# Problemi di soddisfacimento di vincoli

*Maria Simi*
*a.a. 2010/2011*

## Problemi CSP: soddisfacimento di vincoli

- Sono problemi con una struttura particolare, per cui conviene pensare ad algoritmi specializzati.
- La classe di problemi formulabili in questo modo è piuttosto ampia: layout di circuiti, scheduling, …
- Esistono euristiche generali che si applicano e che consentono la risoluzione di problemi significativi per questa classe

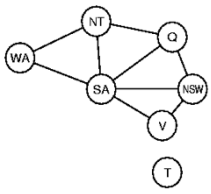
## Formulazione di problemi CSP

- *Problema*:
  - insieme di variabili ($X_1$ $X_2$ … $X_n$) con associato dominio ($D_1$ $D_2$ … $D_n$)
  - insieme di vincoli (relazioni tra le variabili): $C_1$ $C_2$ … $C_m$
- *Stato:* un assegnamento parziale di valori a variabili

  $\{X_i = v_i , X_j = v_j \ldots\}$
- *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 soddisfatti)

## Colorazione di una mappa


*Grafo dei vincoli*




## Formulazione

- Variabili, i paesi nella mappa:
  - WA, NT, SA, Q, V, NSW, T
- Domini: i colori possibili
  - r, g, b
- Vincoli:
  - WA ≠ NT, WA ≠ SA, NT ≠ SA, NT ≠ Q, SA ≠ Q, SA ≠ NSW, SA ≠ V, NSW ≠ Q, NSW ≠ V

## Tipi di problemi CSP

- Variabili discrete con domini finiti o infiniti
  - CSP booleani (valori *vero* e *falso*)
- Variabili con domini continui:
  - programmazione lineare
- I vincoli possono essere:
  - unari (es. "x pari")
  - binari (es. "x > y")
  - di grado superiore (es. x+y = z)
- Vincoli assoluti o di preferenza

## 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

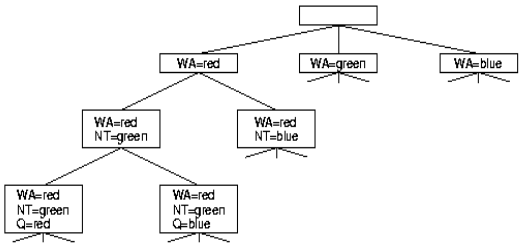
  $|D_1| \times |D_2| \times \ldots \times |D_n|$

  dove $|D_i|$ è la cardinalità del dominio di $X_i$
- Il fattore di diramazione pari alla dimensione massima dei domini *d* (e non *nd* + (*n*-1)*d* + ...)
- Riduzione drastica dello spazio di ricerca dovuta al fatto che il *goal-test* è decomponibile e commutativo

## Strategie di ricerca

- *Ricerca con backtracking* (BT): ad ogni passo si assegna una variabile e si torna indietro in caso di fallimento (vs. *Generate and Test*).
- *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

**function** Ricerca-Backtracking (csp) **returns** *una soluzione o fail*
  **return** Backtracking-Ricorsivo({ }, csp)

**function** Backtracking-Ricorsivo(ass, csp) **returns** *una soluzione o fail*
  **if** ass *è completo* **then return** ass
  var ← Scegli-var-non-assegnata(Variabili[csp], ass, csp)
  **for each** val **in** Ordina-Valori-Dominio(var, ass, csp) **do**
    **if** val *consistente con* ass *in base a* Vincoli[csp] **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

- Quale variabile scegliere?
- Quali valori scegliere?
- Qual è l'influenza di una scelta sulle altre variabili? (*Propagazione di vincoli*)
- Come evitare di ripetere i fallimenti? (*Backtracking intelligente*)

## Scelta delle variabili

1. MRV (*Minimum Remaining Values*): scegliere la variabile che ha meno valori possibili, la variabile *più vincolata*. Si scoprono prima i fallimenti
2. In base *al 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

## Propagazione di vincoli

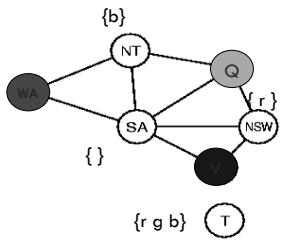
1. Verifica in avanti (*Forward Checking* o FC): assegnato un valore ad una variabile si possono eliminare i valori incompatibili per le altre var. collegate da vincoli (un passo solo)
2. *Propagazione di vincoli:* si itera il *FC*; se una variabile ha il suo dominio ristretto per effetto del *forward checking* si vanno a controllare le variabili collegate …

## 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* | (R) | G B | R G B | R G B | R G B | G B | R G B |
| After *Q=green* | (R) | B | (G) | R B | R G B | B | R G B |
| After *V=blue* | (R) | B | (G) | R | (B) | | R G B |

## Consistenza degli archi

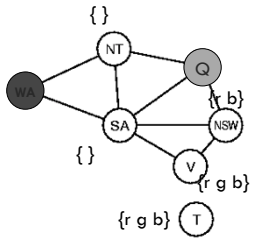
- Un metodo efficace per propagare i vincoli.
- Nel grafo di vincoli, un arco orientato da A a B è c*onsistente* se per ogni valore x di A c'è almeno un valore *y* di B consistente con x.
- Quello che si fa è controllare la consistenza degli archi all'inizio e dopo ogni assegnamento (MAC – Maintaining Arc Consistency)

## Esempio di MAC

WA=r

Q=g

Si scopre subito che non va bene



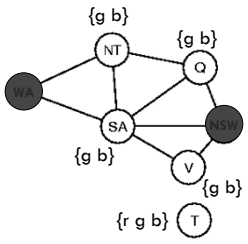
## Complessità di MAC (o AC-3)

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

## MAC incompleto

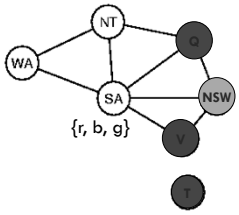
- Più efficace di forward-checking, n non rileva tutte le inconsistenze Esempi

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



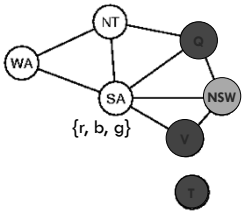

## Backtracking cronologico

- 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*



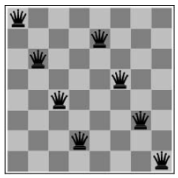

## CSP con miglioramento iterativo

- 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.

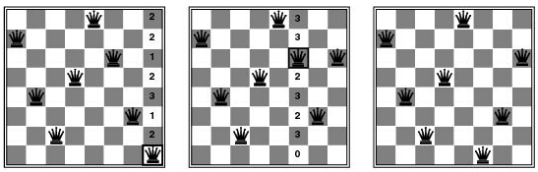
## Le 8 regine come CSP

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

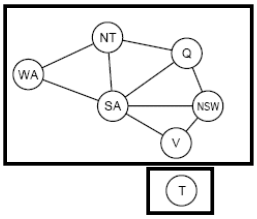


- *Molto efficace*: 1 milione di regine in 50 passi!
- Utilizzabile in problemi reali di progettazione e *scheduling*.

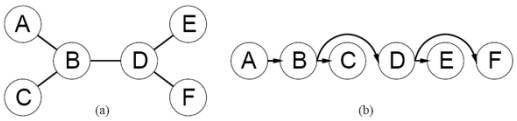
## Sottoproblemi indipendenti

- *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*!

*2 componenti connessi*

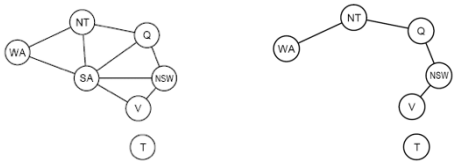



## Struttura dei problemi: albero



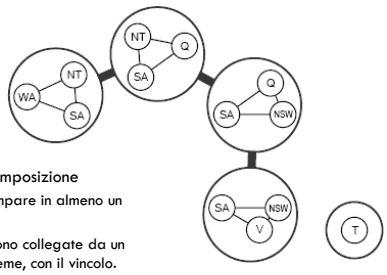

- Consistenza d'arco orientato (DAC)
  1. Ordinamento variabili: $X_1$, $X_2$, … $X_n$
  2. Da $X_n$ a $X_2$ verificare la consistenza degli archi $X_i \rightarrow X_j$ *riducendo il dominio di* $X_i$ se necessario
  3. Da $X_1$ a $X_n$ assegnare i valori
- Complessità: O($nd^2$), lineare in *n*

## 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
  1. ogni variabile compare in almeno un sottoproblema
  2. se due variabili sono collegate da un vincolo vanno insieme, con il vincolo.
  3. se una variabile è in due sottoproblemi deve anche comparire nei sottoproblemi sul cammino che le congiunge

## Soluzione

- Ogni sotto-problema viene risolto in maniera indipendente
- Sotto-problemi più piccoli possibile
- 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 soluzioni
  - Vincoli: i valori assegnati alle variabili nei diversi sotto-problemi devono essere gli sono gli stessi