Unione europea
Fondo sociale europeo

MINISTERO DEL LAVORO
E DELLE POLITICHE SOCIALI
Direzione Generale per le Politiche
per l'Orientamento e la Formazione

REGIONE DEL VENETO

Algoritmo proposto

Maria Silvia Pini, Francesca Rossi, K. Brent Venable

Dipartimento di Matematica Pura e Applicata

Università di Padova

Algoritmo proposto

- L'agoritmo che proponiamo
  - Parte da una soluzione (cioe' un assegnamento di valori alle variabili) che soddisfa i vincoli irrinunciabili
  - Poi migliora la soluzione tenendo conto delle preferenze
- Considereremo separatamente le due fasi
  - Prima fase: trovare una soluzione
  - Seconda fase: migliorare la soluzione

Prima fase: trovare una soluzione (1)

- L'algoritmo
  - Considera una variabile alla volta in un certo ordine (dopo vedremo quale)
  - Cerca un valore dal suo dominio che possa essere assegnato a tale variabile senza violare nessun vincolo irrinunciabile (dopo vedremo come)
  - Dopo ogni istanziazione cerca di ridurre i domini delle variabili non ancora istanziate eliminando quei valori che sono in contrasto con le istanziazioni gia' fatte

Prima fase: trovare una soluzione (2)

- Per ogni variabile, per ogni suo possibile valore, i vincoli vengono testati sulla istanziazione corrente, e
  - se qualcuno e' violato si passa al prossimo valore
  - altrimenti si istanzia la variabile, si riducono i domini e si passa alla prossima variabile
- Se si sono provati tutti i valori di una variabile senza successo, si ritorna all'ultima variabile istanziata e si cerca un nuovo valore per lei, finche' non vengono istanziate tutte le variabili oppure finche' siamo ritornati indietro fino alla prima variabile
  - Se tutte le variabili sono state istanziate, abbiamo trovato una soluzione
  - Se siamo ritornati indietro fino alla prima variabile, il problema non ha soluzione

Prima fase: trovare una soluzione (2)

- Ordinamento delle variabili
  - L'ordine in cui le variabili vengono considerate influisce molto sull'efficienza dell'algoritmo
  - L'euristica piu' usata considera come prossima variabile da istanziare quella che ha
    - il piu' piccolo valore di (Dominio/Grado)
      - Dominio: e' il numero di elementi rimasti nel dominio della variabile
      - Grado: e' il numero di vincoli che coinvolgono quella variabile
  - Quindi si cerca di istanziare la variabile che ha
    - numero piu' piccolo di elementi nel suo dominio
    - il numero piu' grande di vincoli che la coinvolgono

Prima fase: trovare una soluzione (3)

- Ordinamento dei valori
  - Anche l'ordine in cui vengono considerati i valori nel dominio di una variabile influisce sull'efficienza dell'algoritmo (ma meno dell'ordine delle variabili)
  - In generale, e' bene considerare i valori in ordine di quanto sono promettenti rispetto alla non violazione dei vincoli
    - Esempio: se c'e' un vincolo sul valore massimo di una certa variabile, allora e' meglio scegliere i suoi valori partendo dal piu' piccolo

Prima fase: strategie di ricerca in Choco (1)

- In Choco per trovare una soluzione che soddisfa i vincoli irrinunciabili dobbiamo
  - Definire un modello
  - Definire le variabili e i vincoli
  - Aggiungere le variabili e i vincoli al modello
  - Definire un solver
  - Far leggere al solver il modello
  - Far partire la ricerca

```
Model m= new CPmodel();
…;
Solver s=new CPsolver();
s.read(m);
s.solve() ;
```

Prima fase: strategie di ricerca in Choco (2)

- Una strategia di branching pre-definita
  - DomOverWDegBranchingNew
    - E' una strategia di branching n-aria che assegna valori distinti ad una variabile intera
    - Ad ogni variabile sono associati tre valori
      - Dom: taglia del dominio corrente
      - Deg: numero corrente dei vincoli che coinvolgono quella variabile non ancora istanziati
      - W: somma dei costi associati con questi Deg vincoli
    - La strategia seleziona la variabile con
      - **piu' piccola ratio r = Dom/W*Deg**
    - Le tie sono rotte in modo random

      DomOverWDegBranchingNew(Solver s, IntDomainVar[] vars, VallIterator vallt, Number seed)

Prima fase: strategie di ricerca in Choco (3)

- Alcune delle strategie di selezione del valore da assegnare alla variabile in Choco
  - MaxVal
    - seleziona il piu' grande valore nel dominio della variabile
  - MinVal
    - seleziona il piu' piccolo valore nel dominio della variabile
  - RandomIntValSelector
    - RandomIntValSelector: seleziona un valore random nel dominio della variabile intera

Seconda fase: migliorare la soluzione (1)

- La prima fase
  - restituisce una soluzione che soddisfa i vincoli irrinunciabili
- La seconda fase
  - restituisce una soluzione di qualita' "alta" rispetto alla soddisfazione dei vincoli soft
- La qualita' di una soluzione e' definita da una funzione obiettivo F che e' uguale alla somma dei costi dei vincoli soft violati
- Soluzioni migliori hanno valori minori di funzione obiettivo
- Indichiamo con F* il valore massimo della funzione obiettivo che deve avere una soluzione per essere considerata di buona qualita'

Seconda fase: migliorare la soluzione (2)

- La prima fase del nostro algoritmo restituisce una prima soluzione s1 di qualita' F1
- Se F1≤F* allora l'algoritmo termina restituendo s1
- Altrimenti la seconda fase dell'algoritmo cerca una soluzione con funzione obiettivo ≤F* ricordandosi sempre la soluzione migliore trovata fino ad ora
  - Finche' trova una soluzione s=(x1=v1, …,xn=vn) con valore F'>F*
    - Sceglie (ad esempio a caso) alcune variabili da "liberare" e tiene gli assegnamenti delle altre variabili come nella soluzione s
    - Aggiunge al solver i vincoli xi=vi, presi da s, relativi alle variabili non liberate
    - Richiama il risolutore (s.solve) sul nuovo problema che ha
      - Le stesse variabili e domini di prima
      - Vincoli xi=vi= in piu' rispetto a prima.
  - In pratica si parte da una istanziazione parziale, e si chiede alrisolutore di completare l'istanziazione della variabili in modo da soddisfare I vincoli
  - Il risolutore ritornera' una nuova soluzione, e la funzione obbiettivo dira' qual e' la sua qualita'.
  - Se migliore della migliore trovata fino ad ora, si tiene, altrimenti si butta via.
  - L'algoritmo termina quando
    - Ha trovato una soluzione di qualita' ≤F* . In tal caso la restituisce
    - Ha esaminato tutte le soluzioni e nessuna e' di qualita' F (oppure ha raggiunto un limite di tempo). In tal caso restituisce la soluzione migliore trovata

Sommario

- Per effettuare le due fasi dell'algoritmo, basta quindi avere un risolutore che sappia trovare una soluzione che soddisfi i vincoli
- Questo risolutore verra' chiamato
  - La prima volta partendo dal modello del problema con nessuna variabile istanziata
  - Le volte successive partendo dal modello piu' l'istanziazione di alcune variabili
  - Queste istanziazioni vengono prese dalla soluzione trovata nell'iterazione precedente, dove vengono "liberate" alcune delle variabili, scelte ad aesempio a caso
- E' inoltre necessario saper scrivere una funzione obbiettivo che ci dica la qualita' di una soluzione (o meglio il costo di una soluzione, da minimizzare)
  - Per ogni vincolo "soft", dobbiamo scrivere un fattore della somma che costituira' la funzione obbiettivo
  - Questo fattore dovra' rappresentare la penalita' da pagare per quanto riguarda quel vincolo soft
  - Eesempio: se ogni classe deve avere almeno 20 studenti, ma e' meglio averne circa 25, possiamo scrivere il fattore |25-nstud| che dara' costo 0 se ci sono 25 studenti e costo crescente a mano che ci si allontana da 25. Dovremo poi considerare una normalizzazione dei costi dati dai vari fattori, in modoche ogni vincolo possa contribuire in modo paragonabile agli altri.