Pseudocodici degli algoritmi

September 5, 2011

1 Prima fase: trovare una soluzione

Pseudocodice per l'algoritmo di ricerca di una soluzione che soddisfi tutti i vincoli. Questo algoritmo istanzia una variabile alla volta riducendo i domini delle variabili non ancora instanziate per rendere la ricerca piu' efficiente.

Si assume che ci siano n variabili da istanziare. La procedura sol cerca una instanziazione di tutte le variabili che non violi nessun vincolo. La procedura riduci-domini elimina dai domini delle variabili non ancora istanziate i valori che sono in contrasto con le istanziazioni gia' fatte. Lo pseudo-codice assume che le variabili siano ordinate e non dice come scegliere il prossimo valore per istanziare una variabile.

```
TYPE domini =  ARRAY [1..n] OF sets;
     istanziazione = ARRAY [1..n] OF INTEGERS;
VAR inst: istanziazione;
    fallimento: BOOLEAN;

PROCEDURE sol(j: INTEGER; D: domini; VAR successo: BOOLEAN);
BEGIN
  WHILE D[j] e' non vuoto AND NOT successo DO
    scegli d da D[j];
    D[j] := D[j] - {d};
    IF (inst, j=d) non viola nessun vincolo THEN
      inst[j] := d;
      successo := (j=n);
      IF NOT successo THEN
        riduci-domini(j,D,fallimento);
        IF NOT fallimento THEN
          sol(j+1,D,successo)
        END
      END
    END
  END
END
```

```
PROCEDURE riduci-domini(j: INTEGER; VAR D: domini;
               VAR fallimento: BOOLEAN);
VAR k: INTEGER;
BEGIN
  fallimento := FALSE;
  k := j+1;
  WHILE k e' diverso da n+1 AND NOT fallimento DO
    D[k] := {d in D[k] tale che (inst, xk=d) non viola nessun vincolo}
    fallimento := (D[k] = insieme vuoto);
    k := k+1
  END
END
```

La chiamata iniziale della procedura sol e' la seguente:

```
BEGIN
  successo := FALSE;
  riduci-domini(0,D,fallimento);
  IF NOT fallimento THEN sol(1,D,successo)
END
```

2 Seconda fase: migliorare la soluzione

Per definire l'algoritmo:

- definire la funzione di valutazione delle soluzioni (esempi: somma dei costi dei vincoli violati)
- definire il vicinato di una soluzione, e quindi le mosse ammissibili (esempio: scambiare due studenti allocati in corsi diversi, o scambiare le date di inizio di due corsi)
- definire come effettuare la scelta nel vicinato (esempio: la soluzione migliore tra quelle nel vicinato, la prima soluzione del vicinato migliore o di qualita' uguale alla corrente, la prima soluzione del vicinato che superi una certa qualita')
- definire le condizioni di terminazione (esempio: limite di tempo, o di numero di mosse, o di qualita' della soluzione)

Si parte da una soluzione iniziale, ottenuta ad esempio con l'algoritmo della sezione precedente. Poi si ripetono i seguenti passi:

1. Se la condizione di terminazione e' soddisfatta, ci si ferma e si ritorna la soluzione corrente (la migliore soluzione trovata fino a quel punto);
2. Considerare la soluzione corrente s (la prima ad esempio puo' essere ottenuta con l'agoritmo della sezione precedente)

3. Passare ad un'altra soluzione nel vicinato di s usando una mossa ammissible e secondo il criterio di scelta nel vicinato

4. Se la si trova, ripartire dal passo 2 con la nuova soluzione; altrimenti, ritornare al passo 2 con una soluzione scelta a caso dal vicinato