

PARTE II: Propagazione dei vincoli e Vincoli globali

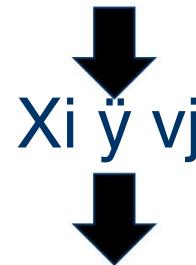
Risolutore di vincoli

- I Enumera tutti i possibili valori variabili combinazioni tramite una **ricerca sistematica di alberi backtracking**.
 - Ricerca sistematica nell'albero backtracking
 - indovina
 - Indovina un valore per ciascuna variabile.
- I Durante la ricerca, esamina i vincoli per **rimuovere i valori incompatibili (incoerenti)**.
 - dai
 - dai domini delle **variabili future (inesplorate)**, tramite propagazione.
 - Restringe i domini delle variabili future.

Ricerca e propagazione

I Le decisioni di ricerca e la propagazione sono interlacciate.

Propagazione



Le decisioni di ricerca e la propagazione sono interlacciate.

Propagazione



X_{i'} ŷ v_{j'}

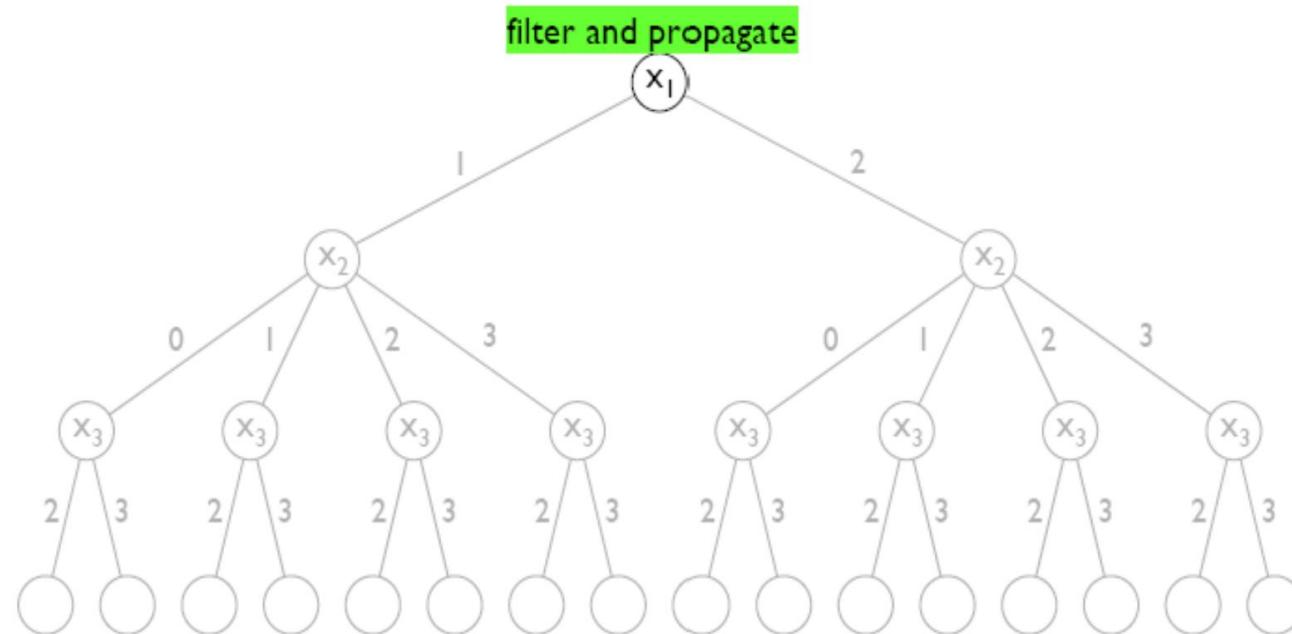


Propagazione

Backtracking nella ricerca e propagazione degli alberi

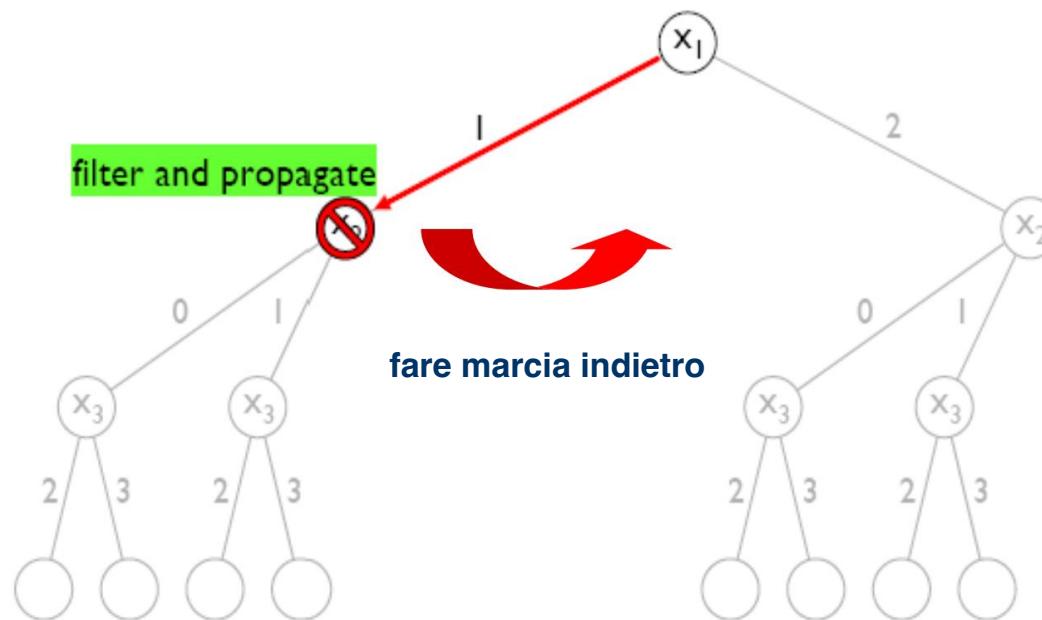
$| X_1 \in \{1,2\} \quad X_2 \in \{0,1,2,3\} \quad \cancel{X_3 \in \{2,3\}} | X_1$

$> X_2 \neq X_1 + X_2 = X_3 \text{ e tutti diversi } [X_1, X_2, X_3]$



Backtracking nella ricerca e propagazione degli alberi

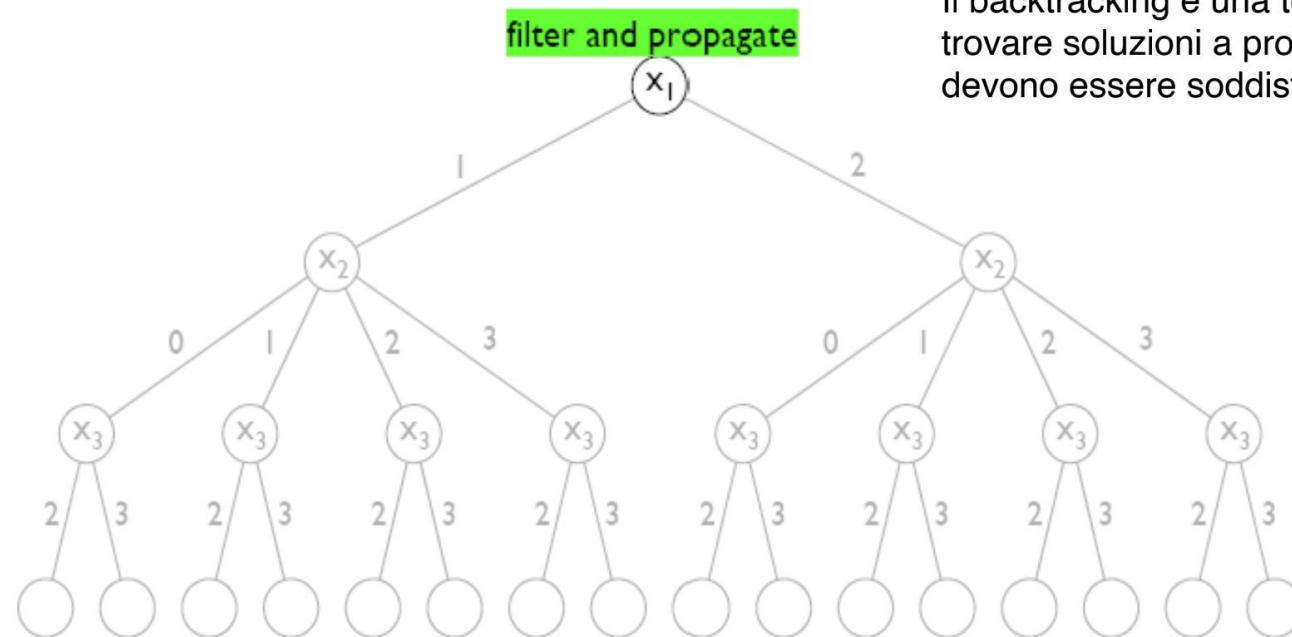
$| X_1 = 1, X_2 \in \{0,1\} X_3 \in \{2,3\} | / X_1$
 $> X_2 \neq X_1 + X_2 = X_3 \text{ e tutti diversi}([X_1, X_2, X_3])$



Backtracking nella ricerca e propagazione degli alberi

$| X_1 \in \{1,2\} \quad X_2 \in \{0,1,2,3\} / \cancel{X_3 \in \{2,3\}} | X_1$

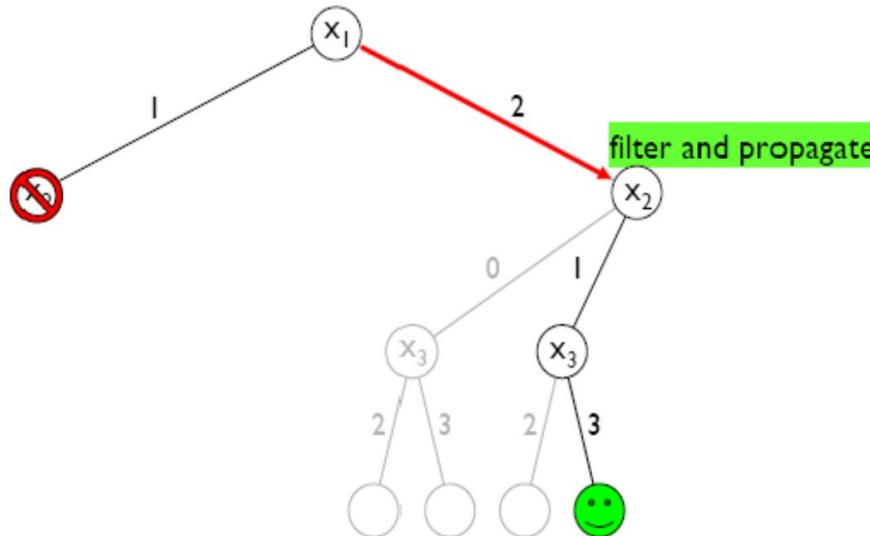
$> X_2 \neq X_1 + X_2 = X_3 \text{ e tutti diversi}([X_1, X_2, X_3])$



Il backtracking è una tecnica per trovare soluzioni a problemi in cui devono essere soddisfatti dei vincoli.

Backtracking nella ricerca e propagazione degli alberi

$| X_1 = 2 \quad X_2 \in \{0,1\} \quad X_3 \in \{2,3\} | / X_1$
 $> X_2 \neq X_1 + X_2 = X_3 \text{ e tutti diversi}([X_1, X_2, X_3])$



Contorno

I Coerenza locale

- Consistenza dell'arco generalizzata (GAC)
- Coerenza dei limiti (BC)

I Propagazione dei vincoli

- Algoritmi di propagazione

I Propagazione specializzata

- Vincoli globali

I Vincoli globali per scopi generici

Coerenza locale

rileva assegnazioni parziali incoerenti.

I Una forma di inferenza che rileva assegnazioni parziali incoerenti.

– Locale, perché esaminiamo i vincoli individuali.

Le coerenze locali più diffuse sono basate sul dominio:

I Le consistenze locali più diffuse sono basate sul dominio: – Generalized Arc

Consistency (GAC). I Chiamato anche Hyper-arc o

Domain Consistency; – Consistenza dei limiti (BC).
limiti

rilevano

– Rilevano assegnazioni parziali incoerenti della forma $X_i = j$,
quindi:

I j può essere rimosso da $D(X_i)$ tramite propagazione; I
la propagazione può essere implementata facilmente.

Consistenza dell'arco generalizzata (GAC)

- I Un vincolo **C** definito su **k** variabili $C(X_1, \dots, X_k)$ fornisce l'insieme delle combinazioni di valori consentite (ovvero le tuple consentite).
 - **C** $\subseteq D(X_1) \times \dots \times D(X_k)$
 - Ad esempio, $D(X_1) = \{0,1\}$, $D(X_2) = \{1,2\}$, $D(X_3) = \{2,3\}$ $C: X_1 + X_2 = X_3$

$$C(X_1, X_2, X_3) = \{(0,2,2), (1,1,2), (1,2,3)\}$$



Ogni tupla consentita $(d_1, \dots, d_k) \models C$ dove di \models X_i è un **supporto** per C .

GAC

I $C(X_1, \dots, X_k)$ è GAC se e solo se:

- per ogni X_i in $\{X_1, \dots, X_k\}$, per ogni $v \in D(X_i)$, v appartiene a a supporto per

C. I Chiamata Arc Consistency (AC) quando $k = 2$. I Un CSP è GAC se e solo se tutti i suoi vincoli sono GAC.

Esempi

- | $D(X_1) = \{1,2,3\}$, $D(X_2) = \{2,3,4\}$, $C: X_1 = X_2$ – AC(C)?
 - | 1 є $D(X_1)$ e 4 є $D(X_2)$ non hanno supporto. | $X_1 = 1$ e $X_2 = 4$ sono assegnazioni parziali incoerenti.
- | $D(X_1) = \{1,2,3\}$, $D(X_2) = \{1,2\}$, $D(X_3) = \{1,2\}$,
 $C: \text{tuttidiversi}([X_1, X_2, X_3])$ – GAC(C)?
 - | 1 є $D(X_1)$ e 2 є $D(X_1)$ non hanno supporto. | $X_1 = 1$ e $X_1 = 2$ sono assegnazioni parziali incoerenti.

Coerenza dei limiti (BC)

I Definito per domini totalmente ordinati (ad esempio interi). I
rilassa

Rilassa il dominio di X_i da $D(X_i)$ a $[\min(X_i)..\max(X_i)]$.

- Ad esempio, $D(X_i) = \{1,3,5\}$ à $[1..5]$

I Un **supporto legato** è una tupla $(d_1, \dots, d_k) \in C$ dove di \in
 $[\min(X_i)..\max(X_i)]$. I

C(X_1, \dots, X_k) è BC se e solo se:

appartengono

- Per ogni X_i in $\{X_1, \dots, X_k\}$, $\min(X_i)$ e $\max(X_i)$ appartengono a un limite
supporto.

AVANTI CRISTO

I Svantaggio : BC

rilevare

potrebbe non rilevare tutte le incoerenze GAC in generale.

I Dobbiamo cercare di più.

I Vantaggi

cercare

che in un

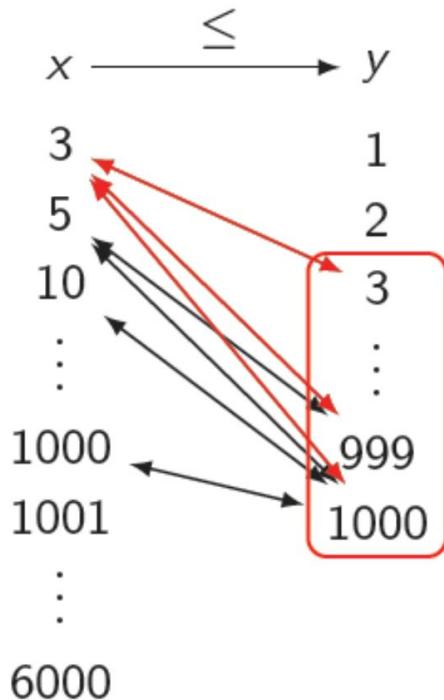
- Potrebbe essere più semplice cercare un supporto in un intervallo che in un dominio.
raggiungere + economico
- Raggiungere BC è spesso più economico che raggiungere GAC.

I Di interesse per i vincoli aritmetici definiti su variabili intere con domini di grandi dimensioni.

monotoni

- Il raggiungimento di BC è sufficiente per ottenere GAC per vincoli monotoni.

GAC = BC



- I Tutti i valori di $D(X) \setminus \max(Y)$ sono GAC.
- I Tutti i valori di $D(Y) \setminus \min(X)$ sono GAC.
- I Sufficiente per regolare $\max(X)$ e $\min(Y)$.
 - $\max(X) \setminus \max(Y)$
 - $\min(X) \setminus \min(Y)$

GAC > a.C

- | $D(X1) = D(X2) = \{1,2\}$, $D(X3) = D(X4) = \{2,3,5,6\}$, $X5 = 5$, $D(X6) = \{3,4,5,6,7\}$, tutti diversi($[X1, X2, X3, X4, X5, X6]$)
- | Solo 2 ѕ D(X3) e 2 ѕ D(X4) non hanno supporto BC.

	X1	X2	X3	X4	X5	X6
1	■	■				
2	■	■	■			
3						■
4						
5			■	■	■	■
6						
7						

Originale

	X1	X2	X3	X4	X5	X6
1	■	■				
2	■	■				
3						
4						
5			■	■	■	■
6						
7						

AVANTI CRISTO

GAC > a.C

| $D(X1) = D(X2) = \{1,2\}$, $D(X3) = D(X4) = \{2,3,5,6\}$, $X5 = 5$, $D(X6) = \{3,4,5,6,7\}$,
 tutti diversi([X1, X2 , X3 , X4 , X5 , X6])

| $\{2,5\} \in D(X3)$, $\{2,5\} \in D(X4)$, $\{3,5,6\} \in D(X6)$ non hanno supporto GAC.

	X1	X2	X3	X4	X5	X6
1	■	■				
2	■	■				
3			■	■		■
4						
5			■	■		■
6						
7			■	■		■

Originale

	X1	X2	X3	X4	X5	X6
1	■	■				
2	■	■				
3			■	■		■
4						
5			■	■		■
6						
7			■	■		■

AVANTI CRISTO

	X1	X2	X3	X4	X5	X6
1	■	■				
2	■	■				
3			■	■		■
4						
5			■	■		■
6						
7			■	■		■

GAC

Contorno

I Coerenza locale

- Consistenza dell'arco generalizzata (GAC)
- Coerenza dei limiti (BC)

I Propagazione dei vincoli

- Algoritmi di propagazione

I Propagazione specializzata

- Vincoli globali

I Vincoli globali per scopi generici

Propagazione dei vincoli

I Può apparire sotto nomi diversi:

- rilassamento dei vincoli
- filtraggio
- applicazione della coerenza locale, ...

I Una nozione di coerenza locale definisce le proprietà che a il vincolo C deve essere soddisfatto dopo la propagazione del vincolo.

comportamento operativo

- Il comportamento operativo è completamente lasciato libero.
- Possiamo sviluppare diversi algoritmi con diverse complessità ottenere lo stesso effetto.
- L'unico requisito è ottenere la proprietà richiesta su C.

Algoritmi di propagazione

I Un algoritmo di propagazione raggiunge un certo
livello di coerenza su un vincolo C rimuovendo
i valori incoerenti dai domini delle variabili in C .

I Il livello di consistenza dipende da C .

- GAC se è possibile sviluppare un algoritmo di propagazione efficiente.
- Altrimenti BC o un livello di consistenza inferiore.

Algoritmi di propagazione

I Quando si risolve un CSP con più vincoli:

- gli algoritmi di propagazione interagiscono;
- un algoritmo di propagazione può risvegliare un vincolo già propagato per essere propagato nuovamente!
- alla fine, la propagazione raggiunge un punto fisso e tutti i vincoli raggiungono un livello di coerenza;
- l'intero processo viene definito **propagazione dei vincoli**.

Esempio

I $D(X_1) = D(X_2) = D(X_3) = \{1, 2, 3\}$

C1: tutti diversi($[X_1, X_2, X_3]$) C2: $X_2 < 3$ C3: $X_3 < 3$

I Assumiamo: –

che l'ordine di propagazione sia C1, C2, C3;
– gli algoritmi di propagazione mantengono (G)AC.

I Propagazione di C1: – non
succede nulla, C1 è GAC.

I Propagazione di C2:
– 3 viene rimosso da $D(X_2)$, C2 ora è AC.

I Propagazione di C3: – 3
viene rimosso da $D(X_3)$, C3 ora è AC.

I C1 non è più GAC, perché i supporti di $\{1, 2\} \subseteq D(X_1)$ in $D(X_2)$ e $D(X_3)$ vengono rimossi dalla propagazione di C2 e C3.

I Ripropagazione di C1:
– 1 e 2 vengono rimossi da $D(X_1)$, C1 ora è AC.

Proprietà degli algoritmi di propagazione

- I Potrebbe non essere sufficiente rimuovere i valori incoerenti da domini una volta.
- I Un algoritmo di propagazione deve riattivarsi quando necessario, altrimenti potrebbe non raggiungere la proprietà di consistenza locale desiderata.
- I Eventi che possono innescare la propagazione di un vincolo:
 - quando cambia il dominio di una variabile (per GAC);
 - quando cambiano i limiti del dominio di una variabile (per BC);
 - quando ad una variabile viene assegnato un valore;
 - ...

Complessità degli algoritmi di propagazione

I Assumiamo $|D(X_i)| = d$.

I Seguendo la definizione della proprietà di consistenza
locale: –

una propagazione AC su un $C(X_1, X_2)$ richiede
un tempo

$O(d^2)$. I Possiamo fare di meglio!

Esempi

I C: $X_1 = X_2$ –

$D(X_1) = D(X_2) = D(X_1) \cup D(X_2)$

– Complessità: il costo dell'operazione di

intersezione tra

insiemi I C: $X_1 \cup X_2$ – Quando $D(X_i) = \{v\}$, rimuovi v da $D(X_j)$.

– Complessità: O(1)

I C: $X_1 \cap X_2$ –

$\max(X_1) \cap \max(X_2), \min(X_1) \cap \min(X_2)$

– Complessità: O(1)

Contorno

I Coerenza locale

- Consistenza dell'arco generalizzata (GAC)
- Coerenza dei limiti (BC)

I Propagazione dei vincoli

- Algoritmi di propagazione

I Propagazione specializzata

- Vincoli globali |
 - Scomposizioni |
 - Algoritmi ad hoc

I Vincoli globali per scopi generici

Propagazione specializzata

I Propagazione **specifica** per un dato **vincolo**.

I Vantaggi

- Sfrutta la semantica dei vincoli.
- Potenzialmente molto più efficiente di un approccio di propagazione generale.

Propagazione BC specializzata

IC : $X_1 = X_2 + X_3$

I Osservazione

- $\min(X_1)$ non può essere minore di $\min(X_2) + \min(X_3)$.
- $\max(X_1)$ non può essere maggiore di $\max(X_2) + \max(X_3)$.
- $\min(X_2)$ non può essere inferiore a $\min(X_1) - \max(X_3)$. –
 $\max(X_2)$ non può essere maggiore di $\max(X_1) - \min(X_3)$.
- X_3 analogo a X_2 .

Regole di propagazione BC

- $\max(X_1) \leq \max(X_2) + \max(X_3)$, $\min(X_1) \geq \min(X_2) + \min(X_3) - \max(X_2) \leq \max(X_1) - \min(X_3)$, $\min(X_2) \geq \min(\min(X_1) - \max(X_3))$
- Allo stesso modo per X_3

Esempio

| $D(X_1) = [4,9]$, $D(X_2) = [3,5]$, $D(X_3) = [2,3]$

C: $X_1 = X_2 + X_3$

Esempio

| $D(X_1) = [5,8]$, $D(X_2) = [3,5]$, $D(X_3) = [2,3]$

$$C: X_1 = X_2 + X_3$$

| Propagazione –

$$\max(X_1) \ddot{y} \max(X_2) + \max(X_3), \min(X_1) \ddot{y} \min(X_2) + \min(X_3)$$

Esempio

| $D(X_1) = [5,8]$, $D(X_2) = [3,5]$, $D(X_3) = [2,3]$

C: $X_1 = X_2 + X_3$

| Propagazione –

max(X_1) $\ddot{=}$ max(X_2) + max(X_3), min(X_1) $\ddot{=}$ min(X_2) + min(X_3)

– max(X_2) $\ddot{=}$ max(X_1) - min(X_3), min(X_2) $\ddot{=}$ min(X_1) - max(X_3)

– Allo stesso modo per X_3

Esempio

| $X_1 = 5$, $D(X_2) = [3,5]$, $D(X_3) = [2,3]$

C: $X_1 = X_2 + X_3$

| Propagazione –

max(X_1) ѕ max(X_2) + max(X_3), min(X_1) ѕ min(X_2) + min(X_3)

– max(X_2) ѕ max(X_1) - min(X_3), min(X_2) ѕ min(X_1) - max(X_3)

– Allo stesso modo per X_3

Esempio

$|X_1 = 5, D(X_2) = [3], D(X_3) = [2]$

C: $X_1 = X_2 + X_3$

I Propagazione –

$\max(X_1) \ddot{=} \max(X_2) + \max(X_3), \min(X_1) \ddot{=} \min(X_2) + \min(X_3)$

– $\max(X_2) \ddot{=} \max(X_1) - \min(X_3), \min(X_2) \ddot{=} \min(X_1) - \max(X_3)$

– Allo stesso modo per X3

Propagazione specializzata

I Propagazione **specifica** per un dato **vincolo**.

I Vantaggi –

Sfrutta la semantica dei vincoli.

- Potenzialmente molto più efficiente di un approccio di

propagazione generale. I Svantaggi

- Utilizzo limitato.
- Non è sempre facile svilupparne uno.

I Vale la pena svilupparlo per i vincoli ricorrenti.

Vincoli globali

- I Cattura complessi, non binari e ricorrenti sottostrutture combinatorie che si presentano in una varietà di applicazioni.
- I Incorpora la propagazione specializzata che sfrutta la sottostruttura.

Vantaggi dei vincoli globali

I Benefici della modellazione

- Ridurre il divario tra la formulazione del problema e il modello.
- Può consentire l'espressione di vincoli che altrimenti non sarebbe possibile enunciare utilizzando vincoli primitivi (**semantici**).

I Risoluzioni dei vantaggi

- **Forte** inferenza nella propagazione (**operazionale**).
- Propagazione **efficiente** (**algoritmica**).

Vincoli globali

I Cattura complessi, non binari e ricorrenti
sottostrutture combinatorie che si presentano
in una varietà di applicazioni.

I Incorpora la propagazione specializzata che
sfrutta la sottostruttura.

Vantaggi dei vincoli globali

I Benefici della modellazione

- Ridurre il divario tra la formulazione del problema e il modello.
- Può consentire l'espressione di vincoli che altrimenti non sarebbe possibile enunciare utilizzando vincoli primitivi (**semantici**).

I Risoluzione dei vantaggi

- Forte inferenza nella propagazione (**operazionale**).
- Propagazione efficiente (**algoritmica**).

Alcuni gruppi di vincoli globali

I Conteggio

I Sequenza I

Programmazione

I Ordinazione

I Bilanciamento

I Distanza

I Imballaggio

I Basato su grafico

I ...

Conteggio dei vincoli

I Limitare il numero di variabili che soddisfano una condizione o il numero di volte in cui vengono presi i valori.

Vincolo tutto diverso

I **tuttodiverso**([X₁, X₂, ..., X_k]) se e solo se

$X_i \neq X_j$ per $i < j \in \{1, \dots, k\}$ –
vincolo di permutazione con $|D(X_i)| = k$. –
alldifferent([3,5,2,1,4]) |

Utile in una varietà di contesti, come: –

- puzzle (ad esempio, sudoku e n-queens);
- calendarizzazione (es. assegnazione delle attività a diversi slot);
- programmazione (es. una squadra può giocare al massimo una volta ogni due giorni). settimana);
- configurazione (ad esempio, un particolare prodotto non può avere componenti ripetitivi).

Vincolo Nvalore

I Vincola il numero di valori distinti assegnati alle variabili.

I **Nvalore([X₁, X₂, ..., X_k], N)** sse $N = |\{X_i \mid 1 \leq i \leq k\}| - Nvalore([1, 2, 2, 1, 3], 3).$
– tutto diverso quando $N = k$.

I Utile ad esempio in:

- allocazione delle risorse (ad esempio limitare il numero di risorse tipi).

Vincolo di cardinalità globale

I Vincola il numero di volte in cui ciascun valore viene assunto dalle variabili.

I $\text{gcc}([X_1, X_2, \dots, X_k], [v_1, \dots, v_m], [O_1, \dots, O_m])$ sse
forall $j \in \{1, \dots, m\}$ $O_j = \{X_i \mid X_i = v_j, 1 \leq i \leq k\} - \text{gcc}([1, 1, 3, 2, 3], [1, 2, 3, 4], [2, 1, 2, 0])$
– tutto diverso quando $O_j \neq 1$.

I Utile ad esempio in:

- allocazione delle risorse (ad esempio, limitare l'utilizzo di ciascuna risorsa).

Tra vincoli

I Vincola il numero di variabili prese da un dato insieme di valori. I

`tra([X1, X2, ..., Xk], s, N) sse N = I`

$\{i \mid X_i \in s, 1 \leq i \leq k\} = I$

`tra([1, 5, 3, 2, 5, 4], {1,2,3,4}, 4) I`

`tra([X1, X2, ..., Xk], s, l, u) sse l \leq i`

$\{i \mid X_i \in s, 1 \leq i \leq k\} \leq u$

`between([1, 5, 3, 2, 5, 4], {1,2,3,4},`

`3, 4) I Utile nei problemi di sequenziamento, come vedremo`

Vincoli di sequenziamento

I Garantire che una sequenza di variabili rispetti determinati schemi.

Vincolo Sequenza/Tra le sequenze

I Vincola il numero di valori presi da un dato insieme in qualsiasi sottosequenza di variabili q.

I **sequenza(l, u, q, [X₁, X₂, ..., X_k], s)** sse
tra([X_i , X_{i+1}, ..., X_{i+q-1}], s, l, u) per 1 ≤ i ≤ k-q+1
– sequenza(1,2,3,[1,0,2,0,3],[0,1])

I Utile ad esempio

in: – turni (ad esempio ogni dipendente ha 2 giorni liberi in un periodo di 7 giorni);

– linea di produzione (ad esempio, al massimo 1 automobile su 3 lungo la linea di produzione può essere dotata di tetto apribile).

Vincoli di pianificazione

I Aiutare a pianificare le attività con i rispettivi tempi di rilascio, durata e scadenze, utilizzando risorse limitate in un intervallo di tempo.

Vincolo disgiuntivo delle risorse

I Richiede che le attività non si sovrappongano nel tempo.

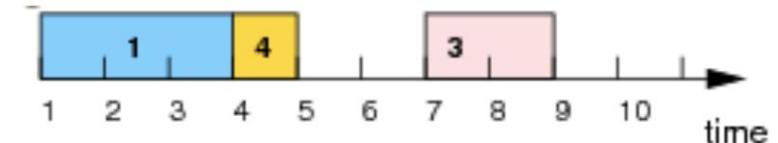
– Noto anche come vincolo **noOverlap**.

I Dati i compiti t_1, \dots, t_k , ciascuno associato a un'ora di inizio

Si e una durata D_i :

disgiuntivo([S₁, ..., S_k],[D₁, ..., D_k]) sse per tutti i
 $i < j \rightarrow (S_i + D_i \leq S_j) \wedge (S_j + D_j \leq S_i)$

I Utile quando una risorsa
può eseguire al massimo
un'attività alla volta.



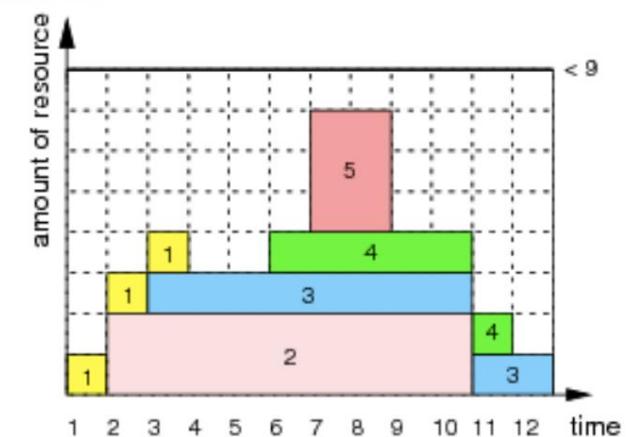
Vincolo di risorse cumulative

- Constrains the usage of a shared resource.
- Given tasks t_1, \dots, t_k , each associated with a start time S_i , duration D_i , resource requirement R_i , and a resource with a capacity C :

cumulative($[S_1, \dots, S_k], [D_1, \dots, D_k], [R_1, \dots, R_k], C$) iff

$$\sum_{i|S_i \leq u < S_i + D_i} R_i \leq C \text{ for all } u \text{ in } D$$

I Utile quando una risorsa con una capacità può eseguire più attività contemporaneamente.



Vincoli di ordinazione

I Applicare un ordinamento tra le variabili o i valori.

Vincolo di ordinamento lessicografico

I Richiede che una sequenza di variabili sia lessicograficamente inferiore o uguale a un'altra sequenza di variabili.

I $\text{lex} \ddot{y}([X_1, X_2, \dots, X_k], [Y_1, Y_2, \dots, Y_k])$ vale se e solo se:

$$X_1 \ddot{y} Y_1 \ddot{y}$$

$$(X_1 = Y_1 \wedge X_2 \ddot{y} Y_2) \ddot{y}$$

$$(X_1 = Y_1 \ddot{y} X_2 = Y_2 \wedge X_3 \ddot{y} Y_3) \dots$$

$$(X_1 = Y_1 \ddot{y} X_2 = Y_2 \dots X_{k-1} = Y_{k-1} \wedge X_k \ddot{y} Y_k)$$

– $\text{lex} \ddot{y}([1, 2, 4], [1, 3, 3])$

I Utile nella rottura della simmetria.

– Evitare permutazioni di (gruppi di) variabili.

Permutazione di variabili

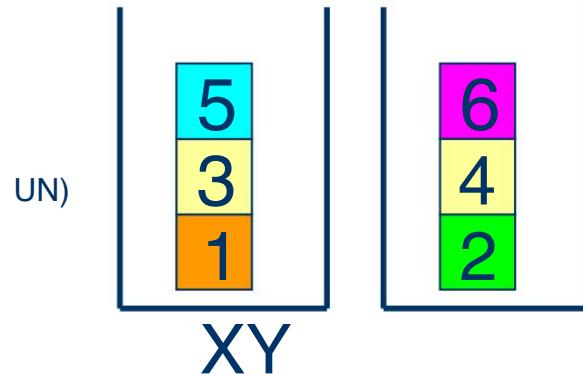
I $\text{lex}([X_1, X_2, \dots, X_k], [Y_1, Y_2, \dots, Y_k])$ per alcuni
 Y .

I Ad esempio, con n-regine:

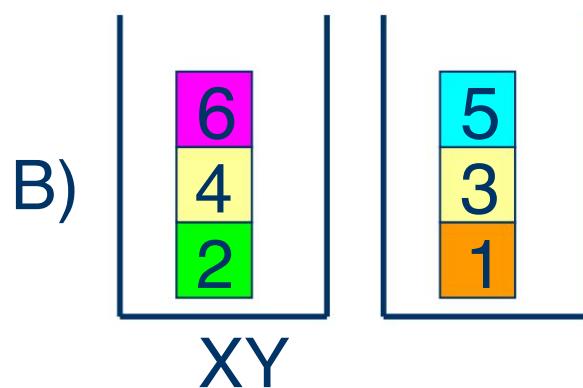
```
constraint
    lex_lesseq(array1d(qb), [ qb[j,i] | i,j in 1..n ])
    /\ lex_lesseq(array1d(qb), [ qb[i,j] | i in reverse(1..n), j in 1..n ])
    /\ lex_lesseq(array1d(qb), [ qb[j,i] | i in 1..n, j in reverse(1..n) ])
    /\ lex_lesseq(array1d(qb), [ qb[i,j] | i in 1..n, j in reverse(1..n) ])
    /\ lex_lesseq(array1d(qb), [ qb[j,i] | i in reverse(1..n), j in 1..n ])
    /\ lex_lesseq(array1d(qb), [ qb[i,j] | i,j in reverse(1..n) ])
    /\ lex_lesseq(array1d(qb), [ qb[j,i] | i,j in reverse(1..n) ])
;
```

Permutazione di due sequenze di Variabili

I Le assegnazioni di elementi a due contenitori identici possono essere rappresentate da una matrice di variabili booleane:



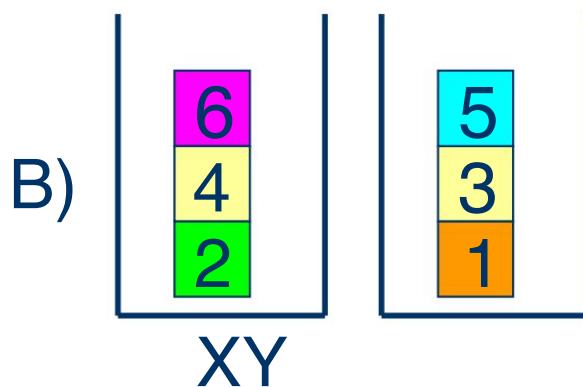
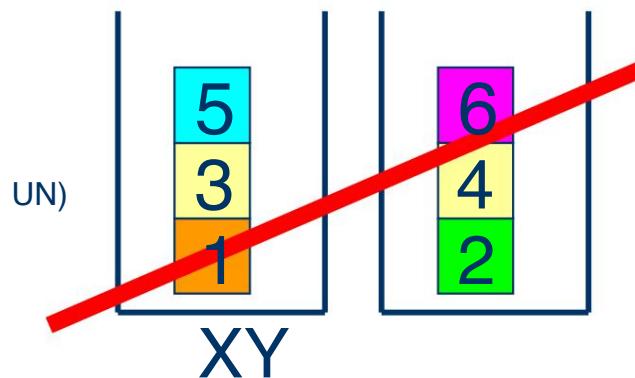
	i1	i2	i3	i4	i5	i6
X	1	0	1	0	1	0
Y	0	1	0	1	0	1



	i1	i2	i3	i4	i5	i6
X	0	1	0	1	0	1
Y	1	0	1	0	1	0

Permutazione di due sequenze di Variabili

I Necessità di evitare assegnazioni simmetriche.



i1 i2 i3 i4 i5 i6

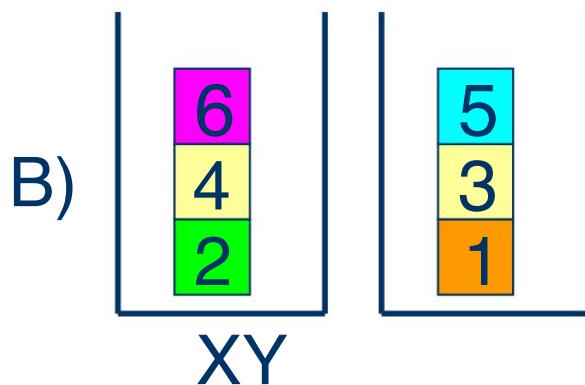
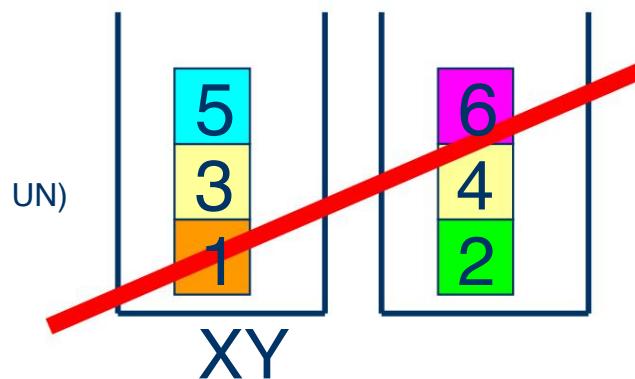
X	1	0	1	0	1	0
Y	0	1	0	1	0	1

i1 i2 i3 i4 i5 i6

X	0	1	0	1	0	1
Y	1	0	1	0	1	0

Permutazione di due sequenze di Variabili

I lex $\ddot{y}(X, Y)$.

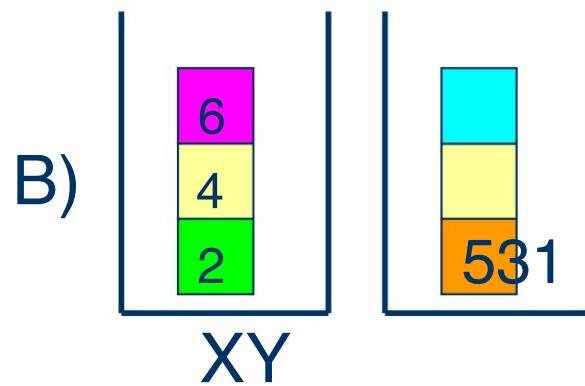


i1 i2 i3 i4 i5
X 1 0 1 0 1 0
Y 0 1 0 1 0 1

i1 i2 i3 i4 i5 i6
X 0 1 0 1 0 1
Y 1 0 1 0 1 0

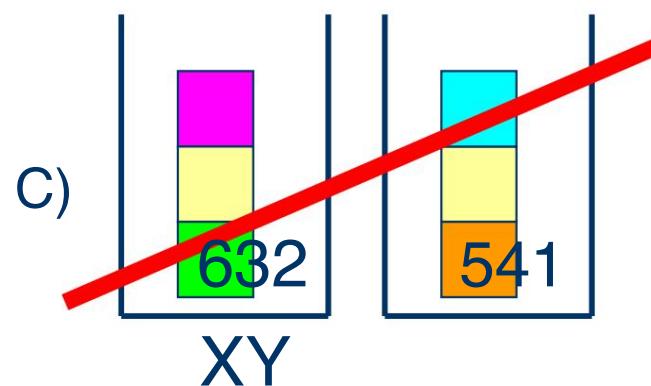
Permutazione di due sequenze di Variabili

I Necessità di evitare assegnazioni simmetriche.



i₁ i₂ i₃ i₄ i₅ i₆

X	0	1	0	1	0	1
Y	1	0	1	0	1	0

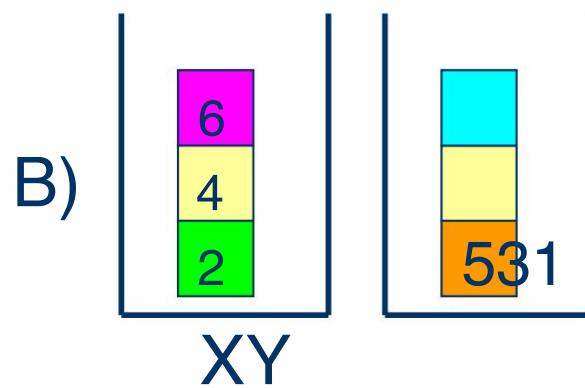


i₁ i₂ i₃ i₄ i₅ i₆

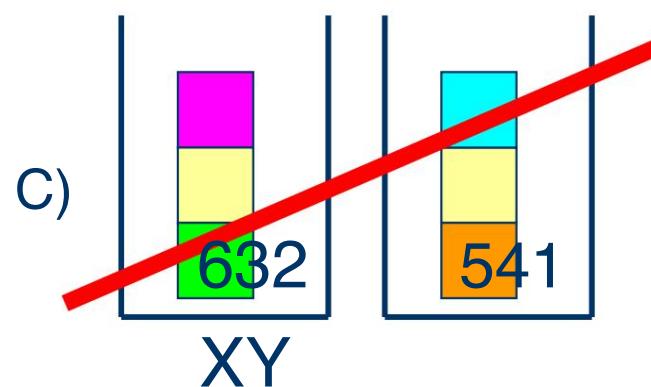
X	0	1	1	0	0	1
Y	1	0	0	1	1	0

Permutazione di due sequenze di Variabili

I lex $\ddot{y}(i_3, i_4)$.



i₁ i₂ i₃ i₄ i₅ i₆
X 0 1 0 1 0 1
Y 1 0 1 0 1 0



i₁ i₂ i₃ i₄ i₅ i₆
X 0 1 1 0 0 1
Y 1 0 0 1 1 0

Vincolo di precedenza dei valori

I Richiede che un valore preceda un altro valore in una sequenza di variabili.

I **value_precede(vj1, vj2, [X1, X2, ..., Xk])** vale se e solo se:

- $\min\{ i \mid X_i = vj1 \wedge i = k+1 \} < \min\{ i \mid X_i = vj2 \wedge i = k + 2 \}$.
- **valore_precede(5, 4, [2, 5, 3, 5, 4])**

I Utile nella rottura della simmetria.

- Evitare permutazioni di valori.

Propagazione specializzata per il globale Vincoli

I Come sviluppiamo la propagazione
specializzata per i vincoli globali?

I Due approcci principali:

- decomposizione dei vincoli;
- algoritmo dedicato ad hoc.

Decomposizione dei vincoli

- I Un vincolo globale viene scomposto in vincoli più piccoli e più semplici, ciascuno dei quali ha un algoritmo di propagazione noto.
- I La propagazione di ciascuno dei vincoli fornisce un algoritmo di propagazione per il vincolo globale originale.
 - Un metodo molto efficace ed efficiente per alcuni vincoli globali.

Una decomposizione di tra

I tra([X₁, X₂, ..., X_k], s, N)

I Scomposizione come congiunzione di vincoli logici e vincolo di somma.

- Bi con D(B_i) = {0, 1} per 1 ≤ i ≤ k
- C_i : B_i = 1 ≤ X_i ≤ s per 1 ≤ i ≤ k
- C_{k+1}: B_i = N ≤ i

I AC(C_i) per tutti i e BC(B_i = N ≤ i) garantisce GAC tra.

€

€

Una decomposizione di Lex

I $\text{lex } \ddot{y}([X_1, X_2, \dots, X_k], [Y_1, Y_2, \dots, Y_k])$

I Decomposizione come congiunzione di disgiunzioni.

- B_i con $D(B_i) = \{0, 1\}$ per $1 \leq i \leq k+1$ per indicare che i vettori sono stati ordinati per posizione $i-1$.

$$-B_1 = 0$$

- $C_i : (B_i = B_{i+1} = 0 \text{ AND } X_i = Y_i) \text{ OR } (B_i = 0 \text{ AND } B_{i+1} = 1 \text{ AND } X_i < Y_i) \text{ OR } (B_i = B_{i+1} = 1)$ per $1 \leq i \leq k$

I GAC (C_i) per ogni i garantisce GAC su $\text{lex } \ddot{y}$.

Decomposizioni dei vincoli

I Potrebbe non fornire sempre una propagazione efficace.

I Spesso il GAC sul vincolo originale è più forte del (G)AC sui vincoli nella scomposizione.

Una decomposizione di tutto diverso

I **tuttodiverso**([X₁, X₂, ..., X_k])

I Scomposizione come congiunzione di differenze vincoli.

– C_{ij}: X_i ≠ X_j per i < j ∈ {1,...,k}

I AC(C_{ij}) per tutti i < j è più debole di GAC su **tutti diversi**.

– Es. **tuttidiversi**([X₁, X₂, X₃]) con D(X₁) = D(X₂) = D(X₃) =

{1,2}. – **alldifferent** non è GAC ma la scomposizione non elimina nulla.

Una decomposizione della sequenza

I sequenza($l, u, q, [X_1, X_2, \dots, X_k], s$) |

Decomposizione come congiunzione di tra
vincoli.

– $C_i : \text{tra}([X_i, X_{i+1}, \dots, X_{i+q-1}], s, l, u)$ per $1 \leq i \leq k-q+1$

I GAC(C_i) per tutti i è più debole di GAC nella
sequenza.

– Es., **successione(2, 3, 5, [X1, X2, ..., X7], {1})**
con $X_1 = X_2 = 1, X_6 = 0, D(X_i) = \{0,1\}$ per $i \in \{3,4,5,7\}$.

– **la sequenza** non è GAC ma la
scomposizione non elimina nulla.

Una decomposizione della sequenza

| 1 1 {0,1} {0,1} {0,1} 0 {0,1} q=5, l =2, u =3,v={1}

| 1 1 {0,1} {0,1} {0,1} 0 {0,1} GAC(tr)

| 1 1 {0,1} {0,1} {0,1} 0 {0,1} GAC(tr)

| 1 1 {0,1} {0,1} {0,1} 0 {0,1} GAC(tr)

Una decomposizione della sequenza

| 1 1 {0,1} {0,1} {0,1} 0 {0,1} q=5, l =2, u =3,v={1}

| 1 1 {0,1} {0,1} {0,1} 0 {0,1} GAC(tr)

| 1 1 {0,1} {0,1} {0,1} 0 {0,1} GAC(tr)

| 1 1 {0,1} {0,1} {0,1} 0 {0,1} GAC(tr)

Una decomposizione di Lex

I **lex** $\ddot{y}([X_1, X_2, \dots, X_k], [Y_1, Y_2, \dots, Y_k])$

I Scomposizione come congiunzione di implicazioni

- $X_1 \ddot{y} Y_1$ AND $(X_1 = Y_1 \ddot{y} X_2 \ddot{y} Y_2)$ AND ...

$(X_1 = Y_1 \text{ AND } X_2 = Y_2 \text{ AND } \dots \text{ } X_{k-1} = Y_{k-1} \ddot{y} X_k \ddot{y} Y_k)$

I AC sulla decomposizione è più debole di GAC su **lex**.

- Ad esempio, **lex** $\ddot{y}([X_1, X_2], [Y_1, Y_2])$ con $D(X_1) = \{0,1\}$, $X_2 = 1$, $D(Y_1) = \{0,1\}$, $Y_2 = 0$

- **lex** \ddot{y} non è GAC ma la scomposizione non elimina nulla.

Decomposizione vs algoritmo ad hoc

- I Anche se una scomposizione è efficace, non sempre può fornire una propagazione efficiente.
- I Spesso propagare un vincolo tramite un algoritmo ad hoc è più veloce che propagare i (molti) vincoli nella scomposizione.
 - Grazie al calcolo incrementale!

Calcolo incrementale

I Un algoritmo di propagazione viene spesso richiamato più volte.
– Non vogliamo ricalcolare tutto ogni volta. I

Il **calcolo incrementale** può migliorare l'efficienza.

- Alla prima chiamata, alcuni risultati parziali vengono memorizzati nella cache.
- Alla successiva invocazione, sfruttiamo i **dati**

memorizzati nella cache. I Ciò richiede l'accesso a maggiori
dettagli sulla propagazione:

- quale variabile è stata eliminata?
- quali valori sono stati potati?

Algoritmo BC dedicato per la somma

IC : $X_i = N \cdot i$ – dove X_i e N sono variabili intere.

- $\min(N) \leq \min(X_i) \leq i$
- massimo(N) $\leq \max(X_i) \leq i$
- $\min(X_i) \leq \min(N) - \max(X_j) \leq j \leq i$ per $1 \leq i \leq n$
- $\max(X_i) \leq \max(N) - \min(X_j) \leq j \leq i$ per $1 \leq i \leq n$

BC Decomposizione per la somma

IC : $X_i = N \ddot{y}_i$ – dove X_i e N sono variabili intere.

- $X_1 + X_2 = Y_1$
- $Y_1 + X_3 = Y_2$
- ...
- $Y_{(n-1)} + X_n = N$

Filtraggio minimo(N)

IC : $X_i = N \ddot{y}_i$ – dove X_i e N sono variabili intere.

- $\min(X_1) + \min(X_2) \leq \min(Y_1)$
- $\min(Y_1) + \min(X_3) \leq \min(Y_2)$
- ...
- $\min(Y_{(n-1)}) + \min(X_n) \leq \min(N)$

che è equivalente a

$$\min(X_i) \leq i \leq \min(N)$$

Numero di operazioni

IC : $X_i = N \ddot{y} i$ _ dove X_i e N sono variabili intere.

- $\min(X_1) + \min(X_2) \ddot{y} \min(Y_1)$
- $\min(Y_1) + \min(X_3) \ddot{y} \min(Y_2)$
- ...
- $\min(Y_{(n-1)}) + \min(X_n) \ddot{y} \min(N)$

Accesso in lettura: $2(n-1)$

Accesso in scrittura: $n-1$

Somma: $n-1$

$\min(X_i) \ddot{y} i \ddot{y} \min(N)$

Accesso in lettura: n

Accesso in scrittura: 1

Somma: $n-1$

Numero di operazioni

IC : $X_i = N \ddot{y} i$ _ dove X_i e N sono variabili intere.

- $\max(X_1) + \max(X_2) \ddot{y} \max(Y_1)$
- $\max(Y_1) + \max(X_3) \ddot{y} \max(Y_2)$
- ...
- $\max(Y_{(n-1)}) + \max(X_n) \ddot{y} \max(N)$

Accesso in lettura: $2(n-1)$

Accesso in scrittura: $n-1$

Somma: $n-1$

$\max(X_i) \ddot{y} i$ _ \ddot{y} massimo(N)

Accesso in lettura: n

Accesso in scrittura: 1

Somma: $n-1$

Calcolo incrementale

IC : $X_i = N \ddot{y} i$ dove X_i e N sono variabili intere.

$\max(N) \ddot{y} l \quad \max(X_i) \ddot{y} i -$

Memorizza nella cache $\max(N)$ as

\max(N)$ l$ Ogni volta che i limiti di una variabile X_i vengono eliminati:

– $\max(N) \ddot{y} \max$(N)$ - (\text{vecchio}(\max(X_i)) - \max(X_i)) \quad O(1)$

Calcolo incrementale

IC : $X_i = N \ddot{y}_i$ _ dove X_i e N sono variabili intere.

– La complessità si riduce a $O(1)$ da $O(n)$

Somma classica

Accesso in lettura: n

Accesso in scrittura: 1

Somma: n-1

Somma incrementale

Accesso in lettura: 3

Accesso in scrittura: 1

Somma: 2

Algoritmi di propagazione dedicati

I Algoritmi dedicati ad hoc forniscono una propagazione **efficace ed efficiente** .

I Spesso:

- GAC è mantenuto in tempo polinomiale;
- vengono rilevati molti più valori incoerenti rispetto alle scomposizioni;
- il calcolo viene effettuato in modo incrementale.

Algoritmi di propagazione dedicati

I Algoritmi dedicati ad hoc forniscono una propagazione **efficace ed efficiente** .

I Spesso:

- GAC è mantenuto in tempo polinomiale;
- vengono rilevati molti più valori incoerenti rispetto alle scomposizioni;
- il calcolo viene effettuato in modo incrementale.

Un algoritmo di propagazione GAC

I Mantiene GAC su **alldifferent([X₁, X₂, ..., X_k])** e viene eseguito in tempo polinomiale.

- Jean-Charles Régin, “Un algoritmo di filtraggio per i vincoli di Differenza dei CSP”, nella Proc. dell'AAAI'1994

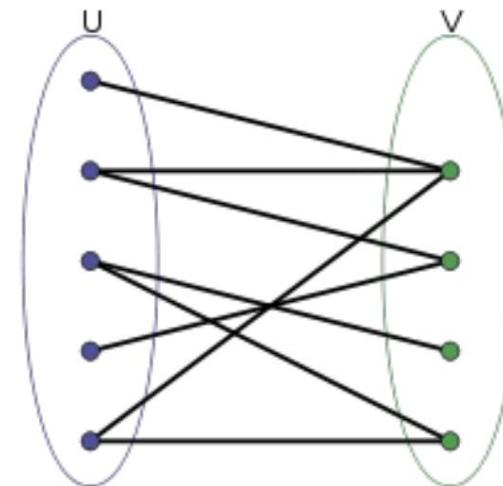
I Stabilisce una relazione tra le soluzioni del vincolo e le proprietà di un grafico.

- Matching massimo in un grafo bipartito.

I Un algoritmo simile può essere ottenuto con l'uso del flusso teoria.

Un algoritmo GAC per tutti diversi

I **Un grafo bipartito** è un grafo i cui vertici sono divisi in due insiemi disgiunti U e V tali che ogni spigolo collega un vertice in U con uno in V .

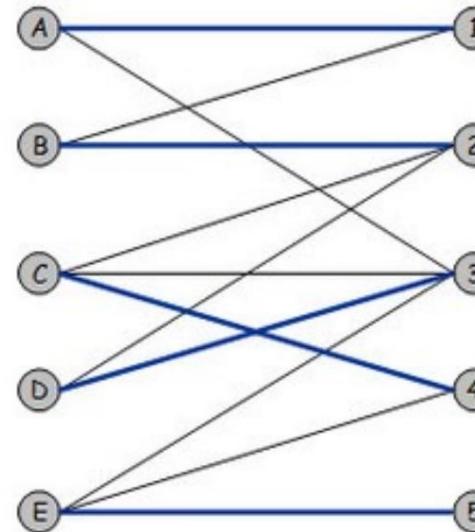


Un algoritmo GAC per tutti diversi

I Una **corrispondenza** in un grafico è un sottoinsieme dei suoi archi tale che non esistono due bordi che abbiano un nodo in comune.

– L'**abbinamento massimo** è l'abbinamento più grande possibile.

I In un grafo bipartito, la corrispondenza massima copre un insieme di nodi.



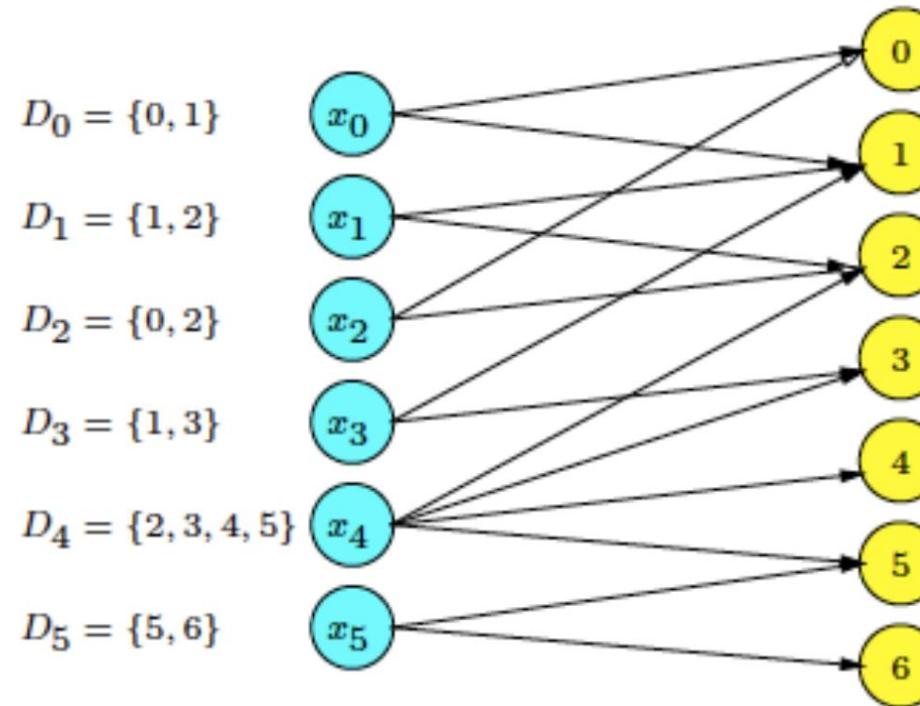
Un algoritmo GAC per tutti diversi

I Osservazione

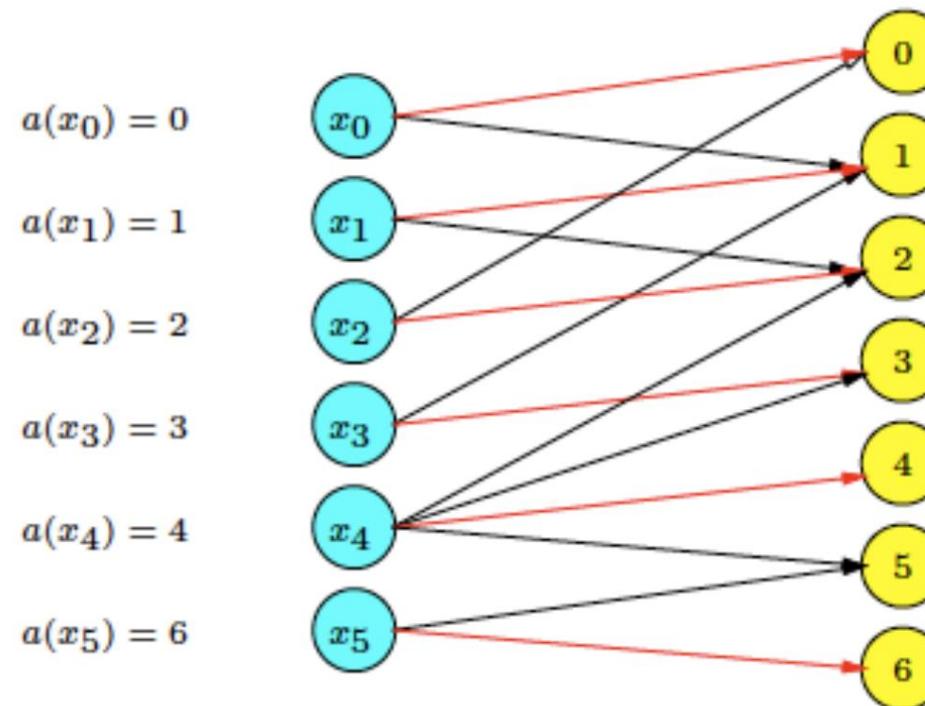
- Dato un **grafo bipartito G** costruito tra le variabili $[X_1, X_2, \dots, X_k]$ e i loro possibili valori (**grafico a valori variabili**),
- un'assegnazione di **valori** alle variabili è una soluzione se e solo se corrisponde a un **match massimale** in G.
 - I Un abbinamento massimale copre tutte le variabili.
- Calcolando **tutti gli abbinamenti massimali**, possiamo trovare tutti i assegnazioni parziali coerenti.

Esempio

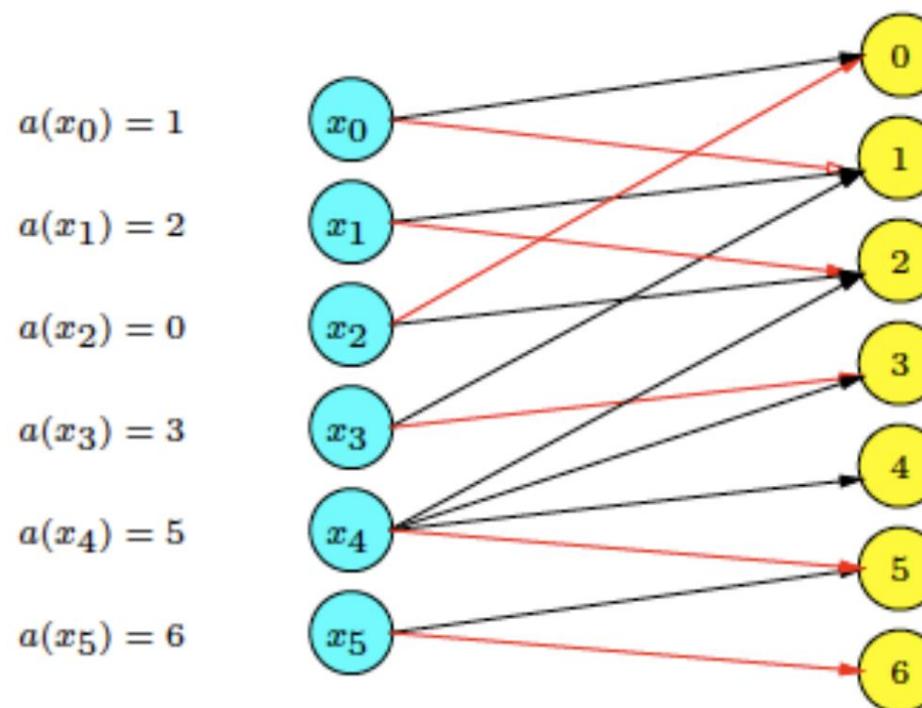
Grafico a valore variabile



Una corrispondenza massima



Un altro abbinamento massimale



Notazioni corrispondenti

I Edge –

abbinamento se prende parte ad un
abbinamento; – libero altrimenti.

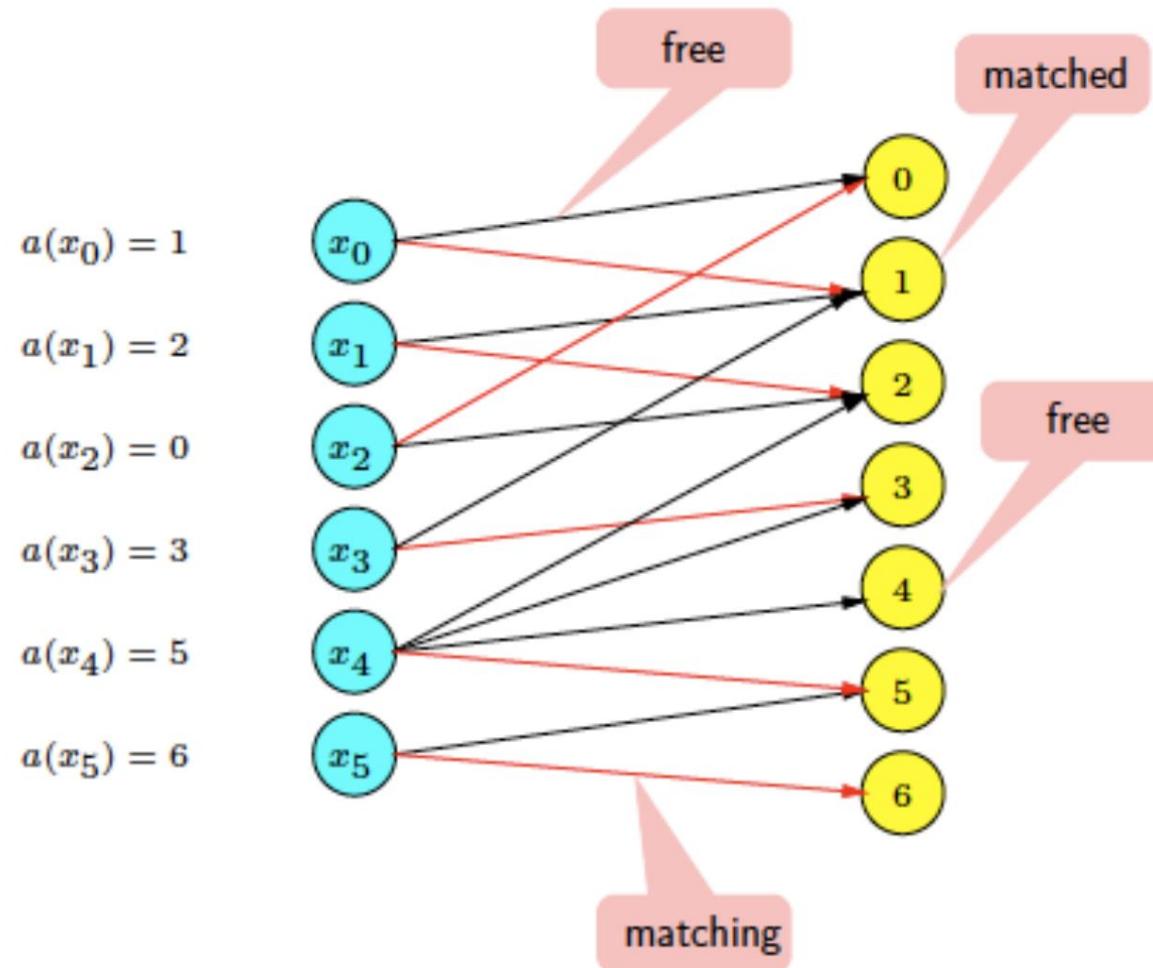
I Nodo

- abbinato se incidente su un bordo corrispondente;
- libero altrimenti.

I Bordo vitale –

appartiene ad ogni abbinamento massimale.

Gratuito, abbinato, abbinato



Algoritmo

I Calcolare tutti gli abbinamenti massimali. I

Non esiste un abbinamento massimale in caso di fallimento. I Un **bordo libero** in tutti gli abbinamenti massimali à

- **Rimuovere il bordo.**
- Equivale a **rimuovere il valore** corrispondente dal dominio del **variabile** corrispondente . I

Un vantaggio vitale à

- **Mantieni il vantaggio.**
- Equivale ad **assegnare il valore** corrispondente al corrispondente **variabile**.

I Corrispondenza dei bordi in alcuni ma non in tutti gli abbinamenti massimi à – **Mantieni il bordo.**

Tutti gli abbinamenti massimi

I Inefficiente calcolarli ingenuamente. I Utilizzare la teoria delle corrispondenze per calcolarli in modo efficiente.

- Un abbinamento massimale può descrivere tutti gli abbinamenti massimali!

Percorso e Ciclo Alternati

I **Percorso alternato**

– Percorso semplice con bordi alternati liberi e combacianti.

I **Ciclo alternato** – Ciclo

con bordi alternati liberi e accostati.

I Lunghezza del percorso/

ciclo – Numero di bordi nel percorso/

ciclo. I **Percorso/ciclo**

pari – Percorso/ciclo di lunghezza pari.

Teoria della corrispondenza

I Un risultato dovuto a Claude Berge nel 1970. I Un arco **e** appartiene ad un **abbinamento massimale** se e solo se per qualche **abbinamento massimale arbitrario M**:

- o **e** appartiene a M;
- oppure **e** appartiene ad un percorso pari alternato che parte da un nodo libero;
- oppure **e** appartiene ad un **ciclo alternato pari**.

Grafico orientato

I Per calcolare percorsi/cicli alternati, lo faremo
orientare i bordi di un abbinamento massimale arbitrario:
– abbinamento dei bordi à da **variabile a
valore**; – bordi liberi à da **valore a variabile**.

Un abbinamento massimale arbitrario

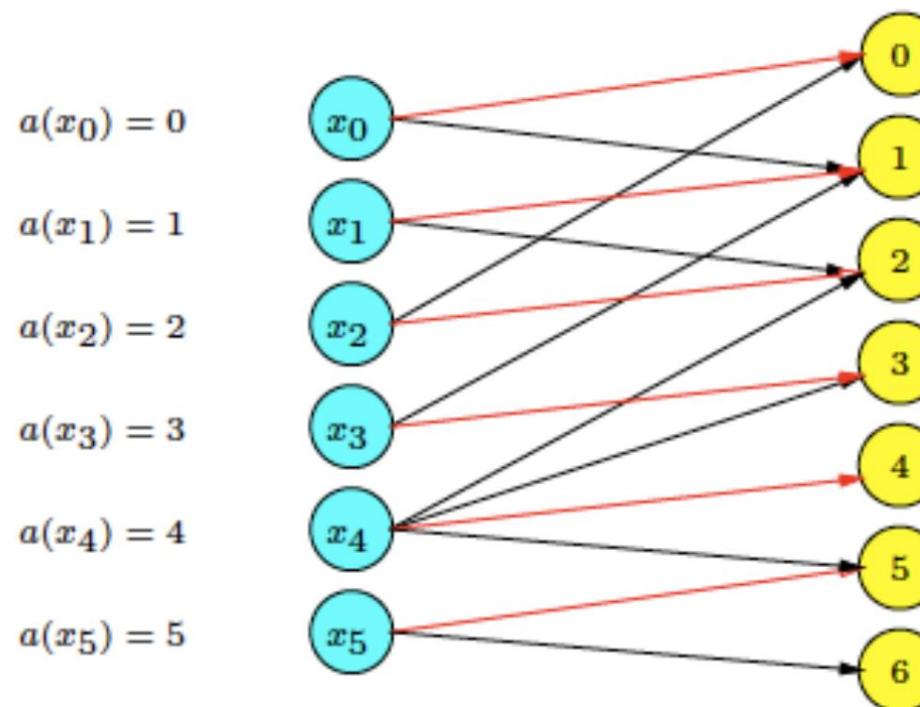
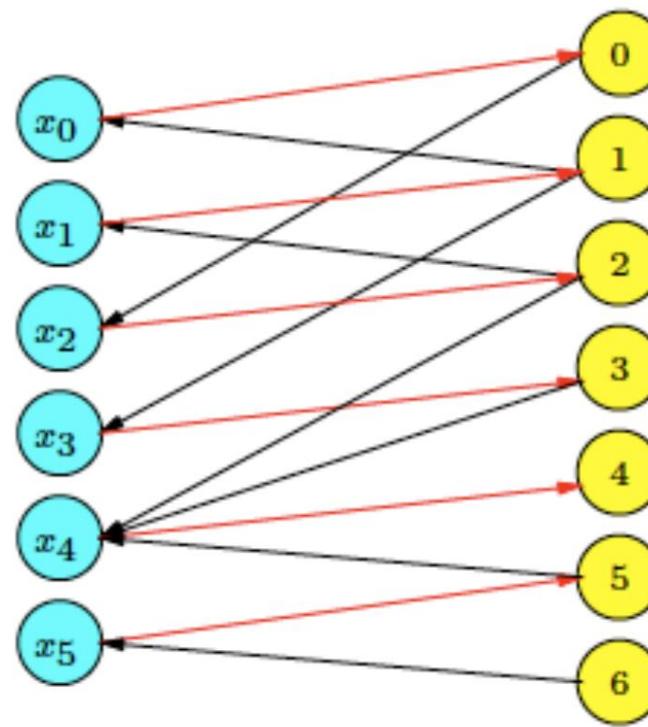


Grafico orientato



Anche percorsi alternati

I Iniziare da un nodo libero e cercare tutti i nodi sul percorso semplice diretto.

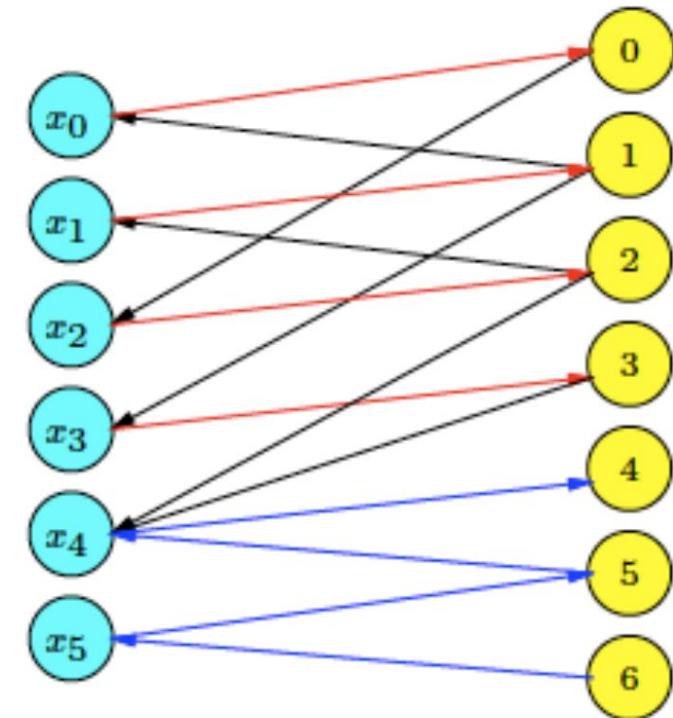
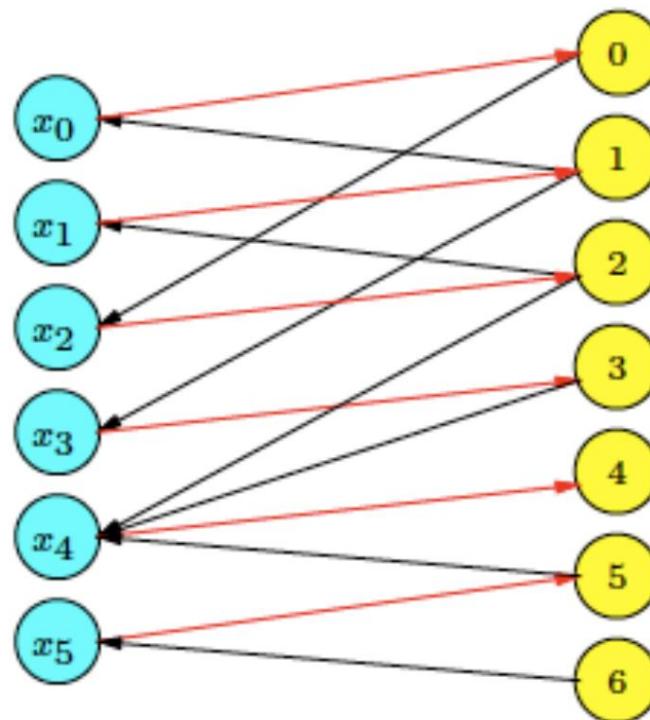
- Segna tutti i bordi sul percorso.
- Alternanza incorporata.

I Iniziare da un nodo valore.

- I nodi variabili sono tutti abbinati.

I Termina con un nodo di valore per una lunghezza pari.

Anche percorsi alternati



- Intuizione: i bordi possono essere permutati.

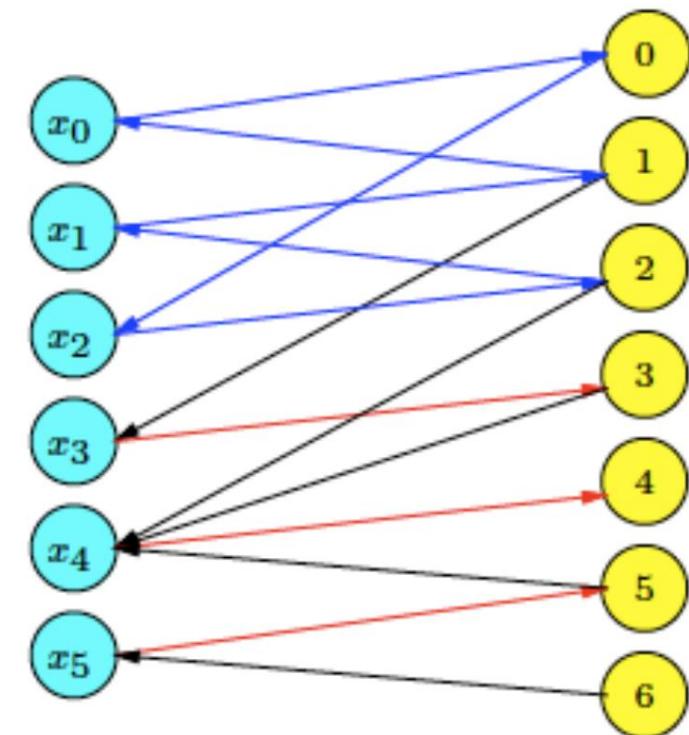
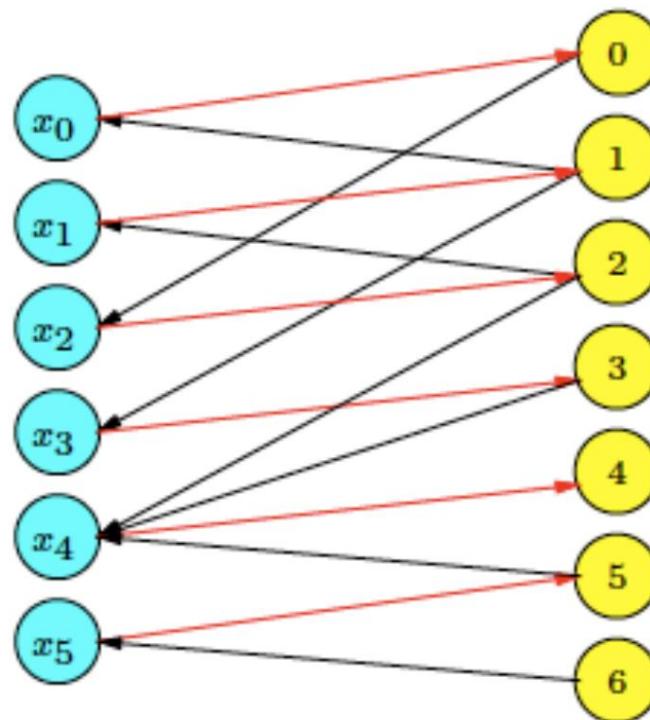
Anche Cicli Alternati

I Calcolare componenti fortemente connessi (SCC).

- Due nodi **aeb** sono fortemente connessi **se e solo se** esiste un **cammino da a a b e un percorso da b ad a.**
- **Componente fortemente connessa:** due nodi qualsiasi sono fortemente connessi collegato.
- Alternanza e lunghezza uniforme integrate.

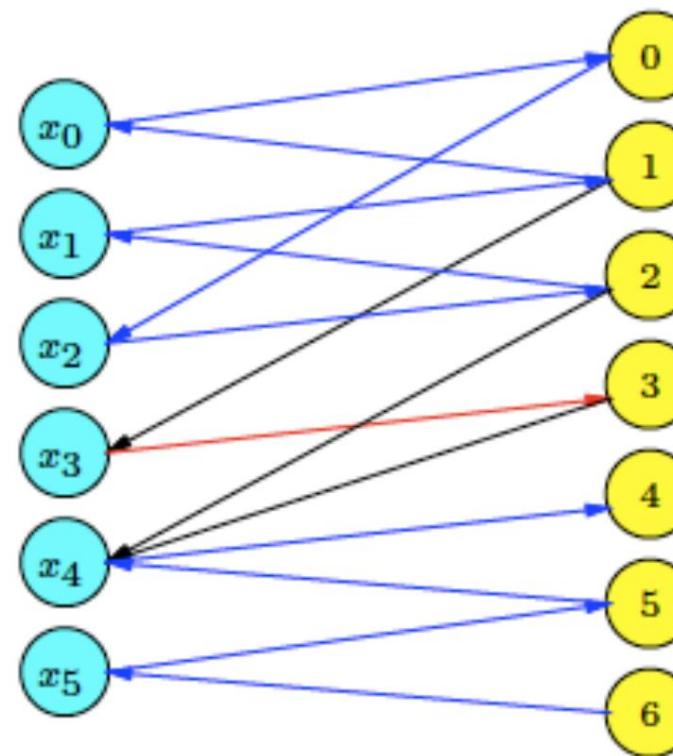
I Contrassegnare tutti i bordi in tutti i componenti fortemente collegati.

Anche Cicli Alternati



- Intuizione: le variabili consumano tutti i valori.

Tutti i bordi contrassegnati



Rimozione dei bordi

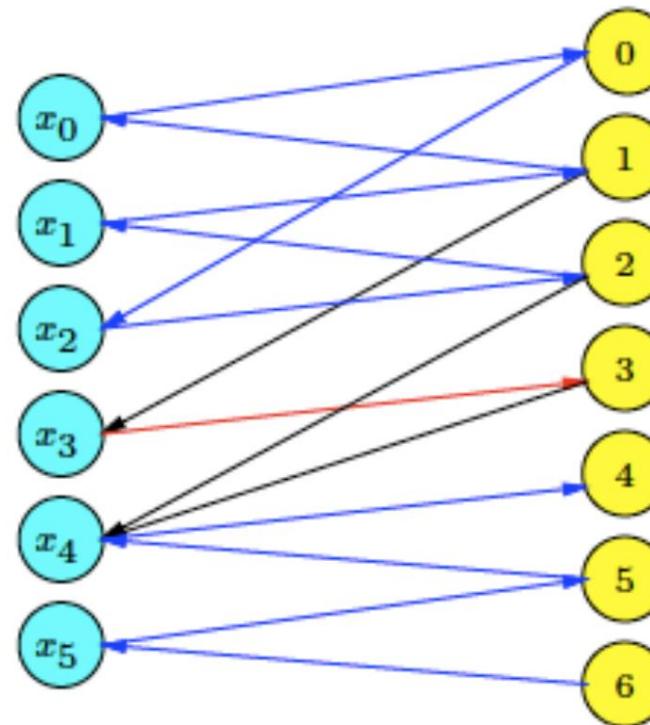
I Rimuovere i bordi che sono:

- **libero** (non si verifica nel nostro abbinamento massimale arbitrario) e
non marcato (non si verifica in nessun abbinamento massimale);
- contrassegnato come nero nel nostro esempio.

I Mantenere il bordo **abbinato** e **non segnato**.

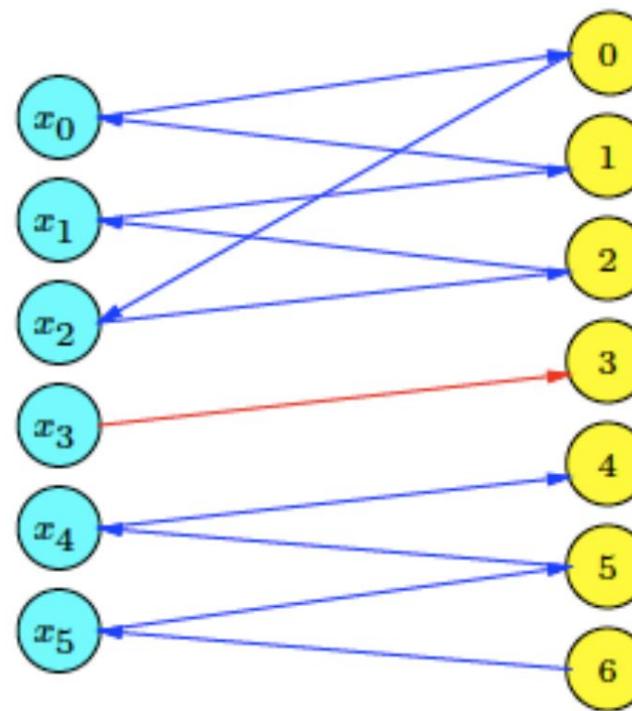
- Contrassegnato in rosso nel nostro esempio.
- Bordo vitale!

Rimozione dei bordi



$D(X0) = \{0, 1\}$, $D(X1) = \{1, 2\}$, $D(X2) = \{0, 2\}$, $D(X3) = \{1, 3\}$
 $D(X4) = \{2, 3, 4, 5\}$, $D(X5) = \{5, 6\}$

Bordi rimossi



$D(X0) = \{0, 1\}$, $D(X1) = \{1, 2\}$, $D(X2) = \{0, 2\}$, $D(X3) = \cancel{\{1, 3\}}$
 $D(X4) = \cancel{\{2, 3, 4, 5\}}$, $D(X5) = \{5, 6\}$

Riepilogo dell'algoritmo

- I Costruire il grafico dei valori variabili.
- I Trovare un abbinamento massimale M ; altrimenti fallisci.
- I Orientare il grafico (eseguito durante il calcolo di M). I
Contrassegnare i bordi partendo da nodi a valore libero utilizzando
la ricerca nel grafo.
- I Calcolare gli SCC e contrassegnare i bordi di unione. I
Rimuovere i bordi non segnati e liberi.

Proprietà incrementali

I Conservare il grafico della variabile e del valore tra le diverse invocazioni.

I Quando viene eseguito nuovamente:

- rimuovere i segni sui bordi;
- rimuovere i bordi non nei domini delle rispettive variabili;
- se viene rimosso un arco corrispondente, calcola un nuovo abbinamento massimale;
- altrimenti basta ripetere la marcatura e la rimozione.

Complessità di esecuzione

- **alldifferent([X₁, X₂, ..., X_k])** with m = $\sum_{i \in \{1,..k\}} |D(X_i)|$
- First call
 - Consistency check in $O(\sqrt{k}m)$ time.
 - Matching → $O(\sqrt{k} m)$
 - Alternating path → $O(m)$
 - SCCs → $O(k+m)$
 - Establishing GAC in $O(m)$ time.
- After q variable domains have been modified
 - Matching in $O(\min\{qm, \sqrt{k} m\})$ time.
 - Establishing GAC in $O(m)$ time.

Algoritmi dedicati ad hoc

I È sempre facile sviluppare un algoritmo dedicato per un dato vincolo? I Una bella semantica spesso ci dà un indizio!

- Teoria dei grafi
- Teoria dei flussi – Combinatoria
- Teoria degli automi
- Programmazione dinamica
- Teoria della complessità, ...

Algoritmi dedicati ad hoc

I GAC potrebbe anche essere NP-difficile!

- Ad esempio, **nvalue**, **sequenza+gcc**, **gcc** utilizzando variabili per gli eventi.
- Sono interessanti gli algoritmi che mantengono coerenze più deboli.

I aC

I Tra GAC e BC

I GAC su alcune variabili, BC su altre

I ...

Algoritmi dedicati ad hoc

I Cosa succede se è difficile:

- decomporre un vincolo;
- costruire un algoritmo dedicato efficiente ed efficace?

Contorno

I Coerenza locale

- Consistenza dell'arco generalizzata (GAC)
- Coerenza dei limiti (BC)

I Propagazione dei vincoli

- Algoritmi di propagazione

I Propagazione specializzata

- Vincoli globali I
 - Scomposizioni I
 - Algoritmi ad hoc

I Vincoli globali per scopi generici

Vincoli globali per scopi generici

I Contribuire a propagare un'ampia gamma di vincoli.

- Vincolo della tabella.
- Vincoli formali basati sul linguaggio.

Vincolo (estensionale) della tabella

| $C(X_1, X_2) = \{(0,0), (0,2), (1,3), (2,1)\}$

| Esistono diversi algoritmi per mantenere GAC.

- Più efficiente di $O(|ID(X_1)| * |ID(X_2)| * ... * |ID(X_k)|)$.
- Più efficace della decomposizione.

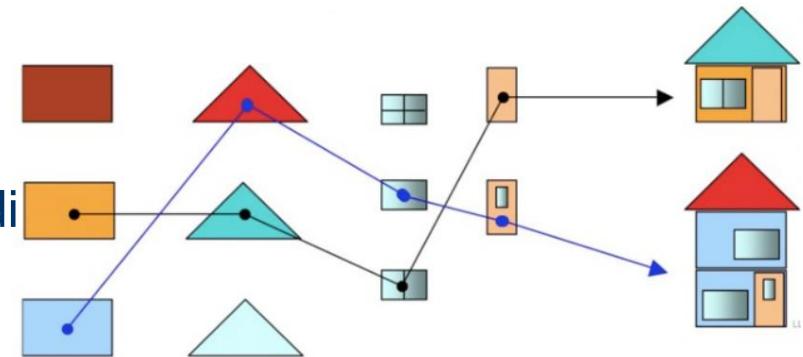
| Ad esempio, $(X_1=0 \text{ AND } X_2=2 \text{ AND } X_3=2) \text{ OR } (X_1=1 \text{ AND } X_2=1 \text{ AND } X_3=2) \text{ OR } (X_1=1 \text{ AND } X_2=2 \text{ AND } X_3=3)$

Problemi di configurazione del prodotto

I Vincoli di compatibilità sui componenti del prodotto.

- Spesso solo alcune combinazioni di componenti funzionano insieme.

I La compatibilità potrebbe non essere a semplice relazione di coppia.



Un problema di configurazione

I prodotti hw validi sono definiti in una tabella dei componenti compatibili (Prodotti):

Prodotti Scheda madre CPU Freq RAM Disco rigido

Prodotto1	Digitare un	Intel 2 GHz 5 GB 100 GB
Prodotto2	Tipo B	Intel 3GHz 8GB 200GB
Prodotto3	Tipo B	AMD 2GHz 5GB 200GB
...		

I Supponiamo di avere prodotti Pi da configurare ciascuno con 5 componenti per scheda madre, CPU, Freq, RAM e h. azionamento [Xi1, Xi2, Xi3, Xi4, Xi5].

I Per ogni prodotto Pi , pubblichiamo la tabella([Xi1,Xi2,Xi3, Xi4, Xi5], Prodotti).

Parole crociate

I Le parole valide sono definite in una tabella di

lettere compatibili (cioè dizionario). –

tabella([X1,X2,X3],

dizionario)

– tabella([X1,X13,X16], dizionario)

– tabella([X4,X5,X6,X7], dizionario)

– ...

I Non esiste un modo semplice per decidere
parole accettabili se non per
metterle in una tabella.

1	2	3	A	T	4	T	5	S	N	6	I	7		8	P	9	E	10	R	11	C	12	H
13	E	C	A		14	H	T	O	G			15	T	U	R	T	L	E					
16	S	H	I	17	B	A	I	N	U			18	O	R	R		19	O	R				
				20	L	A	I	C		21	A	22	B	E	R		23	F	W	D			
24	B	25	W	L		26	K	A	N	E		27		28	S	H	E	D	I				
30	S	W	A	L	31	C		32	R	A	S	33	P		34	O	W	E	N				
35	E	N	G		36	H	37	A	M	S	T	E	38	R	S		39	R	G				
				40	S	41	S	I	M			42	T	A	E	43	M						
44	S	45	F		46	P	A	R	47	A	48	K	49	E	E	T	50	U	51	S	52	A	
53	C	E	I	54	C		55	E	Y	E	S		56	57	S	K	I	N	S				
58	R	E	T	A	59	W		60	A	N	E	61	W		62	E	R	E	H				
63	A	D	S		64	H	65	A	H	N		66	67	O	K	R	A						
68	T	E		69	A	E	S		70	E	71	U	K	A	N	72	B	73	A				
74	C	R	A	75	T	E	S		76	L	A	E	R		77	Q	U	O					
78	H	S	A	E	L				79	S	E	N	T		80	A	T	L					

Vincoli basati sul linguaggio formale

I Il vincolo della tabella richiede il precalcolo di tutte le soluzioni di un vincolo.

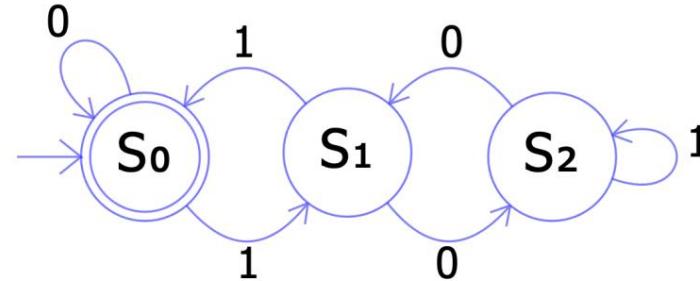
- Potrebbe non essere sempre possibile o pratico.

I Possiamo utilizzare un automa deterministico a stati finiti per definire le soluzioni.

- Utile soprattutto quando gli incarichi validi devono obbedire a determinati schemi.

Automa deterministico a stati finiti

- I Una dfa è una macchina a stati finiti che accetta o rifiuta a data stringa di simboli, eseguendo una sequenza di stati determinata univocamente dalla stringa.
 - Riconosce una lingua normale.
- I Ad esempio, un dfa che accetta numeri binari che sono multipli di 3.



- Alcune stringhe accettate: 0, 11, 110, 1100, 1001, 10111101, ...
- Stringhe non accettate: 10, 100, 101, 10100, ...

Vincolo regolare

I A ddfa A è definito da una tupla di 5 (q , σ , t , q_0 , f) dove:

- q : un insieme finito di stati
- σ : un insieme di simboli (cioè alfabeto)
- t : una funzione di transizione parziale $q \times \sigma \rightarrow q$
- q_0 : stato iniziale
- $f \subseteq q$: accetta stati (finali) I

regolare([X₁, X₂, ..., X_k], A) vale se e solo se $\langle X_1, X_2, \dots, X_k \rangle$ forma una stringa accettata da un ddfa A.

Problemi di roster

I turni sono soggetti a regolamentazione.

- Ad esempio, i turni notturni successivi devono essere limitati.

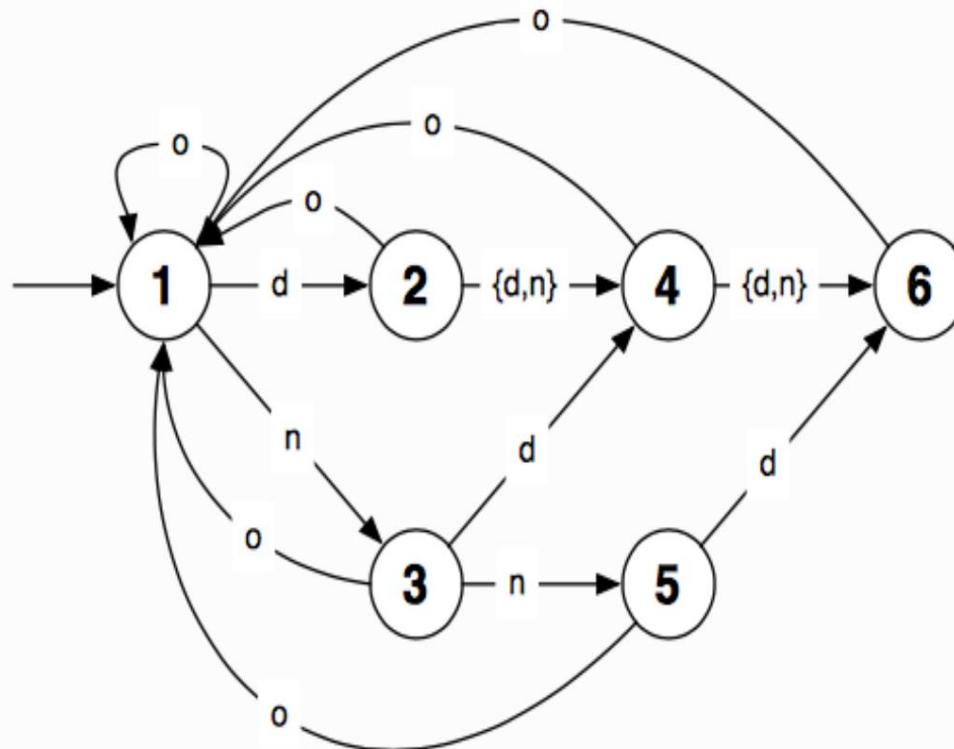
In un problema di turni degli infermieri, supponiamo:

- ogni infermiere è programmato per ogni giorno: (d) in turno diurno, (n) in turno notturno, o (o) libero; – in ogni quadriennio

l'infermiere deve avere almeno un giorno libero;

- nessun infermiere può essere programmato per 3 turni notturni in una riga.

Un problema di turni degli infermieri



$$|q = \{q_1, \dots, q_6\}| \\ \sigma = \{d, n, o\} | t:$$

	d	n	o
1	2	3	1
2	4	4	1
3	4	5	1
4	6	6	1
5	6	0	1
6	0	0	1

$$|q_0 : q_1| \\ f = q = \{q_1, \dots, q_6\} | Si$$

supponga che gli infermieri N_i siano programmati per 30 giorni [D_{i1}, \dots, D_{i30}]. Per ogni infermiera N_i , pubblichiamo $\text{normal}([D_{i1}, \dots, D_{i30}], A)$

Vincolo regolare

I Utile nei problemi di sequenziamento e rostering. I Molti vincoli sono istanze di **regolari**:

- tra, lex, precedenza, stretch, ...

I Propagazione efficiente del GAC con un algoritmo dedicato e una scomposizione in una sequenza di vincoli ternari.

- Un altro esempio del potere delle scomposizioni!