Cenni di Programmazione con Vincoli

Domenico Salvagnin

2011-12-10

La programmazione con vincoli (*Constraint Programming*, CP) è lo studio dei modelli computazionali basati su vincoli.

1 Problemi di soddisfacibilitá con vincoli

Informalmente, un problema di soddisfacibilitá con vincoli (*constraint satisfaction problem*, CSP) è un problema composto da un insieme finito di variabili, ciascuna con dominio finito, e da un insieme di vincoli che restringono i valori che le variabili possono assumere contemporaneamente. Lo scopo è quello di assegnare un valore del dominio ad ogni variabile, soddisfando tutti i vincoli.

Vediamo ora delle definizioni più precise.

Definizione 1.1. *Il dominio di una variabile è l'insieme di tutti i possibili valori che tale variabile può assumere. Se x_i è una variabile, indicheremo il suo dominio con D_i.*

Definizione 1.2. *Un vincolo C_j è una coppia (R_j, S_j) dove R_j è una relazione nelle variabili S_j, dette* scope *di C_j. In altre parole, R_j è un sottoinsieme del prodotto cartesiano dei domini delle variabili in S_j.*

Si noti come, dal punto di vista concettuale, un vincolo non è altro che un sottoinsieme del prodotto cartesiano dei domini di un insieme di variabili, cioè una lista degli assegnamenti parziali che sono "accettati" dal vincolo e che lo rendono soddisfatto (rappresentazione *estensionale*). In pratica però un vincolo può essere espresso in molti altri modi equivalenti (e generalmente molto più compatti), quali funzioni o disuguaglianze (rappresentazione *intensionale*).

Definizione 1.3. *Un CSP è definito da un tripletta $P = (X, D, C)$, dove $X = (x_1, \ldots, x_n)$ è un insieme finito di variabili, $D = (D_1, \ldots, D_n)$ è il corrispondente insieme di domini (cioè si ha $x_i \in D_i$) e $C = (C_1, \ldots, C_m)$ è un insieme finito di vincoli.*

Definizione 1.4. *Una soluzione (o assegnamento completo) di un CSP P è una n-pla $A = (a_1, \ldots, a_n)$ tale che $a_i \in D_i$ per ogni i.*

Definizione 1.5. *Una soluzione A si dice ammissibile se ogni vincolo C_j è soddisfatto, cioè la proiezione di A sullo scope S_j di C_j appartiene a R_j, per ogni j.*

A seconda dei casi, risolvere un CSP equivale a:

- trovare una soluzione ammissibile, senza alcuna preferenza
- trovare tutte le soluzioni ammissibili

- trovare una soluzione ottima, rispetto ad una opportuna funzione obiettivo, cioè una funzione che assegna ad ogni soluzione un valore numerico. Si parla in tal caso di problema di ottimizzazione con vincoli (*Constraint Optimisation Problem*, COP).

2 Metodi di risoluzione: ricerca e inferenza

I principi di base per la risoluzione di CSP si dividono essenzialmente in due categorie: ricerca e inferenza.

2.1 Ricerca

Dal punto di vista teorico, la risoluzione di un CSP mediante ricerca sistematica è un problema banale, in quanto è sufficiente generare tutte le possibili soluzione e poi controllare quali siano ammissibili (tale schema molto semplice prende il nome di *generate-and-test*, GT). Dal punto di vista pratico questo metodo è però inaccettabile, in quanto il numero di soluzioni da testare è esponenziale e il metodo è inutilmente inefficiente.

Un miglioramento sostanziale dell'algoritmo GT è la ricerca basata su backtracking (BT), in un cui un assegnamento parziale viene esteso incrementalmente scegliendo dei valori per le variabili fino a che non si raggiunge un conflitto (cioè la soluzione parziale corrente non può essere estesa in modo da soddisfare i vincoli del problema). Quando ciò accade, l'ultima assegnamento viene annullato e se ne prova un altro. Questo viene fatto un modo sistematico, in modo da enumerare (implicitamente) tutte le possibili soluzioni.

L'algoritmo BT è un miglioramento di GT in quanto controlla che i vincoli possano essere soddisfatti ad ogni scelta, senza aspettare di avere un assegnamento completo per controllare. La ricerca BT viene spesso rappresentata mediante un albero di ricerca, dove ad ogni nodo è associata una soluzione parziale. La complessità al caso peggiore è comunque esponenziale.

L'algoritmo standard BT ha i seguenti svantaggi:

- *thrashing*, cioè pu'o generare ripetutamente assegnamenti parziali che sono scartati per lo stesso motivo
- lavoro ridondante, in quanto assegnamenti di variabili che risultano incompatibili non vengono memorizzati
- un conflitto viene individuato solo quanto capita, ma non prima

Nel corso degli anni sono state proposte varie migliorie dell'algoritmo BT di base per mitigare tali svantaggi, molte delle quali basate su tecniche di inferenza.

2.2 Inferenza

Le tecniche di inferenza portano alla luce informazioni nascoste. Quando vengono applicate ad un insieme di vincoli, posso dedurre dei vincoli validi che erano solo impliciti nella formulazione di partenza. Tali nuovi vincoli possono rivelare in modo più diretto che alcune regioni dello spazio delle soluzioni non contengono soluzioni ammissibili (o ottime) e pertanto possono migliorare le prestazioni di un algoritmo di ricerca.

La tecnica di inferenza principale prende il nome di propagazione dei vincoli (*constraint propagation*). Tale propagazione consiste nel rimuovere esplicitamente dal dominio delle variabili valori

(o combinazioni di valori) che non possono essere completati ad una soluzione ammissibile, in quanto alcuni vincoli ne risulterebbero necessariamente violati. Ad esempio, dato un problema con variabili x_1 e x_2, entrambe con valori interi in $\{1, \ldots, 10\}$ e con il vincolo $|x_1 - x_2| > 5$, allora la propagazione può rimuovere i valori 5 e 6 dai domini di entrambe le variabili.

Le tecniche di propagazione dei vincoli possono differire per il livello di coerenza (*consistency*) che mantengono nel problema: in generale livelli di consistency più elevati richiedono un maggiore sforzo computazionale durante la propagazione.

Quasi tutte le tecniche di propagazione *non* sono metodi di inferenza completi (come le tecniche di ricerca) e pertanto non possono essere usate da sole per risolvere un CSP. Lo schema generale consiste pertanto nel mantenere un certo livello di coerenza all'interno di un algoritmo di ricerca di tipo BT, effettuando la propagazione dei vincoli ad ogni nodo dell'albero decisionale. L'applicazione di tecniche di propagazione può ridurre drasticamente la dimensione dell'albero di ricerca.

2.3 Vincoli globali

Un vincolo globale (*global constraint*) è un vincolo che descrive una relazione che coinvolge un numero non-fisso di variabili. Ad esempio, il noto vincolo `all-different`$(x_1, \ldots, x_n)$ è soddisfatto solo se i valori assegnati alle variabili $x_1, \ldots, x_n$ sono tutti diversi tra loro.

Un vincolo globale è spesso semanticamente ridondante, in quanto può essere espresso tramite un insieme di vincoli più semplici (ad esempio, `all-different`$(x_1, \ldots, x_n)$ può essere espresso mediante $x_i \neq x_j$ per ogni $i \neq j$), ma il fatto di poter descrivere dei pattern di vincoli di particolare interesse pratico in modo chiaro e compatto semplifica notevolmente la fase di modellazione. Ancora più importante (e meno ovvio) è il fatto che molti vincoli globali forniscono al risolutore una visione più accurata della struttura del problema e possono migliorarne drasticamente l'efficienza, in quanto possono essere in grado di effettuare propagazioni che non sono possibili altrimenti. Ad esempio, dato un CSP con 3 variabili $x_1 \in \{1, 2\}$, $x_2 \in \{1, 2\}$ e $x_3 \in \{1, 2, 3\}$, il vincolo `all-different`(x_1,x_2,x_3) deduce correttamente $x_3 = 3$, cosa che invece non è possibile con i vincoli $x_1 \neq x_2$, $x_2 \neq x_3$ e $x1 \neq x_3$.

La lista dei vincoli globali è in continuo aumento, e con essa l'espressività ed efficacia del paradigma CP per la risoluzione di problemi di ottimizzazione combinatoria.

Ecco alcuni dei vincoli globali più comuni:

- `element`. Siano y e z variabili e c un array di variabili, cioè $c = [x_1, \ldots, x_n]$. Il vincolo è soddisfatto se $z = x_y$. Più formalmente:

$$\texttt{element}(y, z, x_1, \ldots, x_n) = \{(e, f, d_1, \ldots, d_n) | e \in D_y, f \in D_z, d_i \in D_{x_i} \forall i, f = d_e\}$$

- `all-different`. Siano $x_1, \ldots, x_n$ variabili. Il vincolo è soddisfatto se tali variabili assumono valori tutti diversi tra loro.

$$\texttt{all-different}(x_1, \ldots, x_n) = \{(d_1, \ldots, d_n) | d_i \in D_{x_i} \forall i, d_i \neq d_j \forall i \neq j\}$$

- `gcc`. Il vincolo di *global cardinality constraint* è una generalizzazione di `all-different`. Mentre il primo richiede che ogni valore sia assegnato ad al più una variabile, il vincolo `gcc` richiede che ogni valore v_i sia assegnato almeno l_i volte e al più u_i volte. Dato un valore v_i, sia `occ`(v_i,t) il numero di volte che v compare in t. Il vincolo `gcc` si può specificare come:

$$\texttt{gcc}(x_1, \ldots, x_n, v_1, \ldots, v_m, l_1, \ldots, l_m, u_1, \ldots, u_m) = \{d = (d_1, \ldots, d_n) | l_i \leq \texttt{occ}(v_i, d) \leq u_i \forall i\}$$

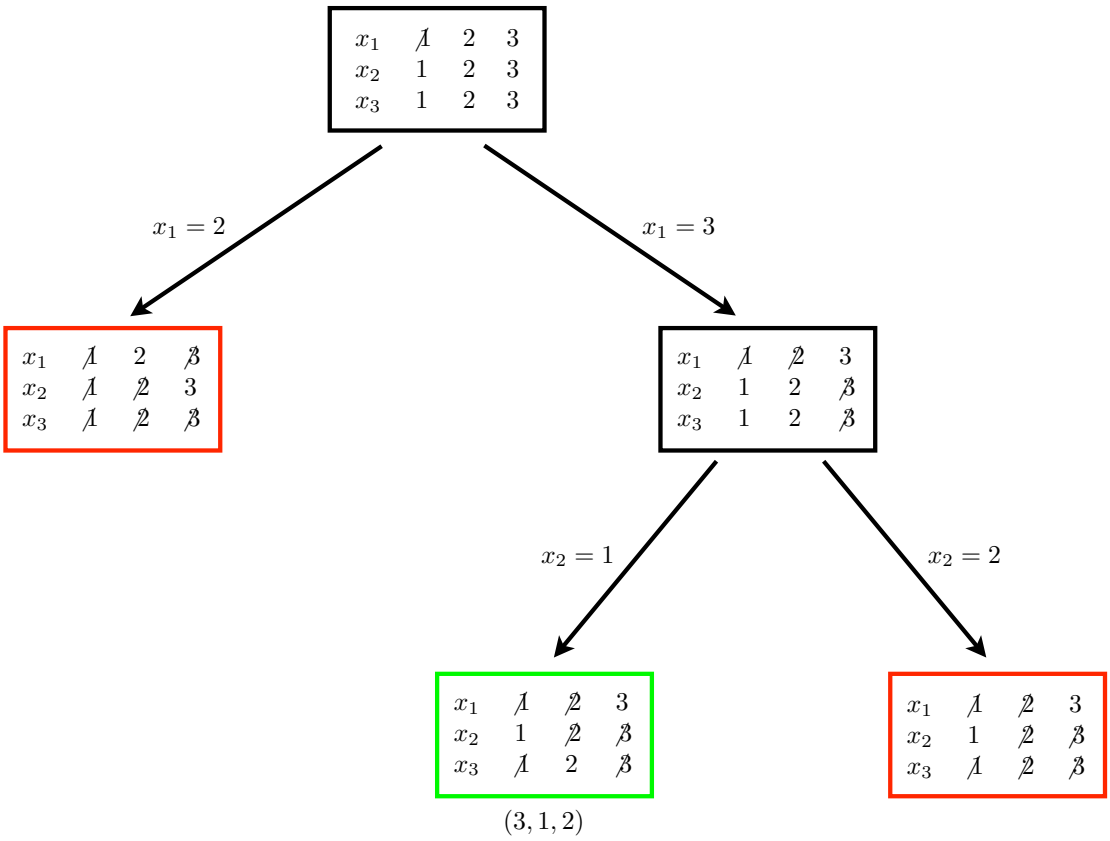
- `regular`. Tale vincolo è soddisfatto se la stringa $d_1 \dots d_n$ formata dai valori assegnati alle variabili $(x_1, \dots, x_n)$ viene accettata da un automa a stati finiti M, in altre parole la stringa appartiene al linguaggio regolare $L(M)$ accettato da M.

$$\texttt{regular}(x_1, \dots, x_n, M) = \{(d_1, \dots, d_n) | d_1 \dots d_n \in L(M)\}$$

Un esempio di ricerca di tutte le soluzioni ammissibili del problema

$$\begin{cases} \texttt{all-different}(x_1, x_2, x_3) \\ 2x_1 + x_2 \geq 6 \\ x2 \leq x_3 \\ x_1, x_2, x_3 \in \{1, 2, 3\} \end{cases}$$

è illustrato nella seguente figura



Si noti come un approccio GT avrebbe dovuto generate tutte e 27 le possibili soluzioni, e anche un approccio BT *senza* propagazione avrebbe impiegato più di 20 nodi per risolvere il problema, contro i soli 5 con la propagazione dei vincoli attiva.