanziata, della propagazione causata da quest'ultimo e al successivo tentativo con il valore X_4=7.

- 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'istanziazione della variabile X_5 al valore 2 e X_6 al valore 4 eliminando ogni possibile istanziazione per la variabile X_8.

Forward Checking: 8 Regine (cont.)



Poi si procede all'istanziazione della variabile X5 al valore 2 e X6 al valore 4 eliminando ogni possibile istanziazione per la variabile X8.

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

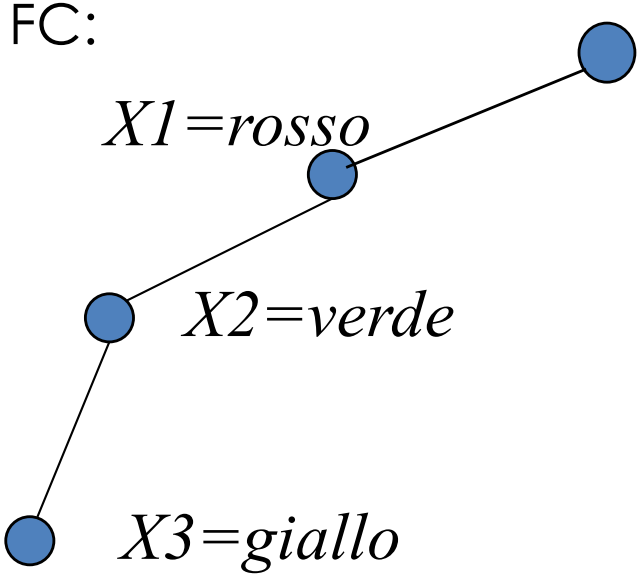
Strategia di Partial Look Ahead (PLA) o Full Look Ahead (FLA). Supponiamo venga assegnata la variabile X_k.

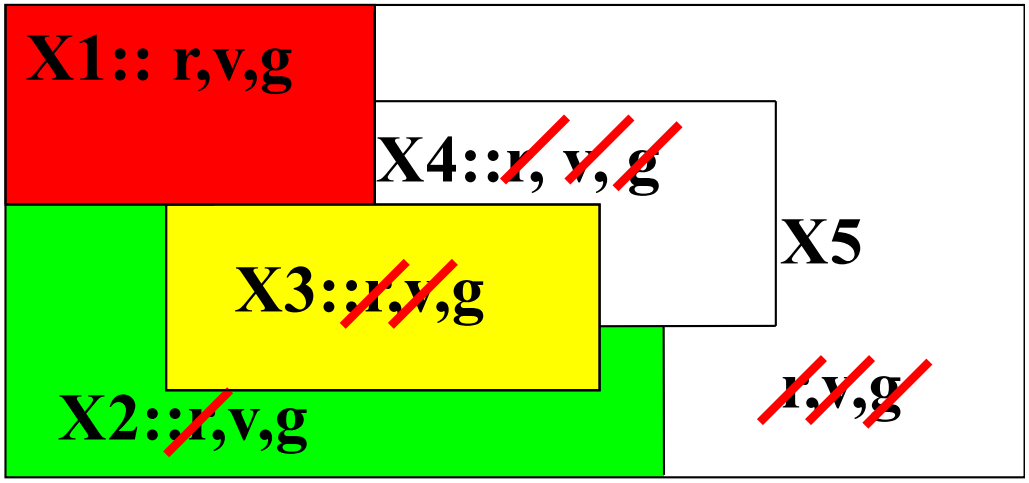
- **PLA**: Per ogni variabile X_h non ancora istanziata si ha una propagazione dei vincoli contenenti la variabile X_h e le variabili "future", ossia le variabili X_{h+1},..., X_n
  - Per ogni variabile non ancora assegnata X_{k+1},.....,X_n, deve esistere un valore per il quale sia possibile trovare, 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},...X_{h-1},X_{h+1}..,X_n.
  - per ogni variabile non ancora assegnata X_{k+1},.....,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.

FC vs PLA per Map Coloring (cont.)

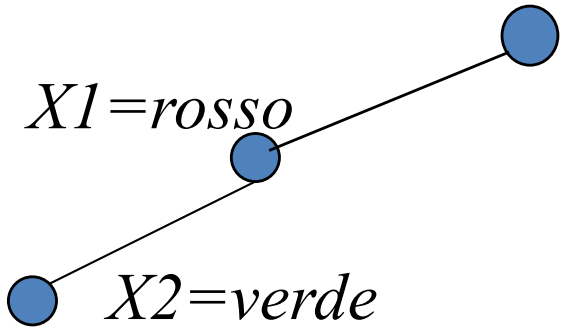
PLA rispetto a FC non aumenta il pruning dell'albero nel caso precedente, ma per una istanza con tre soli valori di dominio (r,v,g), il FC:



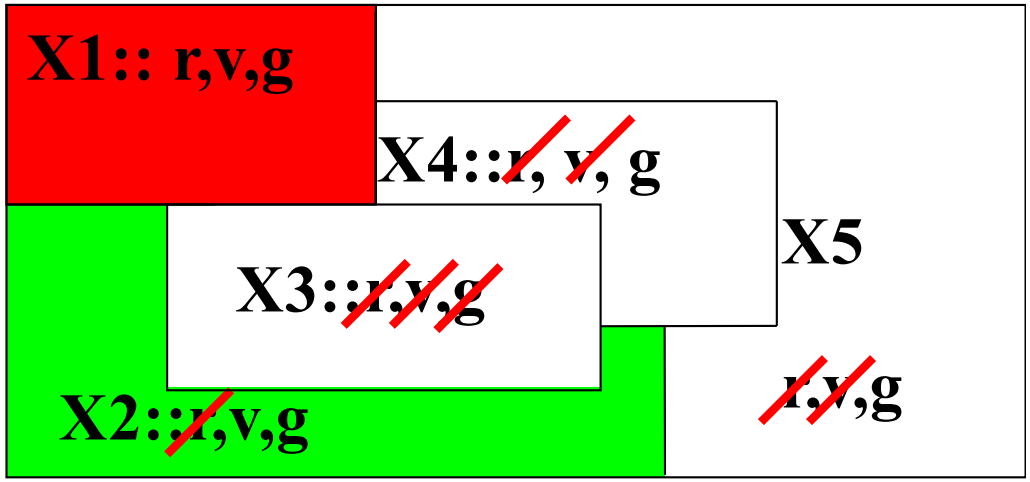


FC vs PLA per Map Coloring (cont.)

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



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)

Look Ahead: Esempio

CSP: X0 < X1 < X2 < X3 con domini per X1, X2, X3::[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 e 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 e in D2)

Look Ahead: Esempio (cont.)

CSP: X0 < X1 < X2 < X3 con dominii X1,X2,X3 :: [1,2,3]

- PLA X1 :: [1,2]
  X2 :: [1,2]
  X3 :: [1,2,3]

- FLA X1 :: [1,2]
  X2 :: [2]
  X3 :: [3]

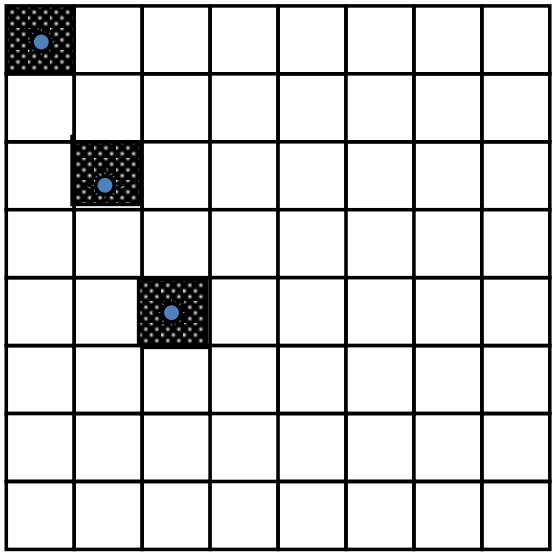
Riduce ulteriormente i dominii delle variabili. In particolare, elimina il valore 1 dal dominio di X2 (non c'è supporto in D1) e i valori 1 e 2 dal dominio di X3 (non c'è supporto in D1 e D2)

Esempio di PLA: le 8 Regine

Supponiamo di avere assegnato alle prime tre variabili X_1,X_2,X_3 rispettivamente i valori 1,3,5.

Domini restanti (identici a quelli ridotti dalla tecnica del Forward Checking):

- X4 è associato al dominio D4=(2,7,8),
- X5 è associato al dominio D5=(2,4,8),
- 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).



Esempio di PLA: le 8 Regine

X4 è associato al dominio D4=(2,7,8),
X5 è associato al dominio D5=(2,4,8),
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).

Eliminiamo quei valori per i quali non esiste, nei dominii associati alle variabili "future" almeno un posizionamento compatibile con il valore estratto. Quindi:

- Estraiamo da D_4 il valore 2. I valori 4 in D_5, 6 in D_7 e 4 in D_8 sono compatibili con il valore 2 di X_4 mentre il valore 4, unico assegnabile a X_6, non è compatibile. Dal dominio D_4 viene eliminato il valore 2.
- Estraiamo da D_4 il valore 7. I valori 2 in D_5, 4 in D_6, 6 in D_7 e 6 in D_8 sono compatibili con il valore 7 che pertanto non va eliminato dal dominio come, del resto, il valore 8. Il dominio associato a X_4 diventa pertanto: D_4=(7,8).

Esempio di PLA: le 8 Regine

- Si estrae da D_5 un valore e si controlla che esista, per D_6, D_7 e D_8, almeno un valore compatibile con questo. Per l'estrazione 2 in D_5 esistono i valori 4 in D_6, 6 in D_7 e 6 in D_8 che rispettano tutti i vincoli.
- Estraendo il valore 4 in 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:

  $D_5=(2,8)$.
- Gli insiemi D_6 e D_7 non vengono modificati.

- Evitiamo quindi l'assegnazione, tentata dal Forward Checking, alla variabile X_4 del valore 2 che falliva immediatamente a causa di D_6, che rimaneva vuoto.

Esempio di FLA: le 8 Regine

- Riprendiamo l'esempio precedente, supponendo di avere già propagato i domini con la tecnica del PLA e **applichiamo FLA**:
  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).
- 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.

- Nel dominio D_5, pertanto, resta il valore 2. Il valore 4 in D_6 risulta compatibile con 2 in D_5 e con i 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),

Esempio di FLA: le 8 Regine

- Analizziamo ora 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.

  D_7 risulta pertanto: D_7=(6)

- Il dominio D_8, 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 . La computazione, pertanto, fallisce senza tentare ulteriori assegnazioni.

- 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:

- **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.

Le euristiche possono essere classificate in:

- 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 Remaing Values) che scegliere 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**).

Classificazione delle Euristiche

Un'ulteriore classificazione è la seguente:

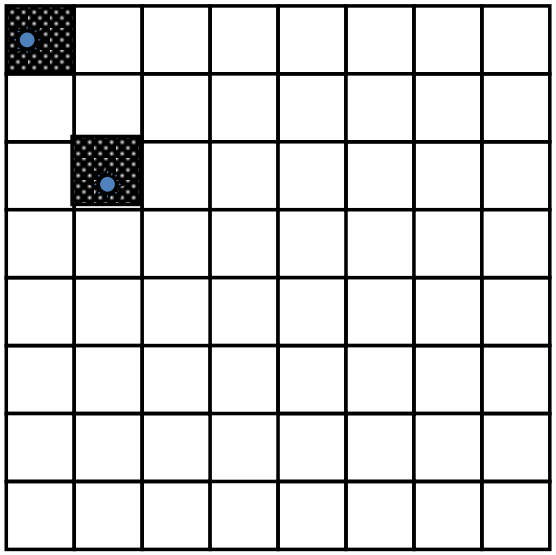
- **Euristiche statiche**: determinano l'ordine in cui le variabili (o i valori) vengono scelti prima di iniziare la ricerca; tale ordine rimane invariato durante tutta la ricerca.

- **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.

Forward Checking per 8 Regine (First Fail)

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'istanziazione 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 precedentemente illustrato delle 8 Regine per l'algoritmo Forward Checking dopo le istanziazioni:

  $X_1=1$, $X_2=3$, $X_3=5$

- 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).

  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).

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.

Senza perdita di generalità ci riferiremo, nel seguito, a CSP su vincoli binari (vincoli cioè che coinvolgono due variabili).

- **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.

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.

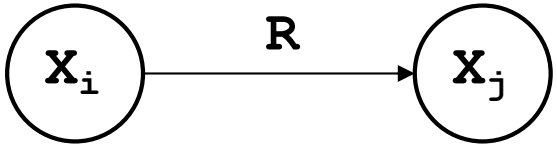
- Possono essere applicate staticamente oppure ad ogni passo di assegnamento (labeling) come potenti tecniche di propagazione per le variabili non ancora istanziate.

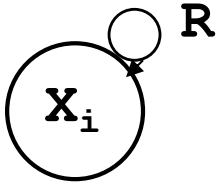
- Tutte le tecniche di consistenza sono basate su una **rappresentazione del problema come una rete (grafo) di vincoli**. Gli archi possono essere orientati o non orientati: ad esempio il vincolo > viene rappresentato da un arco orientato, mentre il vincolo ≠ da un arco semplice (non orientato o doppiamente 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.

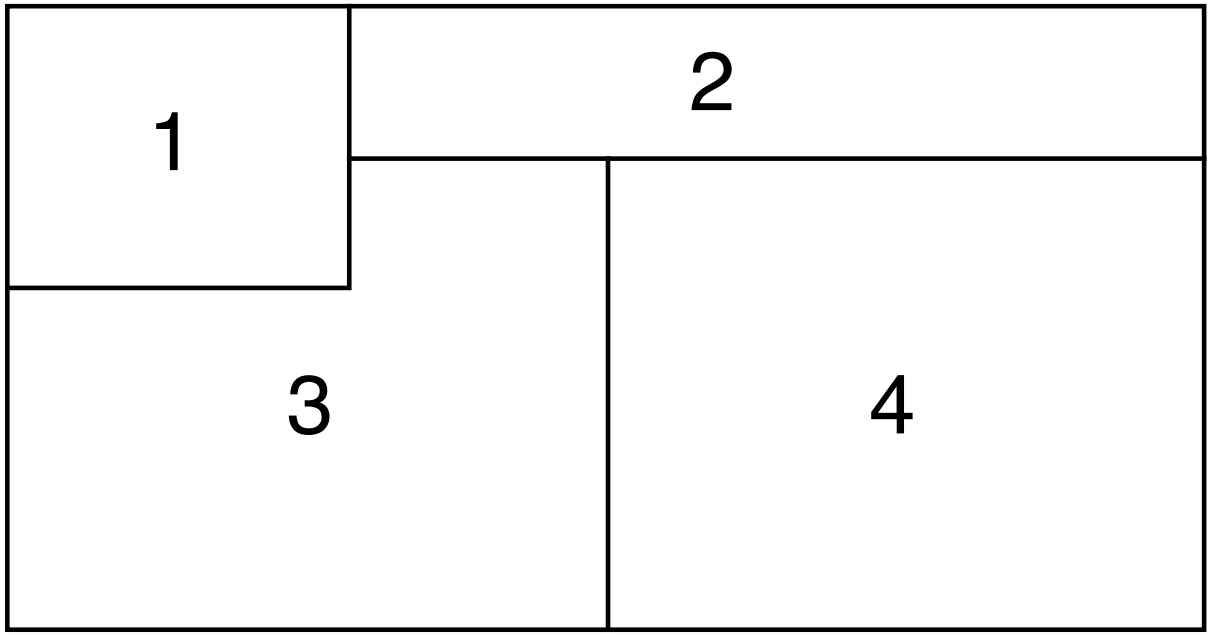
- I vincoli binari (R) collegano due nodi X_i e Xj:
- I vincoli unari sono rappresentati da archi che iniziano e terminano sullo stesso nodo X_i





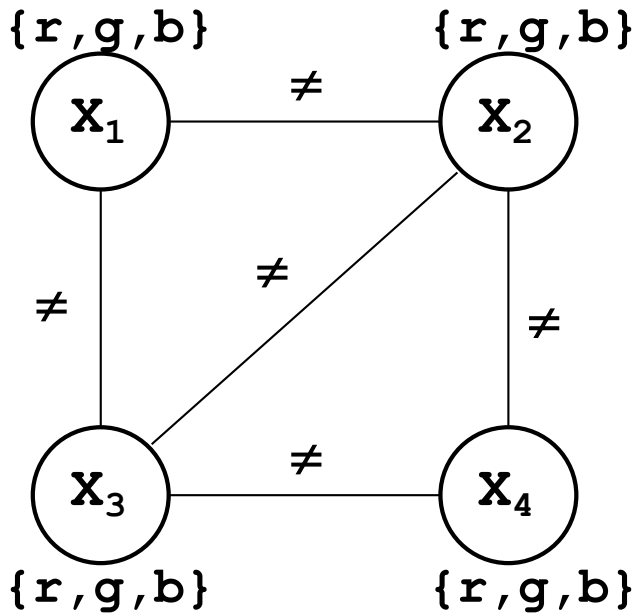
Esempio: Map Coloring Problem

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: X1=r, X2=r, X3=r, X4=r).
- Esistono diversi algoritmi che realizzano gradi diversi di consistenza.

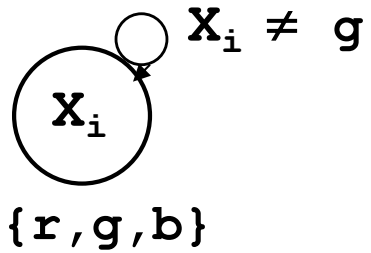


Node Consistency

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 è node consistent perchè il valore $g \in D_i$ viola il vincolo unario su X_i.
- Per rendere il nodo consistente è necessario eliminare dal dominio di X_i il valore g.

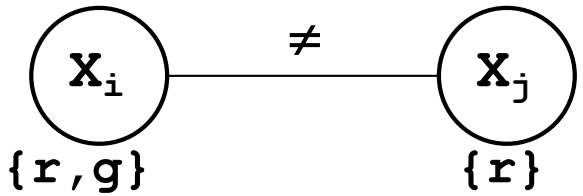


Un **grafo** è **node consistent** se tutti i suoi nodi sono consistenti.

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.

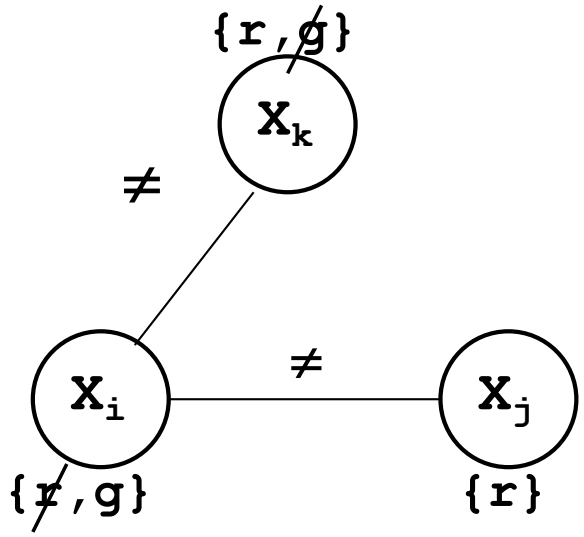
ARC CONSISTENCY: un **arco** A(i,j) è **consistente** se per ogni valore $X \in D_i$ esiste almeno un valore $Y \in D_j$ tale che il vincolo tra i e j P(i,j) sia soddisfatto



- 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 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

- 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**

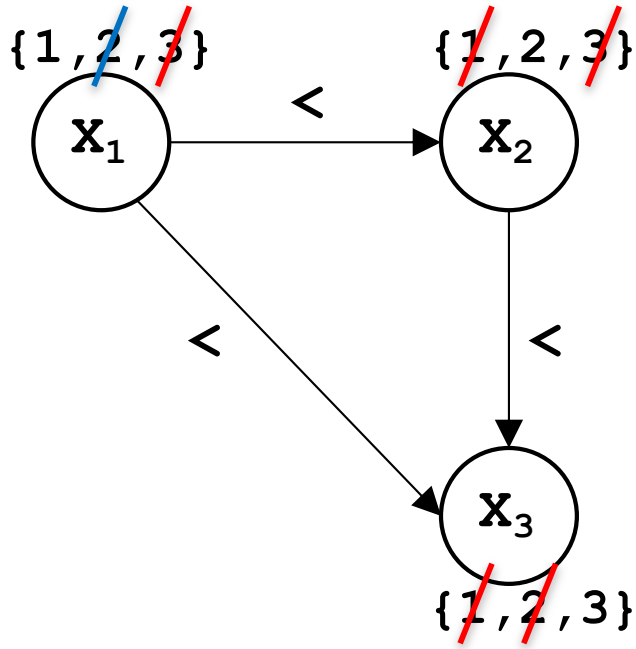


- La rimozione del valore $X_i = r$ a causa del vincolo ≠ tra X_i e X_j comporta la rimozione del valore $X_k = g$ a causa del vincolo tra X_i e X_k
- Quindi l'arc consistency è un procedimento iterativo che converge ad una rete stabile e arc-consistente.

Arc Consistency: Esempio (cont.)

- X1 < X2 < X3 con dominii X1,X2,X3 :: [1,2,3]

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



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

X1 :: [1,2]
X2 :: [2]
X3 :: [3]

Meno *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*), ciclando finché non è vuota
  - Non appena il dominio di una variabile Xi è ridotto, si riaggiungono a queue gli archi (Xk,Xi) per ciascuna variabile Xk collegata da un arco incidente su Xi.

Algoritmo AC-3 (Mackworth, 1977)

```
function AC-3( csp) returns the CSP, possibly with reduced domains
  inputs: csp, a binary CSP with variables {X_1, X_2, ..., 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) ← REMOVE-FIRST(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 ← 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 ← true
  return removed
```

Da: S. Russell & P. Norvig: "Intelligenza Artificiale: un approccio moderno", Pearson ed.

Path Consistency: grado 3

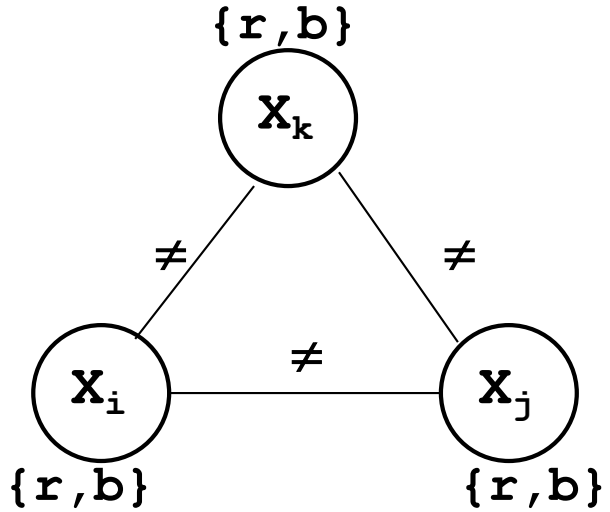
La consistenza di grado 3 si ottiene partendo da un grafo arc-consistente.

PATH CONSISTENCY: Un **cammino** tra i nodi (i,j,k) è **path consistent** se, per ogni valore $x \in D_i$, e $y \in Dj$ (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)

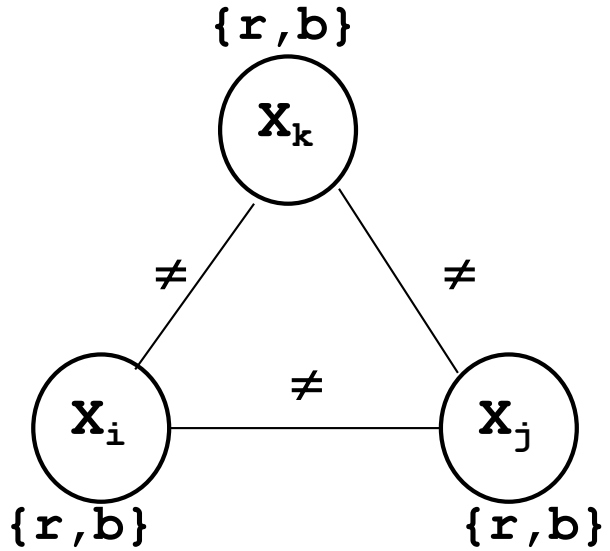
Path Consistency: Esempio1

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



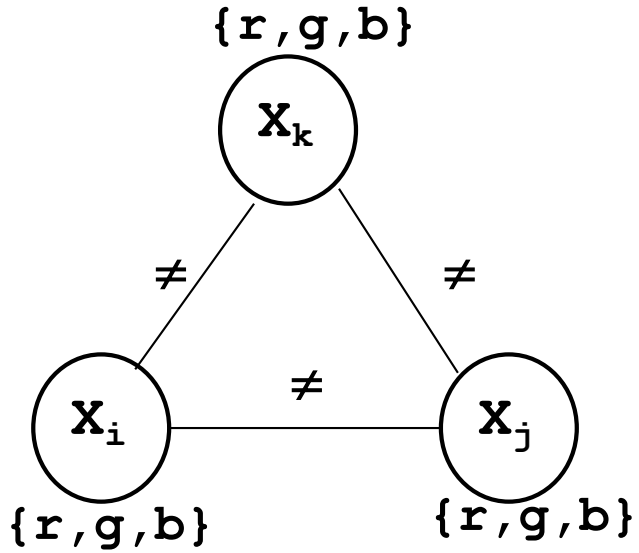
- Questa rete è arc-consistente: infatti, per ogni valore di ciascun dominio, esiste almeno un valore in ogni altro dominio che soddisfa il vincolo esistente tra i due nodi.
- Tuttavia, è immediato verificare che la rete non presenta soluzioni -> **non è path-consistente**: presi i valori $r \in D_i$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

Path Consistency: Esempio 1 (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 Xi a Xj, nessuna delle coppie <r,b> e <b,r> ha un valore a supporto nel dominio di Xk)
- Verificare, per questo esempio, che la rete non è PC equivale alla situazione di *fallimento* nella ricerca

Path Consistency: Esempio 1 (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)***

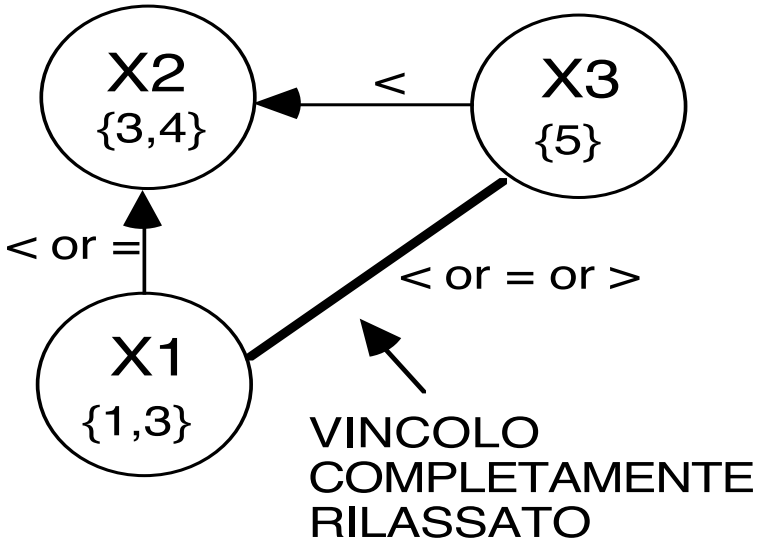
Path Consistency

Osservazione: la definizione originale di path-consistenza si riferisce a cammini di lunghezza arbitraria m.

Tuttavia, esiste il seguente teorema:

TEOREMA: **Se ogni cammino di lunghezza 2 di un grafo completo è path-consistent 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



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.

- Se un grafo contenente n variabili è n consistente, allora si può trovare una soluzione senza ricerca.
- **Freuder** nel 1978 ha definito un algoritmo generale per rendere una rete k-consistente con k qualunque.
- 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 ≠ 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
- *Vedrete alcuni vincoli globali nel corso di Intelligent Systems*

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, 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.