Solución de Problemas con CCP restricción sobre conjuntos finitos

slides basados en el curso “constraint Programming” de Christian Schulte [2]
Profesor: Camilo Rueda [1]

[1]Universidad Javeriana-Cali,
[2]KTH Royal Institute of Technology, Sweden

PUJ 2008

Restricciones de conjuntos finitos

- Las variables toman conjuntos de enteros como valores
  - asignación: asocia variables con conjuntos de enteros
  - store: asocia variables con conjuntos de conjuntos de enteros
- Restricciones típicas
  - operaciones de conjuntos: unión, intersección,...
  - cardinalidad: variable de conjunto y variable de dominio finito

Ejemplo: códigos de Hamming

- Generar códigos de Hamming
  - usar b bits
  - para codificar n símbolos
- tales que
  - la distancia Hamming entre dos códigos de símbolos es al menos d.

Generar n cadenas de bits de longitud b con distancias entre cada par de al menos d

Distancia de Hamming

- La distancia Hamming h es el número de bits diferentes para dos cadenas de bits
- Por ejemplo:
  - $h(10001, 01001) = 2$
  - $h(11010, 11110) = 1$
- Códigos con distancia Hamming grande
  - robustez en transmisiones

Usar conjuntos para cadenas de bits

- La distancia Hamming h es el número de bits diferentes para dos cadenas de bits
- Cadena

  $$b_4 b_3 b_2 b_1$$

  se representa con el conjunto:

  $$\{1 \times b_1, 2 \times b_2, 3 \times b_3, 4 \times b_4\}$$
- Por ejemplo
  - 1001 se representa con $\{1, 4\}$
  - 0111 se representa con $\{1, 2, 3\}$

El problema

- Para cada símbolo, un código c_i con
  $$\emptyset \subseteq c_i \subseteq \{1, ..., b\}$$
- Suponga los códigos c_1 y c_2
  - número de unos comunes
    $$u := card(c_1 \cap c_2)$$
  - número de ceros comunes
    $$z := card((\{1, ..., b\} - c_1) \cap (\{1, ..., b\} - c_2))$$
  - distancia de al menos d
    $$b - u - z \geq d$$

Script: las constantes

```
static final int n = 16;

static final int b = 20;

static final int d = 3;
```

Script: las variables

```
// variables
c = newVarArray < SetVar > (this , n, SetVar.class);
for (int i = 0; i < n; i++){
  dom(this , c.get(i), SRT_SUB, 1, b);
}

// Variables para el complemento
VarArray < SetVar > cc
   = new VarArray < SetVar > (this , n, SetVar.class);
for (int i = 0; i < n; i++){
   rel(this , cc.get(i), SRT_CMPL, c.get(i));
}
```

Script: ciclos y ramificación

```
for (int i = 0; i < n; i++){
  SetVar c1 = c.get(i);
  SetVar cc1 = cc.get(i);
  for (int j = i + 1; j < n; j++){
     SetVar c2 = c.get(j);
     SetVar cc2 = cc.get(j);
...
  }
}

branch(this , c,
          SETBVAR_NONE, SETBVAL_MIN);
```

Script: intersecciones

```
SetVar c_int_c = new SetVar(this);
SetVar cc_int_cc = new SetVar(this);
rel(this , c1, SOT_INTER, c2, SRT_EQ,
    c_int_c);
rel(this , cc1, SOT_INTER, cc2, SRT_EQ,
    cc_int_cc);
```

Script: cardinalidad

```
IntVar u = new IntVar(this, 0, b);
IntVar z = new IntVar(this, 0, b);

cardinality(this, c_int_c, u);
cardinality(this, cc_int_cc, z);

int cs[] = {1,1};
VarArray<IntVar> uz =
        new VarArray<IntVar>(u,z);
linear(this, cs, uz, IRT_LQ, b-d);
```

Ramificación para conjuntos

- Seleccionar variable
  - de acuerdo a la cardinalidad
  - de menor elemento ya en el conjunto
  - ...

  Alternativas
  - incluir o excluir valor

    $n \in x$ o $n \notin x$

Modelo para conjuntos

- La mayor diferencia con respecto a los enteros es:
  - los enteros están totalmente ordenados
  - los conjuntos están solamente parcialmente ordenados
    - menor elemento, glb
    - mayor elemento lub
- los stores en el modelo pueden representar cualquier subconjunto del universo
  - demasiado para connuntos finitos

Ejemplos de store para conjuntos

- Asuma que
  - $glb(s(x)) = \{1\}$
  - $lub(s(x)) = \{1, 2, 3, 4\}$
- Conjuntos entre $glb(s(x))$ y $lub(s(x))$

  $\{1\}$ $\{1, 2\}, \{1, 3\}, \{1, 4\}$
  $\{1, 2, 3\}, \{1, 2, 4\}, \{1, 2, 3, 4\}$
  - demasiados!

Adaptar el modelo

- Universo: U
  - conjunto de valores, finito
- Descripciones de valores: $D \subseteq 2^U$
  - no todos los subconjuntos de U se permiten
- Stores $V \to D$
- Requisitos
  - vacío, universo $\emptyset, U \in D$
  - asignaciones $\{u\} \in D$ para todo $u \in U$

Descripciones de valores para conjuntos

- Universo: $U := 2^{\{-m,\ldots,n\}}$
  - conjunto de valores, finito
- Descripciones de valores: $D \subseteq 2^U$
  - algún conjunto finito de enteros
- Conjuntos convexos:
  el conjunto $Y \subseteq X$ para un orden parcial $(X, \leq)$
  - es convexo ssi para todo $x \in X$
    $$glb(Y) \leq x \leq lub(Y) \Rightarrow x \in y$$
- Descripciones de valores
  $D := \{d \mid d \subseteq U,\ d \text{ convexo}\}$
  - completamente definido mediante *glb* y *lub*!

Conjuntos finitos vs variables 0/1

- Modelar con conjuntos convexos como descripciones de valores
  - $n \in glb(d)$ n está definitivamente en el conjunto
  - $n \notin lub(d)$ n definitivamnte no está en el conjunto
- Para conjuntos que son subconjuntos de $\{1, ..., n\}$
  - corresponde a modelar el conjunto con n variables 0/1 de dominio finito $b_1, ..., b_n$

    $b_i = 1 \Leftrightarrow$ el i-ésimo elemento est en el conjunto
  - en Gecode: también se lleva información de cardinalidad