Appunti del Corso di Intelligenza Artificiale

Problemi di Soddisfacimento di Vincoli

Prof. Federico Bergenti

21 aprile 2024

1 Problemi di Soddisfacimento di Vincoli

Un **problema di soddisfacimento di vincoli** (**Constraint Satisfaction Problem**, **CSP**) è una tripla $\langle V, D, C \rangle$ in cui

- $V \neq \emptyset$ è un insieme non vuoto e finito di simboli detti variabili con $n = |V|$;
- $D \neq \emptyset$ è un insieme non vuoto di insiemi detti **dominî** delle variabili con $|D| \leq n$; e
- C è un insieme finito di **vincoli**.

Detta $dom : V \to D$ una funzione suriettiva totale che associa un dominio ad ogni variabile, si può parlare, per ogni variabile $x \in V$, del suo dominio $dom(x)$. In più, si suppone sempre che esista un ordinamento totale delle variabili e che questo ordinamento venga sempre usato in senso crescente quando vengono elencate le variabili. Normalmente, l'ordinamento utilizzato per le variabili è lasciato implicito ed evidente dal contesto. Utilizzando l'ordinamento delle variabili è possibile definire il dominio di un CSP $\mathcal{P}$ come

$$dom(\mathcal{P}) = \prod_{x \in V} dom(x) \tag{1}$$

dominio del problema di soddisfacimento dei vincoli

Prodotto cartesiano dei domini delle variabili

dove V è l'insieme delle variabili del CSP e le variabili vengono elencate nella costruzione del prodotto cartesiano in senso crescente secondo l'ordinamento scelto.

Un vincolo è un legame tra qualche variante

Per definire un vincolo:
- sottoinsieme delle variabili su cui è definito il vincolo

Dato un CSP con variabili in V, ogni vincolo $c \in C$ è una coppia $\langle V_c, \Delta_c \rangle$ dove $V_c \subseteq V$ è un insieme di variabili del CSP e Δ_c è

$$\Delta_c \subseteq \prod_{x \in V_c} dom(x) \tag{2}$$

dove le variabili vengono elencate nella costruzione del prodotto cartesiano in senso crescente secondo l'ordinamento scelto e Δ_c viene detto insieme degli **assegnamenti (parziali) consistenti (o ammissibili)** del vincolo c. Se $|V_c| = 1$, allora c viene detto **unario**, se $|V_c| = 2$, allora c viene detto **binario**, mentre c viene detto **globale** in tutti gli altri casi. Infine, si noti che $vars(c) = V_c$ è un modo per fare riferimento alle variabili di un vincolo e

→ insieme delle n-ple con un numero di elementi pari alla cardinalità di V_c, che identifica l'insieme degli aggiornamenti delle variabili che sono consistenti con il vincolo

$$dom(c) = \prod_{x \in V_c} dom(x) \tag{3}$$

→ prodotto cartesiano dei domini delle variabili coinvolte nel vincolo

viene detto dominio del vincolo c se si assume che le variabili vengano elencate in senso crescente secondo l'ordinamento scelto.

Fissato un CSP $\mathcal{P}$, è sempre possibile estendere Δ_c ad un sottoinsieme del dominio del CSP $\Delta = dom(\mathcal{P})$ aggiungendo alle ennuple di Δ_c le componenti mancanti. Siccome se una variabile non è coinvolta in un vincolo i suoi valori sono tutti e soli quelli del proprio dominio, le componenti aggiunte alle ennuple di Δ_c potranno assumere un valore qualsiasi nei dominî delle rispettive variabili. Detta imguna funzione suriettiva totale che associa ad ogni vincolo l'estensione a Δ dell'insieme degli assegnamenti (parziali) consistenti, si può parlare, per ogni vincolo $c \in C$, del suo insieme degli **assegnamenti totali consistenti (o ammissibili)** $img(c) \subseteq \Delta$.

Se tutti i dominî delle variabili di un vincolo sono finiti, allora è possibile descrivere il vincolo elencando tutti gli assegnamenti parziali consistenti in modo **estensionale** e il vincolo viene detto **tabellare**. La rappresentazione tabellare diventa impraticabile anche per dominî con pochi elementi e quindi spesso si preferisce la rappresentazione **intensionale**.

Esempio 1. Si consideri un CSP con tre variabili, x, y e z, tali che $dom(x) = dom(y) = dom(z) = [1..6]$. L'insieme delle variabili del CSP è $V = \{x, y, z\}$ e il dominio del CSP è $\Delta = [1..6]^3$ dove, come di consueto, le variabili vengono ordinate in senso alfabetico. Il CSP contiene un vincolo c definito dalla proprietà

$$x \text{ } e \text{ } z \text{ } devono \text{ } essere \text{ } diversi \text{ } ed \text{ } entrambi \text{ } pari$$

Quindi, $c = \langle V_c, \Delta_c \rangle$ dove $V_c = \{x, z\}$ e Δ_c può essere espresso in forma estensionale come segue

$$\Delta_c = \{(2,4), (2,6), (4,2), (4,6), (6,2), (6,4)\} \tag{4}$$

dove, come di consueto, è stato utilizzato l'ordinamento alfabetico per le variabili. Si noti che l'estensione di Δ_c ad un sottoinsieme di Δ può essere espressa in forma estensionale come segue

$$img(c) = \{(2,1,4), (2,2,4), \ldots, (6,6,4)\} \tag{5}$$

dove ogni elemento di Δ_c è stato utilizzato per produrre 6 elementi di $img(c)$. L'insieme degli assegnamenti parziali consistenti di c può essere espresso in forma intensionale come segue

$$\Delta_c = \{(x, z) : x \in I \wedge z \in I \wedge x \neq z \wedge x \bmod 2 = 0 \wedge z \bmod 2 = 0\} \tag{6}$$

e quindi

$$img(c) = \{(x, y, z) : x \in I \wedge y \in I \wedge z \in I \wedge x \neq z \wedge x \bmod 2 = 0 \wedge z \bmod 2 = 0\} \tag{7}$$

La possibilità di estendere gli insiemi degli assegnamenti parziali consistenti ad insiemi di assegnamenti totali consistenti permette di definire in modo semplice cosa si intenda per soluzione di un CSP. Dato un CSP $\mathcal{P} = \langle V, D, C \rangle$, l'insieme delle **soluzioni** di $\mathcal{P}$ è

$$sol(\mathcal{P}) = \bigcap_{c \in C} img(c) \tag{8}$$

l'intersezione di tutte le immagini di tutti i vincoli che stiamo considerando

e il $\mathcal{P}$ si dice **risolubile** (o **risolvibile**) se questo insieme non è vuoto. Dati due CSP $\mathcal{P}_1$ e $\mathcal{P}_2$ definiti sullo stesso insieme di variabili, si dice che $\mathcal{P}_1$ è **equivalente** a $\mathcal{P}_2$ se $sol(\mathcal{P}_1) = sol(\mathcal{P}_2)$. Si noti che questa definizione può essere estesa a problemi con lo stesso numero di variabili rinominando opportunamente le variabili dei problemi.

Dato un CSP è interessante cercare delle trasformazioni in grado di generare dei CSP ad esso equivalenti ma che siano più utili per la ricerca delle soluzioni. In generale, però,

sono interessanti anche le trasformazioni in grado di produrre dei CSP non equivalenti a quello di partenza ma che permettano, comunque, di studiare le proprietà di quest'ultimo.

Esempio 2. Si consideri un CSP $\mathcal{P}$ con due variabili x e y entrambe con dominio $[-10..10]$. Il CSP ha un unico vincolo descritto dalla proprietà $y = x^2$. Si noti che il CSP è risolubile perché, ad esempio, $x = -2$ e $y = 4$ è una soluzione. Il CSP $\mathcal{P}_1$ che restringe il dominio di y a $[0..10]$ è equivalente a $\mathcal{P}$ perché è facile vedere che entrambi ammettono lo stesso insieme di soluzioni. Viceversa, il CSP $\mathcal{P}_2$ che restringe anche il dominio di x all'insieme $[0..10]$ non è equivalente a $\mathcal{P}$ perché, ad esempio, $x = -2$ e $y = 4$ è una soluzione di $\mathcal{P}$ ma non è una soluzione di $\mathcal{P}_2$. Però, il CSP $\mathcal{P}_2$, che è ottenuto da $\mathcal{P}$ con una trasformazione detta **di rottura della simmetria** (o **symmetry breaking**), è comunque interessante perché una volta risolto è possibile ottenere le soluzioni di $\mathcal{P}$. In più, $\mathcal{P}_2$ ha un numero inferiore di elementi nel dominio di x e quindi la ricerca esaustiva delle soluzioni è facilitata.

Dato un CSP, ogni trasformazione che produce un CSP ad esso equivalente ma con meno vincoli o meno valori nei dominî viene detta **propagazione dei vincoli**. Se la trasformazione utilizza un sottoinsieme dei vincoli, allora viene detta **propagazione locale (dei vincoli)**. Se la trasformazione utilizza un sottoinsieme dei vincoli per rimuovere alcuni valori dai dominî delle variabili coinvolte nei vincoli considerati, allora viene detta **filtro (dei valori)**.

Normalmente, le trasformazioni tra CSP equivalenti vengono utilizzate per ottenere nuovi CSP con proprietà interessanti. Ad esempio, una trasformazione che spesso viene utilizzata permette di ottenere un CSP nodo-consistente equivalente ad un CSP dato. Si noti che un CSP si dice **nodo-consistente** (o **node-consistent**) se per ogni vincolo unario c vale la seguente proprietà

$$\forall v \in dom(x), v \in \Delta_c \tag{9}$$

se per ogni vincolo unario c, vale la proprietà che qualsiasi sia il valore nel dominio della variabile x coinvolta nel vincolo, il valore appartiene al dominio di c

dove x è l'unica variabile del vincolo c e Δ_c è l'insieme degli assegnamenti parziali consistenti di c.

Un **algoritmo di nodo-consistenza** lavora su un CSP per produrre un CSP nodo-consistente ad esso equivalente eliminando dai dominî delle variabili tutti i valori che non rispettano la definizione precedente. In più, un algoritmo di nodo-consistenza rimuove anche tutti i vincoli unari dal CSP perché, una volta eliminati i valori inconsistenti dai dominî, i vincoli unari non sono più necessari.

In modo simile, un **algoritmo di arco-consistenza** lavora su un CSP per produrre un CSP arco-consistente ad esso equivalente eliminando dai dominî delle variabili tutti i valori che non rispettano la definizione di arco-consistenza. Un CSP si dice **arco-consistente** (o **arc-consistent**) se per ogni vincolo binario c valgono le seguenti proprietà

$$\forall v_x \in dom(x), \exists v_y \in dom(y) : (v_x, v_y) \in \Delta_c$$
$$\forall v_y \in dom(y), \exists v_x \in dom(x) : (v_x, v_y) \in \Delta_c$$

qualsiasi sia v_x, elemento del dominio di x, esiste un v_y, elemento del dominio di y, tali che la coppia v_x e v_y è un elemento delle ple di questo vincolo binario
Per ogni v_y nel dominio di y, esiste un v_x appartenente al dominio di x, tali che la coppia v_x e v_y è un elemento delle ple del vincolo binario

dove x e y sono le due variabili del vincolo c e Δ_c è l'insieme degli assegnamenti parziali consistenti di c.

Si noti che esistono vari algoritmi di arco-consistenza che assumono che i dominî delle variabili siano finiti. L'algoritmo normalmente più utilizzato è AC-3, che ha una complessità temporale asintotica di caso pessimo di classe $O(e\,k^3)$, dove e è il numero di vincoli binari e k è la cardinalità massima dei dominî delle variabili coinvolte nei vincoli binari. Si noti, comunque, che la complessità temporale asintotica di caso pessimo di un algoritmo

di arco-consistenza ottimo è di classe $O(e\,k^2)$, a cui si può arrivare con l'algoritmo AC-4 mediante un significativo incremento della memoria utilizzata rispetto ad AC-3.

In più, si noti che la proprietà di arco-consistenza può essere estesa a vincoli globali introducendo la cosiddetta **iperarco-consistenza**. Infine, si noti che la nodo-consistenza coinvolge una sola variabile mentre la arco-consistenza coinvolge due variabili. Quindi è ragionevole aspettarsi di poter definire la cosiddetta **consistenza di percorso** costruendo percorsi di più di due variabili collegate da vincoli binari.

PRIMA di fare qua, è PASSATO alle SLIDE, appunti presi a pg. 8

2 Problemi con Vincoli Polinomiali

La consistenza di nodo e la consistenza di arco permettono di rimuovere valori dai dominî delle variabili e sono spesso efficaci, specialmente se i vincoli sono espressi in forma tabellare. Viceversa, se i vincoli sono espressi in forma intensionale, ad esempio perché i dominî delle variabili non sono finiti, vengono spesso utilizzate altre forme di consistenza. Un esempio interessante a questo riguardo è rappresentato dai vincoli espressi mediante equazioni, disequazioni e disuguaglianze tra polinomi a coefficienti di variabili reali.

Si consideri un CSP $\mathcal{P}$ con $n \in \mathbb{N}_+$ variabili in $V = \{x_i\}_{i=1}^n$ alle quali sono associati i dominî $dom(x_i) = [\underline{x_i}, \overline{x_i}]$ rappresentati da intervalli chiusi in $\mathbb{R}$. Il problema può contenere anche più variabili, ma quelle di V sono quelle interessanti per i vincoli espressi mediante polinomi. Un vincolo c sulle variabili $V_c \subseteq V$ di $\mathcal{P}$ si dice **polinomiale** se il suo insieme degli assegnamenti parziali consistenti può essere espresso in modo intensionale come

$$\Delta_c = \{\mathbf{x} \in dom(c) : p(\mathbf{x}) \odot q(\mathbf{x})\}$$

→ insieme delle variabili x_i tale per cui $p(x)$ con p polinomio è in una certa relazione del tipo $<, \leq, >, \geq, =, \neq$ con un altro polinomio $q(x)$

dove $\odot \in \{<, \leq, =, \neq, \geq, >\}$, $p : \mathbb{R}^k \to \mathbb{R}$ e $q : \mathbb{R}^k \to \mathbb{R}$ sono due funzioni polinomiali e $k = |V_c|$.

Si noti subito che ogni vincolo polinomiale può essere ridotto ad una delle due seguenti forme

$$\{\mathbf{x} \in dom(c) : p(\mathbf{x}) \geq 0\} \quad \text{oppure} \quad \{\mathbf{x} \in dom(c0 : p(\mathbf{x}) > 0\} \tag{13}$$

utilizzando semplici manipolazioni algebriche e sfruttando il fatto che, data una funzione polinomiale $p : \mathbb{R}^k \to \mathbb{R}$, vale, per ogni $\mathbf{x} \in \mathbb{R}^k$,

$$p(\mathbf{x}) = 0 \iff p(\mathbf{x}) \leq 0 \wedge p(\mathbf{x}) \geq 0 \tag{14}$$

$$p(\mathbf{x}) \neq 0 \iff p^2(\mathbf{x}) > 0 \tag{15}$$

Infine, si noti che è sempre possibile riscrivere i vincoli in modo che tutte le variabili abbiano dominî definiti come intervalli chiusi di $\mathbb{R}_+$ mediante opportune trasformazioni affini operate sulle variabili. Infatti, essendo tutti i dominî delle variabili degli intervalli chiusi, è sufficiente traslare tutti gli intervalli per garantire che le variabili abbiano dominî definiti come intervalli chiusi di $\mathbb{R}_+$.

Definizione: Un CSP si dice **consistente sugli intervalli** (o **bounds consistent**) se per ogni vincolo polinomiale c vale che per ognuna delle sue variabili x con dominio $[\underline{x}, \overline{x}]$ esiste almeno un assegnamento parziale consistente del vincolo che abbia il valore $\underline{x}$ per x ed, eventualmente un altro, assegnamento parziale consistente che abbia il valore $\overline{x}$ per x.

Quindi, indipendentemente dalla consistenza dei valori all'interno dell'intervallo $[\underline{x}, \overline{x}]$, la bounds consistency considera solo la consistenza dei valori ai due estremi di ogni intervallo coinvolto nel vincolo. Si noti che normalmente si è interessati al più piccolo intervallo, nel senso del contenimento, che garantisca la bounds consistency di una variabile in uno o più vincoli del problema.

Un algoritmo di bounds consistency è un algoritmo che trasforma un CSP in un secondo CSP ad esso equivalente in cui gli intervalli che definiscono i dominî delle variabili garantiscono la bounds consistency. Come già discusso, un algoritmo di bounds consistency può considerare solo variabili definite su $\mathbb{R}_+$ e vincoli in forma di disequazione, stretta o meno. In queste ipotesi, è possibile definire un opportuno algoritmo di bounds consistency sfruttando la seguente proposizione.

Proposizione 1. *Dato $n \in \mathbb{N}$, sia $p : \mathbb{R}^n \to \mathbb{R}$ una funzione polinomiale a coefficienti reali positivi, se $\underline{\mathbf{x}} \in \mathbb{R}^n_+$ e $\overline{\mathbf{x}} \in \mathbb{R}^n_+$, allora per ogni $\mathbf{v} \in [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$ vale*

$$p(\underline{\mathbf{x}}) \leq p(\mathbf{v}) \leq p(\overline{\mathbf{x}}) \tag{16}$$

nell'ipotesi non restrittiva che per ogni $1 \leq i \leq n$ valga $\underline{x}_i \leq \overline{x}_i$.

La descrizione generale di un algoritmo di bounds consistency che sfrutti la proposizione precedente richiede l'introduzione della notazione dei **multi-indici**. Quindi, anziché descrivere un algoritmo di bounds consistency nella sua generalità, verrà descritto sommariamente un metodo che permette di ridurre i domini delle variabili sfruttando il fatto che queste siano definite su intervalli chiusi di $\mathbb{R}_+$.

Si consideri vincolo descritto dalla proprietà $p(\mathbf{x}) \geq 0$ nel caso in cui p sia una funzione lineare a coefficienti reali di $n \in \mathbb{N}_+$ variabili

$$p(\mathbf{x}) = \sum_{i=1}^{n} a_i x_i + b \tag{17}$$

Spiegazione Lezione 16, 1:25:00

con, per semplicità $b \geq 0$. La proprietà che definisce il vincolo può essere riscritta spostando a destra tutti i termini di p con coefficienti negativi e quindi

$$\sum_{a_i > 0} a_i x_i + b \geq \sum_{a_i < 0} -a_i x_i \tag{18}$$

x sempre definiti positivi, il segno quindi lo decide a

Fissata una variabile di indice k, se $a_k > 0$ è possibile scrivere

$$\sum_{i \neq k, a_i > 0} a_i x_i + a_k x_k + b \geq \sum_{a_i < 0} -a_i x_i \tag{19}$$

Sfruttando la proposizione precedente è possibile ottenere

$$\sum_{i \neq k, a_i > 0} a_i x_i \leq \underbrace{\sum_{i \neq k, a_i > 0} a_i \overline{x_i} = u}_{\text{UPPER}} \qquad \underbrace{l = \sum_{a_i < 0} -a_i \underline{x_i}}_{\text{LOWER}} \leq \sum_{a_i < 0} -a_i x_i \tag{20}$$

e quindi la proprietà che descrive il vincolo impone che

$$x_k \geq \frac{l - u - b}{a_k} \tag{21}$$

per ogni variabile x_k che viene moltiplicata in p per un coefficiente positivo. Si noti subito che è semplice rimuovere l'ipotesi $b \geq 0$ e che, ragionamenti simili a quelli appena visti, consentono di identificare una disequazione da utilizzare per le variabili che vengono moltiplicate per un coefficiente negativo in p. Entrambe queste disequazioni possono anche essere utilizzate per trattare vincoli descritti mediante disequazioni strette, previo cambiamento del simbolo di disuguaglianza. Si capisce quindi facilmente come la possibilità di

stimare i valori massimi e minimi di un polinomio di una variabile in un intervallo chiuso sia di fondamentale importanza per gli algoritmi di bounds consistency.

Tutte queste disequazioni possono essere utilizzate ripetutamente per restringere i dominî delle variabili del vincolo considerato fino al raggiungimento di almeno una delle seguenti condizioni:

1. L'applicazione delle disequazioni non genera nuovi restringimenti dei dominî e quindi è possibile cercare le soluzioni del CSP nei nuovi dominî; oppure
2. L'applicazione delle disequazioni ha svuotato almeno uno dei dominî e quindi è stato dimostrato che il CSP non ammette soluzioni.

I due esempi che seguono mostrano come applicare questo metodo nei due casi appena descritti. In particolare, il seguente esempio consente di restringere i dominî delle variabili.

???**Esempio 3.** Si consideri un CSP con tre variabili x, y e z i cui dominî sono $dom(x) = dom(y) = dom(z) = [1, 10]$. Si consideri il vincolo definito dalla proprietà

$$2x + 3y - z \leq 1 \tag{22}$$

La proprietà che definisce il vincolo può essere riscritta come

verificato di avere intervalli positivi, coefficienti positivi e il termine noto dal lato con segno positivo

$$z + 1 \geq 2x + 3y \tag{23}$$

valuto il caso "sotto" $x, y = 1$ (per il dominio)

$z+1 \geq 2 \cdot 1 + 3 \cdot 1 > 5$

↳ e quindi z deve valere almeno 1, quindi riduco il dominio di $z = [4, 10]$

e quindi

$$z + 1 \geq 2x + 3y \geq 5 \tag{24}$$

da cui si evince che $z \geq 4$. Ma, ragionando sulla variabile x si ottiene ⟶ valutiamo quindi l'upper: $\bar{z} = 10$

$$11 \geq z + 1 \geq 2x + 3y \geq 2x + 3 \tag{25}$$

$11 \geq 10 + 1$

da cui si evince che $x \leq 4$. Infine, ragionando sulla variabile y si ottiene

$$11 \geq z + 1 \geq 2x + 3y \geq 2 + 3y \tag{26}$$

da cui si evince che $y \leq 3$. Si noti che questi ragionamenti possono essere iterati finché nessun dominio viene più modificato oppure finché almeno uno dei dominî non diventa l'insieme vuoto. Nel primo caso si è ottenuto un CSP bounds consistent equivalente al CSP iniziale, mentre nel secondo caso è stato dimostrato che il CSP iniziale non ammette soluzioni. Provando ad iterare i ragionamenti fatti si nota subito che i dominî non vengono ulteriormente ridotti e quindi è stato ottenuto un nuovo CSP, equivalente a quello iniziale ma bounds consistent, con tre variabili x, y e z aventi dominî

$$dom(x) = [1..4] \qquad dom(y) = [1..3] \qquad dom(z) = [4..10] \tag{27}$$

e un solo vincolo che richiede che valga la proprietà $2x + 3y - z \leq 1$. Si noti che il vincolo non può essere rimosso perché la proprietà di bounds consistency non fornisce informazioni sulla consistenza dei valori interni agli intervalli considerati. La bounds consistency studia unicamente gli estremi dei dominî delle variabili.

Il seguente esempio mostra come sia possibile stabilire che il CSP considerato non ammette soluzioni. Infatti, l'applicazione delle disequazioni studiate in precedenza consente di ridurre all'insieme vuoto il dominio di almeno una variabile, terminando il processo di riduzione e stabilendo che il CSP non ammette soluzioni.

Esempio 4. Si consideri un CSP con tre variabili x, y e z i cui dominî sono $dom(x) = dom(y) = dom(z) = [1, 10]$. Si consideri il vincolo definito dalla proprietà

$$12x + 8y - z \leq 1 \tag{28}$$

La proprietà che definisce il vincolo può essere riscritta come

$$z + 1 \geq 12x + 8y \tag{29}$$

e quindi

$$z + 1 \geq 12x + 8y \geq 20 \tag{30}$$

da cui si evince che il vincolo richiede che $z \geq 19$. Però, $z \leq 10$ per ipotesi e quindi è stato facilmente dimostrato che il CSP non ammette soluzioni.

Si noti che il metodo per la riduzione dei dominî delle variabili descritto sommariamente nel caso di funzioni lineari può essere esteso facilmente al caso di funzioni polinomiali non lineari ricordando che, nelle ipotesi considerate, se una variabile x ha per dominio $dom(x) = [\underline{x}, \overline{x}] \subseteq \mathbb{R}_+$, allora se $m \in \mathbb{N}_+$

$$\underline{x}^{m-1} x \leq x^m \leq \overline{x}^{m-1} x \tag{31}$$

e quindi i termini non lineari del tipo x^m possono essere ridotti a termini lineari. In modo simile, se una seconda variabile y ha dominio $dom(y) = [\underline{y}, \overline{y}] \subseteq \mathbb{R}_+$, allora

$$\underline{x}y \leq xy \leq \overline{x}y \quad \text{e} \quad \underline{y}x \leq xy \leq \overline{y}x \tag{32}$$

e quindi i termini non lineari del tipo xy possono essere ridotti a termini lineari.

I vincoli polinomiali vengono spesso utilizzati restringendo i dominî delle variabili a intervalli in $\mathbb{Z}_+$ e richiedendo che i coefficienti delle funzioni polinomi siano in $\mathbb{Z}$. In questo caso si parla di **vincoli polinomiali a dominî finiti** ed è possibile sfruttare le particolarità di questo tipo di vincoli per definire degli algoritmi di propagazione dei vincoli efficaci ed efficienti. In particolare, il fatto che le variabili abbiano dominî in $\mathbb{Z}_+$ e il fatto che i coefficienti delle funzioni polinomiali siano in $\mathbb{Z}$ permette di ridurre lo studio di questo tipo di vincoli a sole disequazioni non strette perché vale

$$p(\mathbf{x}) > 0 \iff p(\mathbf{x}) - 1 \geq 0 \tag{33}$$

per una qualsiasi funzione polinomiale a coefficienti interi e variabili intere positive. In più, si noti che, siccome le variabili sono ristrette ad assumere solo valori interi, in questo caso viene adottata la notazione degli intervalli interi e si scrive $[\underline{x}..\overline{x}]$, con $\underline{x} \in \mathbb{Z}$, $\overline{x} \in \mathbb{Z}$ e $\underline{x} \leq \overline{x}$, per indicare l'intervallo contenente tutti gli interi maggiori o uguali a $\underline{x}$ e minori o uguali a $\underline{x}$. Naturalmente, $[\underline{x}..\overline{x}] = \emptyset$ se $\underline{x} > \overline{x}$.

LEZIONE 15: SLIDE DA 50 A 58

Tutte le volte che si vuole cercare di risolvere un CSP, la cosa più semplice che si possa fare è utilizzare una tecnica che prende il nome di Generate & test, che è la tecnica più generale ma più sfortunata, ma più tornare utile quando si vuole confrontare il risultato di una tecnica raffinata, nell'ipotesi che si possano enumerare i valori nei domini delle variabili in modo efficace.
Quindi partiamo dall'ipotesi che il dominio delle variabili siano abbastanza piccoli (come cardinalità) da poter enumerare tutti i valori possibili in memoria, per ogni variabile noi avremo tutti i valori possibili, scritti da qualche parte.
1) affronteremo sempre CSP con variabili con domini finiti, che però nella pratica sono quelli più utilizzati;
2) utilizziamo domini non troppo grandi, pratica abbastanza comune anche nel quotidiano;
Ipotizzando i punti 1 e 2 possiamo ipotizzare di applicare il G&T, che comunque richiederà troppo tempo ma ci può aiutare a capire dove andremo a finire.
(Vedi codice slide 51)
G&T è una possibilità, non è detto che sia quella ottima, non è detto che non si possa fare meglio: nonostante sia una tecnica grezza, nel caso pessimo di complessità è ottima.
Standard Backtracking è una tecnica che modificando effettivamente pochissimo codice, si hanno guadagni di performance enormi.
L'idea dello SBT è non fare le verifiche dei vincoli solo quando ho un assegnamento completo, tutte le variabili sono state assegnate ad un valore, ma anche quando ho gli assegnamenti parziali (almeno una variabile è stata assegnata) a questo punto posso verificare su quella variabile tutti i vincoli unari, che coinvolgono quella variabile, se sono tutti rispettati vado avanti, assegno la seconda, verifico i vincoli unari sulla singoli e i vincoli binari sulla coppia, assegno una terza, verifico i vincoli unari, binari e globali; continuo così fino a che qualcosa fallisce (e quindi torno indietro, faccio backtracking) oppure arrivo in fondo.
L'unica verifica che faccio è che la verifica del soddisfacimento dei vincoli, lo metto dentro all'assegnamento: assegno la variabile al valore, se i vincoli non sono violati, vado avanti e quindi index ++; altrimenti butto l'assegnamento e riassegno e riprovo.
Quando index vale n, allora ho fatto un assegnamento completo.
Come complessità temporale sono uguali (G&T), come complessità di calcolo sembra pure peggio.

LEZIONE 16: SLIDE DA 59 A 77

Fare nodo consistenza togliere i valori da i domini che non rispettano i valori unari, una volta tolti questi valori, i vincoli unari non servono più e quindi vanno tolti; fare nodo consistenza può essere visto come far è un filtro iniziale.
Attenzione che fare arco consistenza, toglie altri valori e che quindi nuovi vincoli unari si debbano ritogliere.
Una versione più leggera dell'arco consistenza è il forward checking: stessa idea dell'arco consistenza, prendo dei vincoli binari e verifico che ogni valore abbia un supporto, ma contrariamente a quello che fa l'arco consistenza, che quando si toglie un valore pewrchè non ha un supporto, viene riapplicata a tutte le variabili; il forward checking si limita a dire: guardo avanti, se c'è qualcosa da togliere, lo tolgo ma una volta fatto ciò, non riparto a cercare di togliere altre cose. Questo fa si che dipenda molto il risultato dall'ordine in cui si sceglie come fare forward checking, volendo si mette un euristica che sceglie il nome delle variabili ma in realtà quello che si fa di solito è fare forward checking per fare una cosa veloce poco impattante dal punto di vista computazionale che toglie qualche valore.