Solución de Problemas con CCP
Búsqueda

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

Búsqueda

- Ramificar
  - define el árbol de búsqueda
  - qué propiedades tiene?
  - qué es un árbol de búsqueda?
- Exploración
  - cómo explorar el árbol de búsqueda?
  - búsqueda de primera solución
  - búsqueda de mejor solución

Ramificación

- Cómo definir ramificación en el modelo
  - sugiere nuevas restricciones
  - formulación operacional: nuevos propagadores
- Ramificación toma decisiones con base en el store actual y en los propagadores
  - “primera falla”: variable con dominio más pequeño
  - variable más restringida

Propiedades

- árbol construido por ramificación
  - árbol finito decreciente
  - no se pierden soluciones completo
  - no hay soluciones duplicadas no-sobrelape

Notación

- Conjunto de propagadores
  $$\mathcal{P} := S \to S$$
  fnciones de store a store
  - contractores
  - monotónicos
- Conjunto de modelos de restricciones
  $\mathcal{M}$

Ramificación: definición

Función b de propagadores y store
a tuplas de conjuntos finitos de propagadores

$$b(P, s) = (P_1, ..., P_n)$$

donde

$P_i \subseteq \mathcal{P}$ y finito

se llama la i-ésima alternativa

Ramificación: propiedades

- Asumir s no fallido

$$\boxed{\begin{array}{l} R = sol(\langle V, U, P\rangle, s) \\ R_i = sol(\langle V, U, P \cup P_i\rangle, s) \end{array}}$$

- Completo: $R_1 \cup ... \cup R_n = R$
- no-sobrelapamiento: $R_i \cap R_j = \emptyset$ para $i \neq j$
- decreciente:
  si $propague(P \cup P_i, s) = (P', s')$
  entonces $s' < s$

Ramificación: decreciente

- de propiedad decreciente se deduce que $R_i \subset R$
- Sin embargo, $R_i \subset R$ no basta
  - se debe disparar nueva propagación
- si s es punto fijo simultáneo de P, entonces
  $\exists p \in P_i : \quad p(s) < s$

Modelo extendido

Un modelo de restricciones es una cuádrupla $\langle V, U, P, b \rangle$

- V :: variables
- U :: valores
- P :: propagadores
- b :: ramificación

Arbol de búsqueda

Un árbol de búsqueda para un modelo $\langle V, U, P, b \rangle$ consiste de nodos (P', s)

La raiz del árbol es

$propague(P, S_{inic})$

donde

$s_{inic}(x) \subseteq U$ para todo $x \in V$

Arbol de búsqueda

- Para toda hoja (P, s)

$$\begin{cases} \text{o bien} & s \text{ es fallido} \\ \text{o sino} & b(P, S) = () \quad (\text{“resuelto”}) \end{cases}$$

- para todo nodo interno
  - s no es fallido
  - si $b(P, S) = (P_1, ..., P_n)$ entonces el nodo tiene los hijos $propague(P \cup P_1, s), ..., propague(P \cup P_n, s)$

Arbol de búsqueda: propiedades

- invariante: para cada nodo del árbol (P, s):

$$\begin{cases} \text{o bien} & s \text{ es fallido} \\ \text{o sino} & s \text{ es punto fijo simultáneo de } P \end{cases}$$

- Si el nodo (P_1, s_1) está debajo de (P_2, s_2) en el mismo camino:

$$s_1 < s_2$$

- El árbol de búsqueda es finito
  - por decrecimiento de stores

Optimización

- Para propagación en caso de ramificación
  - no empezar suponiendo todo propagador activo
  - solamente suponer activos los agregados por ramificación porque se empieza de un punto fijo simultáneo
- la función *propague* recibe los conjuntos de propagadores activos y no activos

Estrategias de ramificación: notación lambda

- Cualquier función $f \in X \to Y, f(x) = e$, se puede escribir como

$$f = \lambda x \in X.\ e$$

- si es claro qué es X, se escribe:

$$f = \lambda x.\ e$$

ejemplo, $f(x) = x^2$ se escribe $f = \lambda x.\ x^2$

Estrategias de ramificación: “ primera falla”

$b(P, s) =$
  if $\exists x$ tal que
    $card(s(x)) > 1$ *and*
    $card(s(x))$ es mínimo *then*
      $(\{\lambda y.$ *si* $x = y$ *entonces* $\{min(s(x))\}$
        sino $s(y)\}$,
       $\{\lambda y.$ *si* $x = y$ *entonces* $s(x) - \{min(s(x))\}$
        sino $s(y)\}\}$)
  else
    ()

Exploración: primero en profundidad

$$
\begin{array}{l}
dfe(P, s) = \\
\quad (P', s') = propague(P, s) \\
\quad \textit{if } fallido(s') \textit{ then } s' \textit{ else} \\
\quad\quad \textit{case } b(P', s') \textit{ of } () \textit{ then } s' \\
\quad\quad\ [] \ (P_1, P_2) \textit{ then} \\
\quad\quad\quad s_2 := dfe(P' \cup P_1, s'); \\
\quad\quad\quad \textit{if } fallido(s_2) \textit{ then } dfe(P' \cup P_2, s') \\
\quad\quad\quad \textit{else } s_2 \textit{ end} \\
\quad\quad\ \textit{end} \\
\quad \textit{end}
\end{array}
$$

Exploración en Gecode

- modelo son espacios de computación
- los espacios proveen métodos para la búsqueda
  - status devuelve el estado
  - cloneSpace retorna una copia
  - commit se compromete con una alternativa

Restauración del estado

- la búsqueda debe restaurar el estado (“backtracking”)
- estrategias
  - trailing recuerda los cambios
  - recomputation recalcula el estado
  - copying guarda copia de cada estado

Restauración: trailing

- El más usado
- Ventajas
  - registra exactamente lo que pasó
- Desventajas
  - es complicado (toda operación debe recordar cambios)
  - requiere técnicas adicionales (“time stamping”)
  - difícil de aliar con concurrencia

Restauración: recomputation

- Recuerda el camino hacia los nodos
  - el camino empieza en cierta copia *c*
  - recalcula rehacindo ramificación sobre *c* para todos los nodos
  - calcula punto fijo único
  - se conoce como “batch recomputation”
- Ventaja: buen uso de memoria
- Desventaja
  - sobrecosto al recomputar

Restauración: copiar

- Ventajas
  - es simple
  - permite concurrencia
- Desventaja
  - requiere mucha memoria

Restauración en Gecode

- Híbrido entre copiado y recomputación
- recomputación adaptable
  - se guarda una copia de la mitad del camino

Restricciones comunes

- Las más usadas
  - reificadas, aritméticas, “element”, “distinct”, “regular”,...
- cómo se usan para modelar
- qué propagadores para ellas
- qué las hace globales
- fortaleza de propagación

Igualdad lineal

$$\sum_{i=1}^{n} a_i x_i = d$$

donde a_i, d son enteros y $a_i \neq 0$

- cómo propagar a bajo costo información sobre límites?
  - para cada x_i observar qué tan grande o pequeña puede ser
  - consideramos por ahora $ax + by = d$

en Gecode:

```
linear (this , a, x, IRT_EQ, d);
```

Restricción “element”

$a[x] = y$

a es un arreglo de enteros
x, y son variables
el valor de y es el valor de a en la posición x
en particular, $0 \leq x <$ num. de elementos de a

en Gecode:

element (*this* , a, x, y);

Restricción de cardinalidad

- en *distinct*: las variables pueden tener un valor específico a lo sumo una vez
- Generalización: límites inferiores y superiores sobre número de ocurrencias
- Aplicación: asignación de turnos
  - trabajadores: las variables describen su turno
  - patrón: establece requisitos mínimos y máximos de trabajadores por turno

en Gecode:
count (*this* , x, n, m)
impone la restricción:
$card(\{i \in \{0, ..., |x| - 1\} \mid x_i = n\}) = m$

Restricciones de “canal”

- variables x_i y z_i, para $o \leq i < n$
- $canal(x_i, z_i)$ se cumple ssi
$$x_i = j \Leftrightarrow z_j = i \quad (0 \leq i < n)$$
- Aplicaciones: modelos duales (problemas de permutación)
  - la reina i está en la fila $j \Leftrightarrow$ la fila j contiene i
  - poner restricción en los dos conjuntos de variables

Restricción “lex”

- variables x_i y z_i, para $o \leq i < n$
- $lex(x_i, z_i)$ se cumple ssi
  $(x_0, ..., x_{n-1})$ es lexicográficamente más pequeño que $(y_0, ..., y_{n-1})$
- Aplicaciones: para romper simetrías

Restricción “regular”

- variables x_i para $o \leq i < n$
- $regular(x_i, r)$ se cumple ssi
  la cadena $x_0, ..., x_{n-1}$ forma una palabra
  del lenguaje de la expresión regular r
- Aplicaciones: decribir patrones que forman secuencias variables
  - en biología
  - en música

Ejemplo

- dados contenedores de distinto tipo y producos de distinto tipo,
- encontrar un empaquetamiento que minimiza contenedores y satisface ciertas restricciones

Ejemplo

3 tipos de contenedor, *rojo*, *azul*, *verde*
5 tipos de producto, vidrio,plástico,acero,madera,cobre

Restricciones de capacidad:
los azules pueden llevar un solo componente
los rojos a lo sumo 3 componentes y a lo sumo 1 de madera
los verdes a lo sumo 4, y a lo sumo 2 de madera

Restricciones de tipo:
los rojos pueden llevar vidrio, madera y cobre
los azules pueden llevar vidrio, acero y cobre
los verdes pueden llevar plástico, madera y cobre

Restricciones de compatibilidad:
la madera requiere ir con plástico
el vidrio excluye el cobre
el cobre excluye el plástico

La orden: 1 de vidrio, 2 plástico, 1 acero, 3 madera, 2 cobre

Ejemplo: modelo1 (ingenuo)

- Variables

t_i: una por componente. $t_i = j$ dice que el componente i va en el contenedor j
col_j, color del contenedor j
v: componente de vidrio p_1, p_2 componentes de plástico a el de acero m_1, m_2, m_3 los de madera c_1, c_2 los de cobre

Modelo1: restricciones (1)

$color_j = \{3 = rojo, 4 = verde, 1 = azul\}$

Las capacidades máximas se cumplen:

$$\sum_i [t_i = j] \leq color_j$$

$count(this\ ,\ t,\ j,\ IRT\_LQ, col[j])$

Modelo1: restricciones (2)

máximo 1 madera en el rojo y 2 en el verde:
$\sum_i [m_i = j] \leq 4 - color_j]$

nueva variable b
nuevo vector $coef = \{1, 1\}$
nuevo vector $var = \{colorj, b\}$
linear (*this* , *coef*, *var*, *IRT_EQ*, 4)
count(*this* , *m*, *j*, *IRT_LQ*, *b*)

Modelo1: restricciones (3)

no hay acero en los rojos:	$[color_j = 3] \Rightarrow [a \neq j]]$
nueva variable b_1	
rel (*this* , $color_j$, IRT_EQ, 3, b_1)	$b_1 \Leftrightarrow color_j = 3$
nueva variable b_2	
rel (*this* , a, IRT_NQ, j, b_2)	$b_2 \Leftrightarrow a \neq j$
rel (*this* , b_2, IRT_GQ, b_1)	$b_1 \Rightarrow b_2$

Modelo1: restricciones (4)

no hay plástico en los rojos:	$(color_j = 3) \Rightarrow (\sum_i [p_i = j] = 0)$
nueva variable b_1	
rel (*this* , $color_j$, IRT_EQ, 3, b_1)	$b_1 \Leftrightarrow color_j = 3$
nueva variable x	
count(*this* , p, j, IRT_EQ, x)	$\sum_i [p_i = j] = x$
nueva variable b_2	
rel (*this* , x, IRT_EQ, 0, b_2)	$b_2 \Leftrightarrow (x = 0)$
rel (*this* , b_2, IRT_GQ, b_1)	$b_1 \Rightarrow b_2$

(simlar para azules y verdes)

Modelo1: restricciones (3)

- Madera requiere plástico
$$[\sum_i [m_i = j] > 0] \Rightarrow [\sum_i [p_i = j] > 0]$$
- el vidrio excluye el cobre:
$$[v_i = j] \Rightarrow [\sum_i [c_i = j] = 0]$$
- el cobre excluye el plástico:
$$[\sum_i [c_i = j] > 0] \Rightarrow [\sum_i [p_i = j] = 0]$$