Problemi di soddisfacimento di vincoli

Maria Simi
a.a. 2012/2013

Problemi di soddisfacimento di vincoli (CSP)

- Sono problemi con una struttura particolare, per cui conviene pensare ad algoritmi specializzati
- È un esempio di rappresentazione fattorizzata, in cui si comincia a dire qualcosa sulla struttura dello stato
- Esistono euristiche generali che si applicano e che consentono la risoluzione di problemi significativi per questa classe
- La classe di problemi formulabili in questo modo è piuttosto ampia: layout di circuiti, scheduling, …

Formulazione di problemi CSP

- *Problema*: descritto da tre componenti
  1. $X= \{X_1 X_2 \dots X_n\}$ insieme di variabili
  2. $D= \{D_1 D_2 \dots D_n\}$ insieme di domini
     $D_i=\{v_1, \dots, v_k\}$ è l' insieme dei valori possibili per la var X_i
  1. $C= \{C_1 C_2 \dots C_m\}$ insieme di vincoli (relazioni tra le variabili)
- *Stato:* un assegnamento parziale di valori a variabili
  $\{X_i = v_i, X_j = v_j \dots\}$
- *Stato iniziale*: { }
- *Azioni*: assegnamento di un valore a una variabile
- *Soluzione (goal test)*: un assegnamento completo (le variabili hanno tutte un valore) e consistente (i vincoli sono tutti soddisfatti)

Colorazione di una mappa

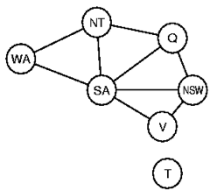
$X = \{WA, NT, SA, Q, NSW, V, T\}$

$D_{WA} = D_{NT} = D_{SA} = D_Q = D_{NSW} = D_V = D_T = \{red, green, blue\}$

$C = \{WA \neq NT, WA \neq SA, NT \neq Q, NT \neq SA, SA \neq Q, SA \neq NSW, SA \neq V, NSW \neq V\}$

Colorazione di una mappa

Grafo dei vincoli



Altri problemi

- Il problema delle 8 regine
  - $X=\{Q_1, \dots, Q_8\}$
  - $D_i=\{1, 2, 3, 4, 5, 6, 7, 8\}$
  - C, i vincoli di non attacco
- Il problema di pianificare una serie di lavori
  - X i tempi di inizio dei lavori da compiere
  - D tempi in un certo intervallo (numero finito)
  - C, es. $X_i + d_i < X_j$ *dove d_i è la durata del lavoro i*
    esempio vincoli di mutua esclusione

Tipi di problemi CSP

- Variabili con domini discreti e finiti
  - CSP booleani (valori *vero* e *falso*)
- Variabili con domini discreti e infiniti
- Variabili con domini continui (ricerca operativa)
  - programmazione lineare

Tipi di vincoli

- I vincoli possono essere:
  - unari (es. "x pari")
  - binari (es. "x > y")
  - di grado superiore (es. x+y = z): *questi possono essere ricondotti a vincoli binari introducendo nuove variabili*
- Vincoli globali
  - Es. TuttiValoriDiversi, come nel Sudoku
- Vincoli assoluti o di preferenza
  - Problemi di ottimizzazione di vincoli

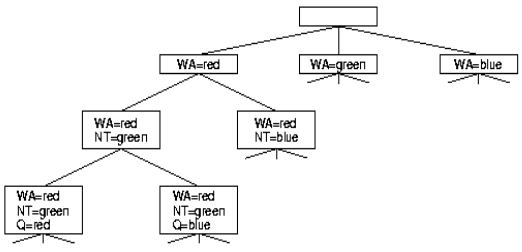
Ricerca in problemi CSP

- Ad ogni passo si assegna una variabile
  - La massima profondità della ricerca è fissata dal numero di variabili *n*
- L'ampiezza dello spazio di ricerca è
  $|D_1| \times |D_2| \times \ldots \times |D_n|$
  dove $|D_i|$ è la cardinalità del dominio di X_i
- Il fattore di diramazione pari alla dimensione media dei domini *d* (e non $nd + (n-1)d + \ldots$)
- Riduzione drastica dello spazio di ricerca dovuta al fatto che il *goal-test* è decomponibile e commutativo

Strategie di ricerca

- *Generate and Test*. Si genera una soluzione e si controlla: poco efficiente
- *Ricerca con backtracking* (BT): ad ogni passo si assegna una variabile ma si controllano i vincoli ogni volta e si torna indietro in caso di fallimento
  - *Controllo anticipato* della violazione dei vincoli: è inutile andare avanti fino alla fine e poi controllare; si può fare *backtracking* non appena si scopre un vincolo violato.
  - La ricerca è naturalmente limitata in profondità dal numero di variabili

Esempio di backtracking



Backtracking ricorsivo per CSP

function Ricerca-Backtracking (csp) **return** *una soluzione o fail*
  return Backtracking-Ricorsivo({ }, *csp*) *//un assegnamento vuoto*

function Backtracking-Ricorsivo(*ass*, *csp*) **return** *una soluzione o fail*
  if ass *è completo* **then return** *ass*
  var ← Scegli-var-non-assegnata(*csp*)
  for each *val* **in** Ordina-Valori-Dominio(var, ass, csp) **do**
    if *val consistente con* ass **then**
      aggiungi [*var=val*] *a* ass
      risultato ← Backtracking-Ricorsivo(*ass, csp*)
      If risultato ≠ fail **then return** risultato
      rimuovi [*var=val*] *da ass*
  return fail

Euristiche e strategie per CSP

- Scegli-var-non-assegnata: Quale variabile scegliere?
- Ordina-Valori-Dominio: Quali valori scegliere?
- Qual è l'influenza di un assegnamento sulle altre variabili? Come restringe i domini?
  - ➔ *propagazione di vincoli*
- Come evitare di ripetere i fallimenti?
  - ➔ b*acktracking intelligente*

Scelta delle variabili

1. MRV (*Minimum Remaining Values* o *fail-first*): scegliere la variabile che ha meno valori possibili, la variabile *più vincolata*. Si scoprono prima i fallimenti
2. Euristica *del grado:* scegliere la variabile coinvolta in più vincoli con le altre variabili (la *variabile più vincolante* o di *grado maggiore*)

   Da usare a parità di MRV

Scelta dei valori

- Una volta scelta la variabile come scegliere il valore da assegnare?

1. Valore ***meno vincolante***: quello che esclude meno valori per le altre variabili direttamente collegate con la variabile scelta

   Se ci serve una sola soluzione, meglio valutare un assegnamento che ha più probaibilità di successo

Propagazione di vincoli

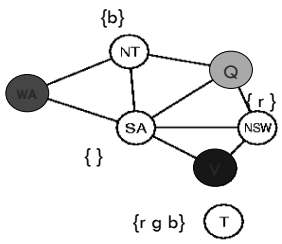
1. Verifica in avanti (*Forward Checking* o *FC*)
   - assegnato un valore ad una variabile si possono eliminare i valori incompatibili per le altre var. *direttamente collegate* da vincoli (non si itera)
2. Consistenza di nodo e d'arco
   - si restringono il valori dei domini delle variabili tenendo conto dei vincoli unari e binari su tutto il grafo (si itera finché tutti i nodi ed archi sono consistenti)

Esempio di FC

WA=r

Q=g

V=b



Stesso esempio in forma tabellare

	WA	*NT*	*Q*	*NSW*	*V*	*SA*	*T*
Initial domains	R G B	R G B	R G B	R G B	R G B	R G B	R G B
After *WA=red*	Ⓡ	G B	R G B	R G B	R G B	G B	R G B
After *Q=green*	Ⓡ	B	Ⓖ	R B	R G B	B	R G B
After *V=blue*	Ⓡ	B	Ⓖ	R	Ⓑ		R G B

Consistenza di nodo

- Un nodo è consistente se tutti i valori nel suo dominio soddisfano i vincoli unari
- Una rete di vincoli è *nodo-consistente* se tutti i suoi nodi sono consistenti
- I vincoli unari quindi possono essere risolti restringendo opportunamente i domini delle variabili

Consistenza degli archi

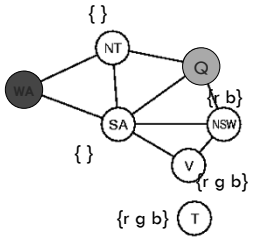
- Nel grafo di vincoli, un arco orientato da X a Y, $X \rightarrow Y$, è *consistente* se per ogni valore *x* di X c'è almeno un valore *y* di *Y* consistente con x.
- Se un arco $X \rightarrow Y$ non è consistente si cerca di renderlo tale, rimuovendo valori dal dominio di X
- Si itera fino a che tutti gli archi sono consistenti
- Un metodo più efficace di FC per propagare i vincoli.
- Algoritmo MAC (Maintaining Arc Consistency, AC-3): controlla la consistenza degli archi all'inizio e dopo ogni assegnamento

Esempio di MAC

WA=r

Q=g

Si scopre subito che non va bene



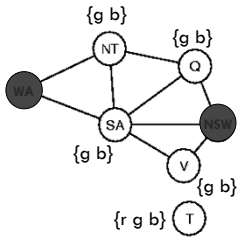
Complessità di MAC (o AC-3)

- Devono essere controllati tutti gli archi (supponiamo c - max k^2, se *k* è il numero di variabili)
- Se durante il controllo di un arco $X \rightarrow Y$ il dominio di X si restringe vanno ricontrollati tutti gli archi entranti $Z \rightarrow X$
- Il controllo di consistenza di un arco ha complessità d^2, se d è la dimensione dei domini
- Un arco deve essere controllato al max *d* volte
- Complessità: $O(c\, d^3)$ ... polinomiale

MAC incompleto

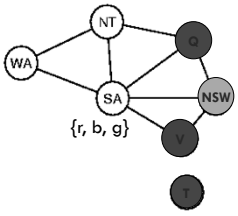
- Più efficace di forward-checking, ma non rileva tutte le inconsistenze Esempio:

  WA=red
  NSW=red
  ... non viene rilevata inconsistenza



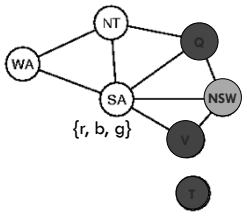
Backtracking cronologico

- Supponiamo di avere {Q=red, NSW=green, V=blue, T=red}
- Cerchiamo di assegnare SA
- Il fallimento genera un backtracking "cronologico"
- … e si provano tutti i valori alternativi per l'ultima variabile, T, continuando a fallire



Backtracking intelligente

- Si considerano alternative solo per le variabili che hanno causato il fallimento {Q, NSW, V}, *l'insieme dei conflitti*
- *Backtracking guidato dalle dipendenze*



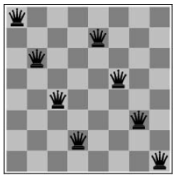
CSP con miglioramento iterativo

- Si parte con tutte le variabili assegnate (tutte le regine sulla scacchiera) e ad ogni passo si modifica l'assegnamento ad una variabile per cui un vincolo è violato (si muove una regina minacciata su una colonna).
- È un algoritmo di *riparazione euristica*.
- Un'euristica nello scegliere un nuovo valore potrebbe essere quella dei *conflitti minimi*: si sceglie il valore che crea meno conflitti.

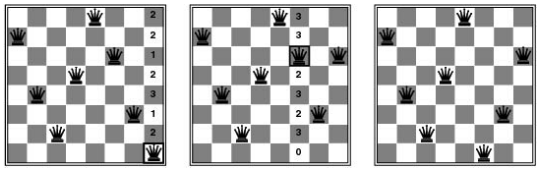
Le 8 regine come CSP

Formulazione come CSP:

- V_i: posizione della regina nella colonna i-esima
- D_i: {1 … 8}
- Vincoli di "*non-attacco*" tra V_1 e V_2: {<1,3> <1,4> <1,5>… <1,8><2,4><2,5> …<2,8> …}



Esempio



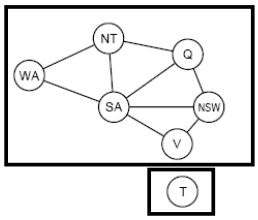
- *Molto efficace*: 1 milione di regine in 50 passi!
- Utilizzabile in problemi reali di progettazione e *scheduling*.

Min-conflicts: ricerca locale e online

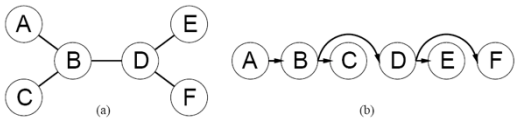
- La strategia dei conflitti minimi è un metodo locale, particolarmente efficiente
- Può essere usato in un ambiente online in cui la situazione cambia nel tempo, come nei problemi reali di schedulazione.

Sottoproblemi indipendenti

- n # variabili
- c # variabili per sottoproblema
- d dimensione domini
- n/c problemi indipendenti
- $O(d^c\, n/c)$ *lineare* in n piuttosto che $O(d^n)$ *esponenziale*!



Struttura dei problemi: albero



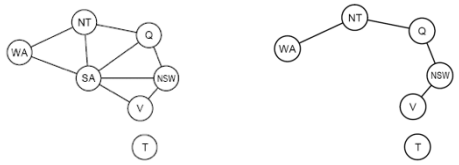
- In un grafo di vincoli ad albero, due variabili sono collegate da un solo cammino (a)
- Scelto un nodo come radice, l'albero induce un ordinamento topologico sulle variabili (b)
- Consistenza d'arco orientato (DAC)
  - Dato un ordinamento per le variabili: X_1, X_2, … X_n ogni arco $X_i \rightarrow X_j$ con i < *j* è consistente

Algoritmo basato su DAC

1. Procedendo da X_n a X_2 rendere gli archi $X_i \rightarrow X_j$ consistenti *riducendo il dominio di* X_i se necessario
2. Procedendo da X_1 a X_n assegnare i valori

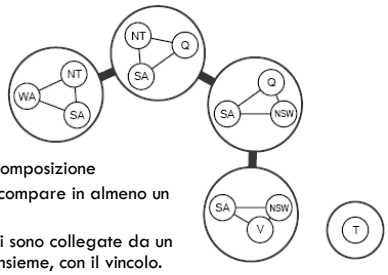
- Complessità: $O(nd^2)$, lineare in *n*

Riduzione ad albero



- Es. Assegnare SA, e ridurre i domini delle variabili collegate. Provare con diversi valori di SA.
- In generale eliminare un insieme [min] S di variabili, fino a ottenere un albero (*insieme di taglio dei cicli*) e provare con tutti gli assegnamenti possibili di S.
- *Condizionamento con insieme di taglio*

Scomposizione ad albero



- Requisiti della scomposizione
  - ogni variabile compare in almeno un sottoproblema
  - se due variabili sono collegate da un vincolo vanno insieme, con il vincolo.
  - se una variabile compare in due sottoproblemi deve anche comparire nei sottoproblemi sul cammino che le congiunge

Soluzione

- Ogni sotto-problema viene risolto in maniera indipendente
- Possiamo vedere il problema originario come un *Mega-problema* con la seguente formulazione:
  - *Mega-variabili* in corrispondenza a sotto-problemi, con dominio le soluzioni ai sottoproblemi

    Es. Dom(X1) ={[WA=r, SA=b, NT=g] ...} 6 sol.

  Vincoli: i valori assegnati alle variabili nei diversi sotto-problemi devono essere gli stessi