Lezione12

Table of contents

Constraint Satisfaction Problems (CSPs)

Un problema di soddisfacimento dei vincoli (CSP) è un problema di ricerca; stesso concetto che risiede negli alberi di ricerca e nei grafi di ricerca con la differenza che

- questi sono abbastanza *generali* da essere interessanti da prendere in considerazione, e sono
- sufficientemente specifici da poter usare delle *euristiche* che permettano la risoluzione dei problemi in modo efficace.

Un CSP è caratterizzato da 3 componenti:

- un insieme di *variabili*, simboli, a cui attribuiamo il significato di variabile (un qualche cosa il quale valore deve essere identificato)

$$V = \{X_1, X_2, \dots, X_n\}$$

- un insieme di *domini*, insieme di valori qualsiasi non necessariamente uguali e non necessariamente diversi, a cui è associata una variabile

$$D = \{\mathtt{dom}(X_1), \mathtt{dom}(X_2), \dots, \mathtt{dom}(X_n)\}$$

- un insieme di *vincoli*, dato dall'associazione tra variabili e domini, sottoinsieme del prodotto cartesiano dei domini di tutte le variabili

$$C \subseteq \mathtt{dom}(X_1) \times \mathtt{dom}(X_2) \times \cdots \times \mathtt{dom}(X_n)$$

Una variabile può assumere qualsiasi valore all'interno del suo dominio, purché siano rispettati tutti i vincoli. L'insieme dei vincoli e variabili è sempre *finito*, i domini sono tutti non vuoti dato un CSP iniziale: in parole più semplici, stiamo parlando di **sistemi di equazioni**.

Un vincolo si dice:

- *unario*, se definito su un'unica variabile, di tutti i domini di una variabile, ce ne interessa uno soltanto (oppure, tutti i valori all'interno dei domini di tutte le variabili tranne una sono possibili, per l'unica in cui così non è, andremo a specificare le possibilità)

> ≡ $x \in \mathbb{R} > 5$ **identifica il sottoinsieme dei reali ($\mathbb{R}$) strettamente maggiori di cinque (> 5). È un vincolo unario, unica variabile.**

- *binario*, se definito su due variabili

  > ≡ $x + y \in \mathbb{R} > 7$ **identifica il sottoinsieme dei reali ($\mathbb{R}$) tale per cui la sommatoria di 2 varibili ($x + y$) è maggiore di 7 (> 7)**

- *globale*, se richiede la presenza di più due variabili

Ognuno dei vincoli considerati non è da pensarsi isolato, se troviamo un'associazione tra una variabile e un suo dominio e tutti i vincoli che vi appartengono sono soddisfatti, allora il CSP è risolubile e quella che abbiamo trovato è una soluzione.
Il **sistema di equazioni lineare** è un esempio di CSP, che come abbiamo imparato in Algebra e Geometria, può avere 0, 1 o infinite soluzioni. Ricordiamo che l'importanza del soddisfacimento non sta tanto nel trovare l'unica soluzione che risolva il problema, piuttosto nell'interesse nel dimostrare che esiste una soluzione e che quindi il sistema sia risolvibile.

Esempio di CSP

Consideriamo il seguente CSP fornito da 2 simboli

$$V = \{X_1, X_2\}$$

Il dominio per X_1 e X_2 è quello definito da

$$\texttt{dom}(X_1) = [1..100], \quad \texttt{dom}(X_2) = \texttt{dom}(X_1)$$

I vincoli associatovi sono 2 e sono

$$X_1 < 5, \quad X_1 + X_2 = 10$$

Osservando la situazione, notiamo che

- per il primo vincolo si assicurano valori del $\texttt{dom}(X_1)$ essere compresi tra $\{1..4\}$
- per il secondo vincolo tuttavia, vale $X_2 = 10 - X_1$ e quindi il $\texttt{dom}(X_2)$ sarà compreso dei valori $\{6..9\}$

L'insieme delle soluzioni è prodotto cartesiano dei domini delle variabili, le cui dimensioni sono rispettivamente 4 e 4. Quindi le possibili soluzioni saranno 16:

$$(X_1 = 1, X_2 = 6), (X_1 = 1, X_2 = 7), (X_1 = 1, X_2 = 8), (X_1 = 1, X_2 = 9)$$
$$(X_1 = 2, X_2 = 6), (X_1 = 2, X_2 = 7), (X_1 = 2, X_2 = 8), (X_1 = 2, X_2 = 9)$$
$$\dots$$

Soluzioni e assegnamenti

Un **assegnamento** per un CSP, è la mappatura tra alcune variabili, a un rispettivo dominio. Un assegnamento può essere:

- *parziale*, nel caso non vengano coinvolte tutte le variabili;
- *totale*, se coinvolge tutte le variabili;
- *inconsistente*, se almeno un vincolo viene violato/non rispettato;
- *consistente*, se rispetta tutti i vincoli del problema.

Un assegnamento totale e consistente, è una soluzione.
Giungere alla soluzione lo possiamo fare estendendo gli assegnamenti e garantendo la consistenza: partiamo assegnando una variabile (parziale), scegliendo il valore di questa in modo che sia consistente; si sceglie poi una seconda variabile estendendo l'assegnamento e avanti così si va finché non raggiungiamo un insieme di assegnamenti che rendano il problema risolvibile.

Tipi di CSPs

Nel paragrafo sopra avviamo visto come una soluzione sia ricercabile per domini finiti.
Nella pratica comune, questo tipo di CSP è il più ricercato, ma ciò non toglie il fatto che esistano anche quelli a domini infiniti: in questo caso in avanti, ipotizzeremo limitandoci fortemente, che i domini saranno a prescindere finiti.

- CSPs con domini *discreti*
  - domini finiti con cardinalità minore o uguale a d
  - insieme infinito contabile di domini ($\mathbb{N}$, $\mathbb{Z}$, stringhe,...)

  Limitare i domini a insiemi finiti non è comunque da sottovalutare dal punto di vista computazionale, siccome n variabili portano a $\mathcal{O}(d^n)$ assegnamenti completi, dove qual'ora d sia uguale per tutti, la complessità sarà pari a $\mathcal{O}(n)$.
- CSPs con domini *continui*
  - insieme infinito non contabile ($\mathbb{R}$, $\mathbb{C}$, ...)
  - vincoli normalmente espressi in termini di equazioni, disuguaglianze, ...

  Sempre problemi di soddisfacimento interessanti, ma che informatica e matematica tendono a lasciare da parte in quanto "casi particolari".

Map coloring

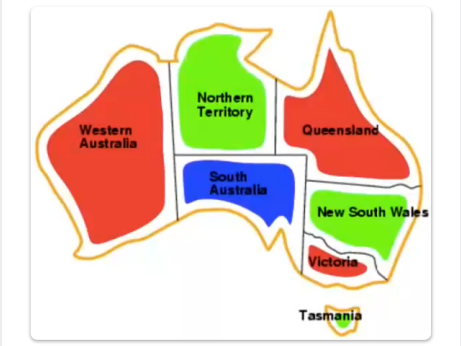
Mappa australiana con stati bianchi

Un problema famoso, perché ripreso nel libro "Artificial Intelligence: A Modern Approach" di Russel e Norvig, è quello del colorare la mappa dell'Australia (WA, NT, Q, NSW, V, SA, T) utilizzando solo 3 colori (R,G,B).

Abbiamo una serie di variabili (gli stati) e una serie di domini (red, green, blue) da associare: scelto un colore da usare per colorare Western Australia (WA), poniamo che suddetto colore sia diverso da tutti gli altri con i 2 vincoli

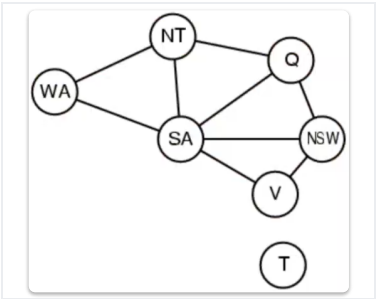
- `WA ≠ NT, WA ≠ SA, NT ≠ SA, NT ≠ Q, SA ≠ Q, ...`
- "zone adiacenti devono avere colore diverso".

Il numero di assegnamenti possibili sono il prodotto cartesiano: 3^6.
Da notare che quello che stiamo dicendo non è "tutti i colori devono essere diversi", quanto piuttosto "tutti i colori adiacenti devono essere diversi" (la Tasmania può assumere qualsiasi colore).



Mappa australiana con stati colorati

Constraint Graph



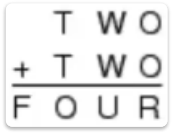
CSP della mappa australiana

La soluzione colorata vista sopra, non per forza è l'unica per il problema: potremmo partire a colorare WA con il verde, o più semplicemente potremmo colorare la T di rosso.

Per trovare le soluzioni, usiamo uno strumento utile che prende nome di **grafo dei vincoli**, costruibile una volta che abbiamo un CSP binario. Per ogni nodo del grafo dei vincoli abbiamo una variabile, per ogni arco che collega i nodi abbiamo un vincolo che coinvolge le variabili. Vincoli unari corrisponderanno ad auto-anelli, vincoli di variabili con se stesse. Vincoli che coinvolgono più di 2 nodi saranno componibili costruendo ipergrafi, semplificabili in grafi binari (consuetudine).

Se il CSP binario non è facilmente componibile perché sono presenti vincoli per più di 2 variabili, tecnicamente il grafo non riusciamo a farlo se non introducendo dei nodi diversi dalle variabili CSP.

L'esempio che vediamo risale da una categoria di problemi molto studiata nella letteratura dei problemi di soddisfacimento dei vincoli che prende il nome di "cripto-aritmetica" o "cript-aritmetica", la cui idea è la stessa dei giochi presenti nei giornalini di "Settimana enigmistica" che ben conosciamo.



(Osserviamo l'immagine) Ci viene detto che la somma delle lettere (TWO), porta alle lettere sotto (FOUR). Ogni lettera della sommatoria, è una cifra che non conosciamo ma che possiamo determinare stabilendo dei vincoli tra le variabili (T, W, O, F, U, R) e il loro rispettivo dominio ($[0..9]$). Il vincolo più significativo è quello dove s'impone che tutte le lettere devono avere cifre a esse associate, diverse, con la dicitura `alldifferent`.

- Variabili

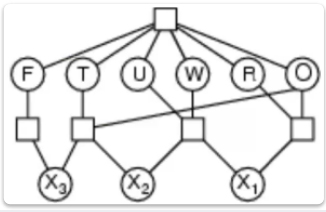
$$T, W, O, F, U, R$$

$$X_1, X_2, X_3$$

- Dominio

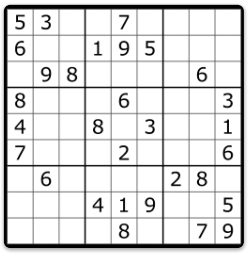
$$D = [0..9]$$

- Vincoli
  - $O + O = R + 10X_1$
  - $X_1 + W + W = U + 10X_2$
  - $X_2 + T + T = O + 10X_2$
  - $X_3 = F, T \neq 0, F \neq 0$
  - $\texttt{alldifferent}(F, T, U, W, R, O)$



Esempio di grafo dei vincoli

A Real-World Case: Sudoku Puzzles



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Un sudoku

Se fino ai paragrafi precedenti abbiamo visto come usare i CSP nei casi reali, dovrebbe sorgere semplice applicare il concetto alla scacchiera del sudoku, no?

Il problema che si nasconde nella realtà dei fatti, è la facilità con cui una serie di condizioni possono "esploderci" tra le mani: se dovessimo elencare tutte le combinazioni possibili per le 50 celle di variabili del sudoku (alcune già riempite), potremmo non finire più e sbagliarci (luogo in cui risiede la difficoltà).

> ⓘ **"sudoku" è la traduzione italianizzata della pronuncia originale "so-u-do-ku", o scritto in kanji come 数独 (数(すう): "numero","cifra" | 独(どく): "singolo","solo").**

I sudoku ben formati sono CSP che sempre e comunque hanno un'unica soluzione, algoritmi per la costruzione di questa soluzione esistono e possono essere classificati come logici o come "bruti". Ad ogni modo, seguendo un ragionamento, la soluzione per un sudoku al giorno d'oggi, può essere trovata in meno di un secondo.

Real-World CSPs

Un altro esempio di problema può essere quello dell'allocamento di aule per i professori: allocare in un ambiente compatto, con tanti vincoli di non sovrapposizione è necessario per non creare conflitti.

Configurare un prodotto su un sito di e-commerce, che sia una tazza, un vestito o un poster, è anche lui un CSP.

Calendario dei tempi di trasporto, configurazione hardware e tempi di fabbrica sono tutti CSP che anche con poche variazioni delle variabili, portano a complessità esagerata se la soluzione cercata con metodi forzati, senza seguire la logica.

Generate & Test (G&T)

Cerchiamo almeno un algoritmo che ci permetta di trovare una soluzione, uno generale che sia in grado di generare soluzione consistente e verificabile.

L'algoritmo Generate & Test di cui stiamo parlando è formato da 2 passi: il passo di generazione d'assegnamento completo e il passo di test della consistenza di questo assegnamento.
È molto semplice da realizzare; si pesca ciascuna variabile dal lago delle possibilità e si verifica se questa è presente in tutti i vincoli: se lo è abbiamo trovato una soluzione, se quello costruito non è consistente allora si riprova con nuove combinazioni.

La ricerca esaustiva è il caso pessimo per una ricerca, perché stiamo cercando un assegnamento consistente che viene verificato solo alla fine del ciclo, motivo per cui non viene utilizzato per problemi complessi (soluzione computabile ma in tempi assurdi).

```
void solveGT(int index) {
        if(index == n) {
                if(areConstraintNotViolated())
                        solutionFound();
        } else {
                if(variable[index] =/= FREE)
                        solveGT(index + 1);
                else {
                        for(int i=0; i < d; i++) {
                                variable[index] = dom[i];
                                solveGT(index + 1);
                        }
                        variable[index] = FREE;
                }
        }
}
```

Ipotizziamo di avere un vettore di variabili per cui ciascun valore corrisponde a uno del dominio, numerati tra 0..i e `dom[i]` l'i-esimo valore del dominio. Con `index` identifichiamo la variabile da usare nella costruzione dell'assegnamento, che se corrisponde alla grandezza del vettore, vuole significare che abbiamo riempito una classe che è pronta per essere verificata.

Se la verifica dei vincoli con `areConstraintNotViolated()` va a buon fine, allora abbiamo trovato una soluzione; altrimenti procediamo.
Se la variabile `index` non è libera (`variable[index] =/= FREE`), perché ha già un valore, allora passiamo alla successiva e assegnamo questa un valore.

Importante da notare come la variabile venga ogni ciclo liberata (`variable[index] = FREE`) per poter permettere al ciclo successivo di lavorare sulla variabile successiva.

L'algoritmo così procede finché tutti i vincoli non vengono rispettati.

Standard Backtracking (SBT)

L'algoritmo G&T può essere modificato in modo drastico, per quanto concerne le prestazioni, con un solo cambiamento di linea di codice.
Il controllo dei vincoli con `areConstraintNotViolated()` non necessariamente lo dobbiamo fare quando tutte le variabili sono pronte, ma possiamo farlo ogni qual volta che una soluzione parziale viene trovata e le variabili riempite esaminate.

La riga di controllo viene semplicemente spostata qualche linea sotto, generando un nuovo algoritmo detto di tracciamento all'indietro, standard backtracking (SBT).

```
// ...
                for(int i=0; i < d; i++) {
                        variable[index] = dom[i];

                        if(areConstraintNotViolated())
                                solveSBT(index + 1);
                        }
                        variable[index] = FREE;
                }
        }
}
```

Se ci sono dei vincoli verificabili non soddisfatti, viene provato un assegnamento successivo; se i vincoli invece sono tutti verificabili, procediamo di 1.

Assegnate n variabili, tutte le volte abbiamo garantito che l'assegnamento è consistente e che quindi abbiamo un assegnamento completo e consistente, soluzione del problema.

Nella risoluzione di un sudoku, SBT funziona molto bene perché ritornante spesso `false` al controllo dei vincoli; usare quindi un G&T sarebbe detrimentale alle prestazioni, per questo tipo di problema.

Se lo guardiamo attentamente, il nostro algoritmo forma un albero di ricerca dove i nodi figlio sono espansi prima del genitore, in ampiezza (depth-first). L'occupazione di memoria è lineare in termini di memoria ($\mathcal{O}(n)$), con n numero di variabili finite.
La costruzione dell'albero termina alla fine della ricerca.

04/04/2023