Solución de Problemas con CCP restricción de canal

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

Restricción de canal: problema

- Dada una baraja de 52 cartas
  - pintas: espada, trébol, corazón, diamante
  - orden: As,2,3,4,5,6,7,8,9,10,J,Q,K
- Repartir las cartas así:
  - poner el as de espadas en una sola pila (“hueco negro”)
  - otras cartas: 17 pilas con tres cartas cada una
- Movida
  - tomar carta del tope de una de las 17 pilas
  - moverla al hueco negro, siempre y cuando la carta sea adyacente a la del tope del hueco negro

Juego del “hueco negro”

- Adyacente
  - independientemente de pinta
  - la siguiente mayor o menor (continúa en ciclo)
  - ejemplo: $9\clubsuit \leftrightarrow 10\clubsuit$, $9\clubsuit \leftrightarrow 8\clubsuit$, $9\heartsuit \leftrightarrow 10\diamondsuit$, $A\heartsuit \leftrightarrow K\clubsuit$
- Objetivo:
  - mover todas las cartas de las 17 pilas al hueco negro

Modelo: restricciones

- A: las cartas deben estar adyacentes en la pila del hueco negro
  - asociar posición en la pila con carta
  - modelar el hecho de ser adyacente
- B: las cartas deben tomarse en orden de las 17 pilas
  - asociar carta con posición en la pila de hueco negro
  - modelar que las posiciones están oredenadas
  - orden en la pila hueco negro: cartas movidas en orden de las 17 pilas
- C: la primera carta en el hueco negro es el as de espadas.

Cartas adyacentes

- numere las cartas de 0 a 51
  - $\spadesuit : 0 - 12, \clubsuit : 13 - 25, \heartsuit : 26 - 38, \diamondsuit : 39 - 51$
- Adyacencia independiente de pinta: para la carta i, considere $i \mod 13$
- cartas i y j adyacentes
  - $(i \mod 13) - (j \mod 13) = 1 \quad \vee$
  - $(i \mod 13) - (j \mod 13) = 12 \quad \vee$
  - $(j \mod 13) - (i \mod 13) = 1 \quad \vee$
  - $(j \mod 13) - (i \mod 13) = 12$

Modelar adyacencia

- Gecode no tiene restricción para módulo
  - ...usar entonces *element*
- Element: arreglo m de 52 elementos
  - inicialice: $m[i] = i \mod 13$
  - el número de la carta es el índice en el arreglo, su módulo 13 es el valor
- Entonces: $adyacente(i, j)$ es
  $(i \mod 13) - (j \mod 13) \in \{-12, -1, 1, 12\}$
  - crear variable d con dominio $\{-12, -1, 1, 12\}$
  - la diferencia debe ser igual a d

Modelo: restricción A

- Variables: *cartas* es un arreglo de 52 variables
  - valores: $\{0, ..., 51\}$
  - asocia posición en hueco negro con carta
- Restricción A: cartas en cada dos posiciones sucesivas deben ser adyacentes

  $$adyacente(carta[i], carta[i+1]) \quad \text{para} \quad 0 \leq i < 51$$
- Cómo modelar la restricción B?

Modelo: restricción B

- Variables: *pos* es un arreglo de 52 variables
  - valores: $\{0, ..., 51\}$
  - asocia carta con posición en hueco negro
- Restricción B: las posiciones deben estar en orden

  $$pos[i] < pos[i+1] \quad \text{para} \quad 0 \leq i < 51$$
- Cómo modelar la restricción A?

Modelo: restricciones A y B

- Tomar ambos: *pos* y *carta*: son duales por intercambiar valores y variables

  $$carta[i] = j \quad \Leftrightarrow \quad pos[j] = i$$

- Incluir:
  - Restricción A sobre variables *carta*
  - Restricción B sobre variables *pos*
  - Restricción que liga valores de *carta* y *pos*: “channel”

Modelo: ajuste

- Simetrías: los colores son irrelevantes
- Ramificacón: tratar de tomar cartas en el tope de cada pila

Restricción “channel”

- Suponga variables $x_i,\ y_i\ (0 \leq i < n)$
- $channel(x_i, y_i)$ se cumpe si
$$x_i = j \quad \Leftrightarrow \quad y_j = i \quad (0 \leq i < n)$$

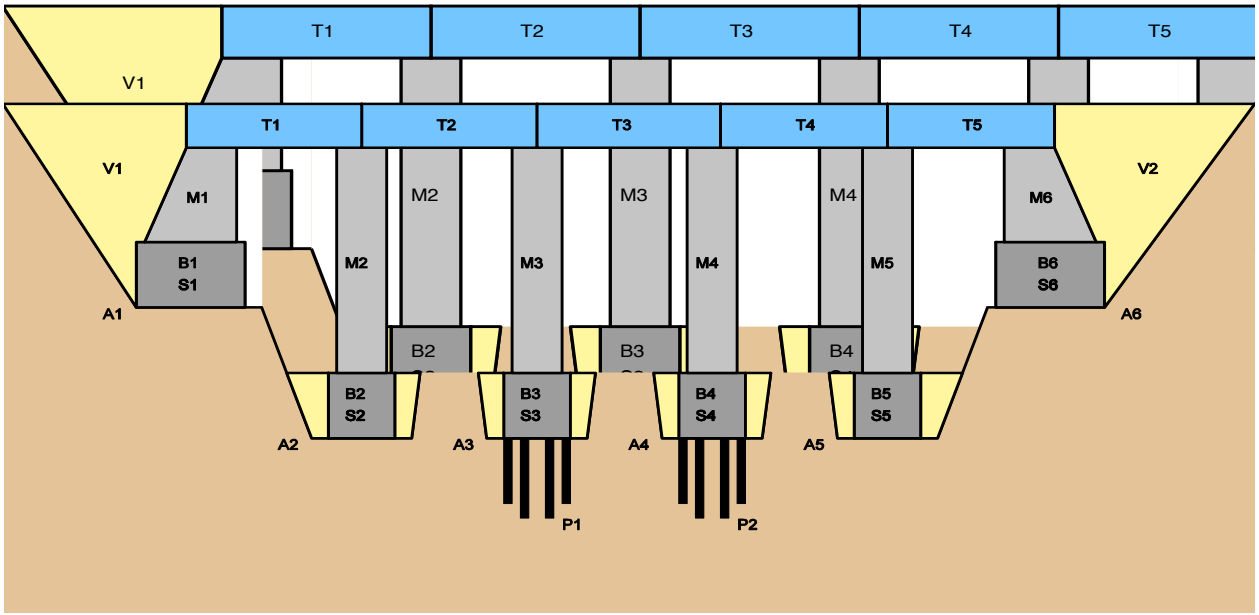
Propagar “channel”

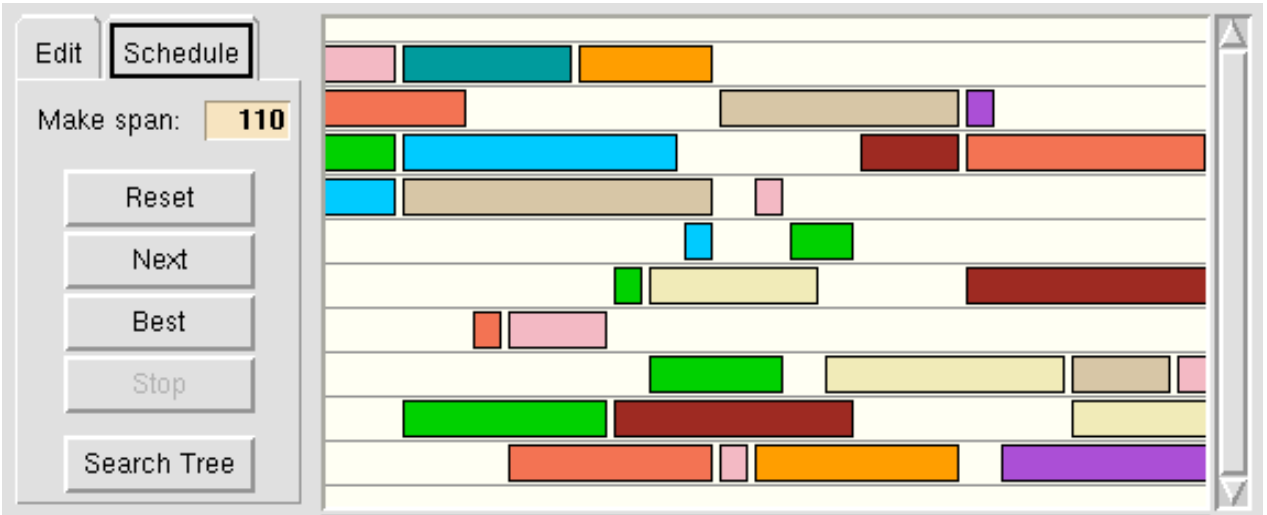
- Método ingenuo: para un store s
  - si $s(x_i) = \{j\}$, entonces $s(y_j)$ debe ser $\{i\}$
  - si $s(y_i) = \{j\}$, entonces $s(x_j)$ debe ser $\{i\}$
- Mejor (método A): para un store s
  - si $j \notin s(x_i)$ entonces elimine i de $s(y_j)$
  - si $j \notin s(y_i)$ entonces elimine i de $s(x_j)$
- Existe algo mejor?

Propagar mejor “channel”

- Observación: para que $channel(x_i, y_i)$ se cumpla, $distinct(x_i)$ y $distinct(y_j)$ deben cumplirse
  - sponga que $x_{i_1} = j$ y que también $x_{i_2} = j$, donde $i_1 \neq i_2$
  - entonces $y_j = i_1$ y también $y_j = i_2$
  - lo que es imposible...
- Solo se necesita un *distinct*
  - propagar *distinct* sobre las x_i
  - la propagación del método A propagará *distinct* para y_i

Prlaneamiento “scheduling”





Solución de un problema de “scheduling”

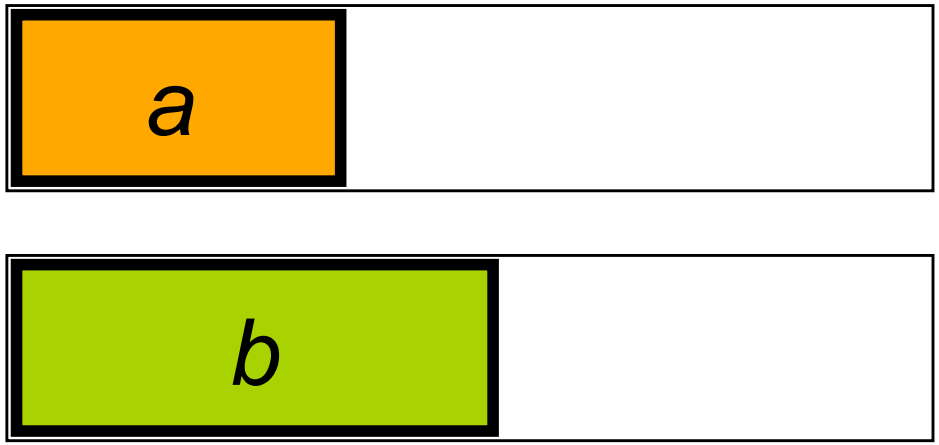
- Tiempo de arranque de cada tarea
- Todas las restricciones satisfechas
- Tiempo de conclusión de tareas mínimo

“scheduling”, modelo

- Variable para tiempo de comienzo de tarea a
  $start(a)$
- restricción de precedencia: a antes que b
  $start(a) + dur(a) \leq start(b)$

Procedimiento de propagación

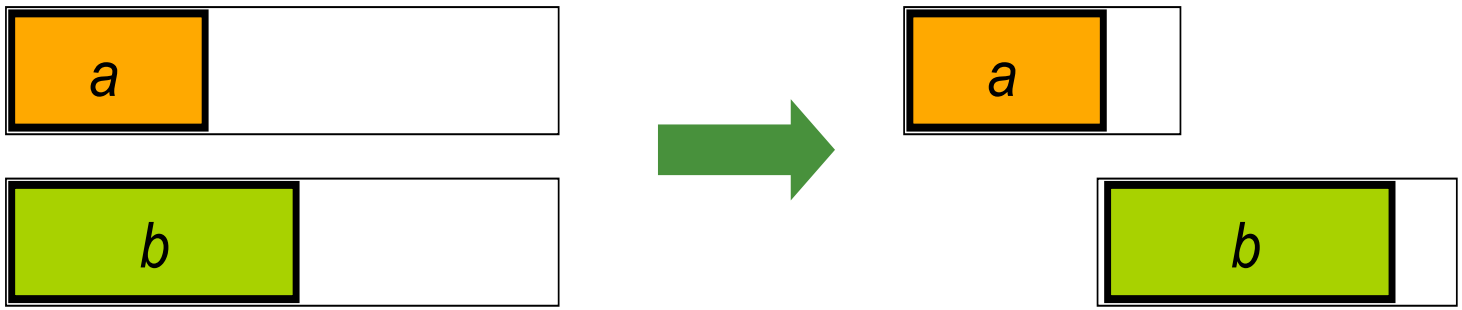
a antes que b



$start(a) \in \{0, ..., 7\}$
$start(b) \in \{0, ..., 5\}$

Procedimiento de propagación

a antes que *b*



$start(a) \in \{0, ..., 7\}$
$start(b) \in \{0, ..., 5\}$

$start(a) \in \{0, 1, 2\}$
$start(b) \in \{3, 4, 5\}$

“scheduling”, modelo

- Variable para tiempo de comienzo de tarea a
  $start(a)$
- restricción de precedencia: a antes que b
  $start(a) + dur(a) \leq start(b)$
- Restricción de recursos
  a antes que b, $start(a) + dur(a) \leq start(b)$
  o
  b antes que a, $start(b) + dur(b) \leq start(a)$

Ramificación

- Establecer orden sobre las tareas
  - qué recurso escoger: el más escaso?
  - qué tarea escoger primero: por ejemplo la más larga

Modelo demasiado ingenuo

- Visión muy local
  - pares individuales de tareas
  - $O(n^2)$ propagadores para n tareas
- isión global
  - todas las tareas sobre un recurso: serializar, ordenar
  - un solo propagador
  - hay buenos algoritmos

Clases de problemas

- Por tipo de recurso
  - a lo sumo una tarea al tiempo: disyuntivo
  - menos que la capacidad: acumulativo
- Por tipos de tareas
  - No interruptibles
  - interruptibles
  - estirables: elástico
- veremos: disyuntivo, no interruptible

Serialización

- Considerar todas las tareas sobre el mismo recurso
- Deducir su orden tanto como sea posible
- menos que la capacidad: acumulativo
- Técnicas
  - Tempotabulación: mirar slots temporales usados/libres
  - búsqueda de arcos: qué tarea ejecuta primero/último?
  - no-primera, no-última: qué tareas no pueden ser primera/última?

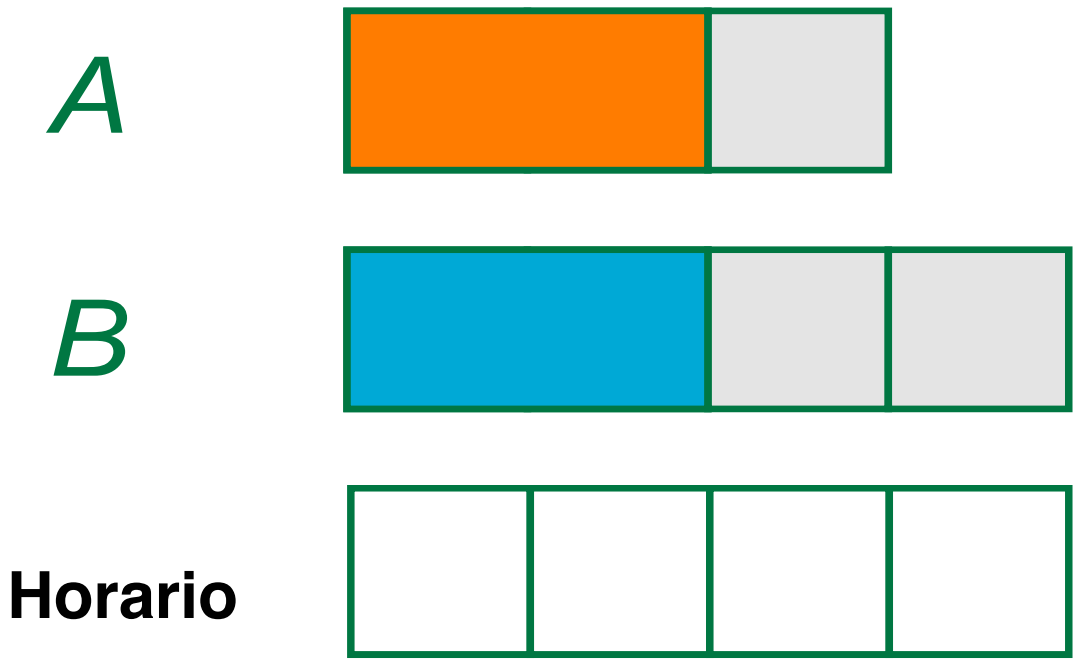
Propagación de tempotabulación

- Horario para recurso
  - Registrar información sobre tiempo en el que una tarea usa con seguridad el recurso
- Propagar en ambas direcciones
  - de tareas a horarios
  - de horarios a tareas

Ejemplo

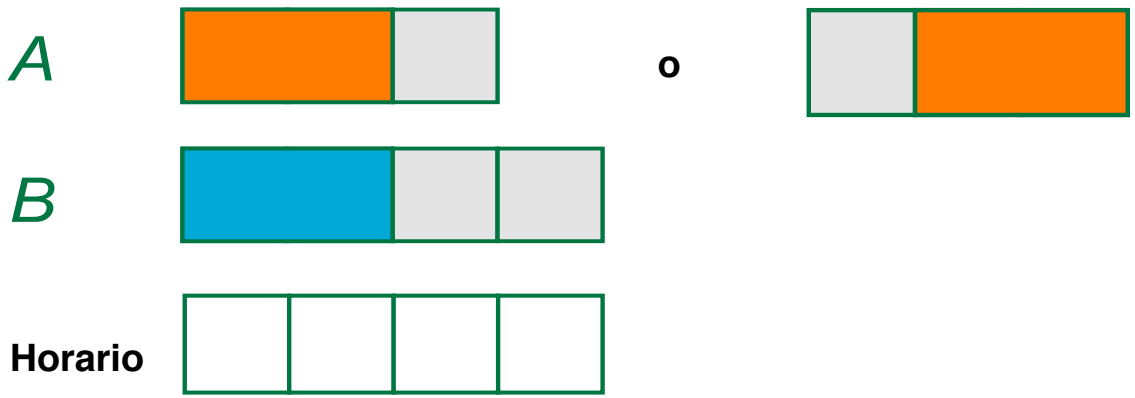
- $start(A) \in \{0, 1\} \quad dur(A) = 2$
- $start(B) \in \{0, 1, 2\} \quad dur(B) = 2$

Hacia horario...



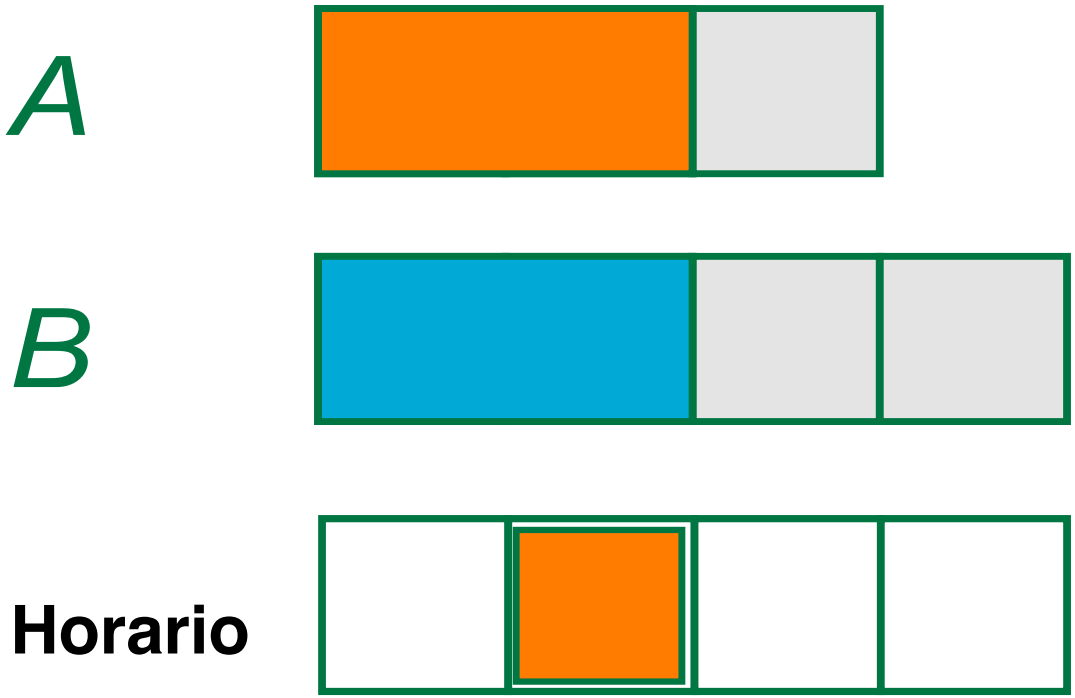
- $start(A) \in \{0, 1\}$ $dur(A) = 2$
- $start(B) \in \{0, 1, 2\}$ $dur(B) = 2$

Hacia horario...



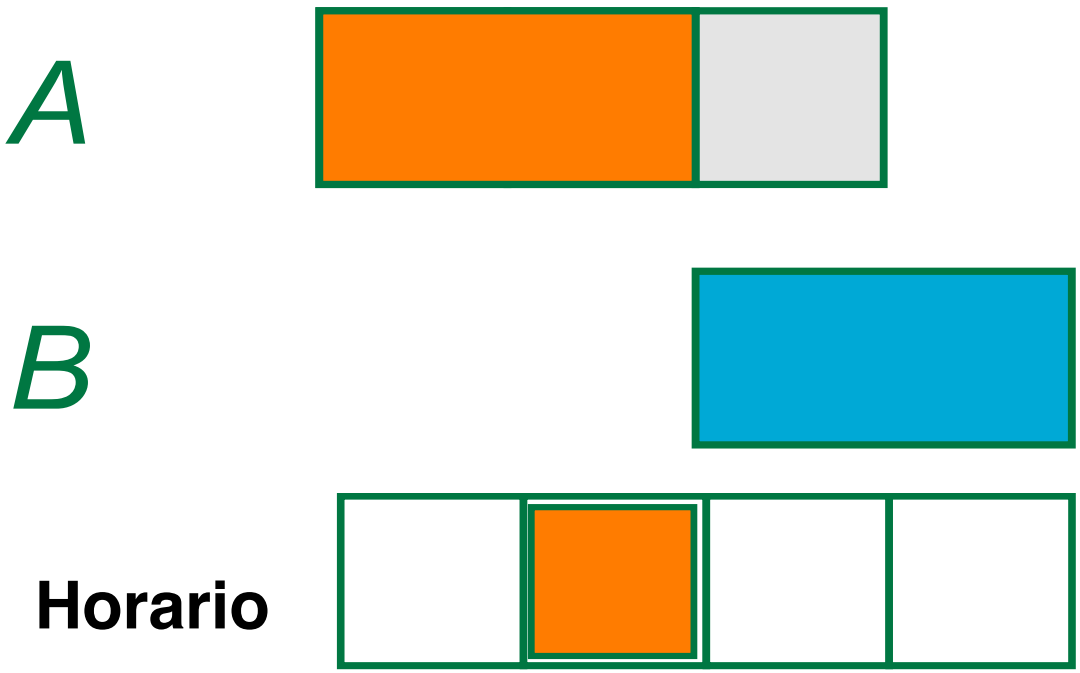
- $start(A) \in \{0, 1\} \quad dur(A) = 2$
- $start(B) \in \{0, 1, 2\} \quad dur(B) = 2$

Hacia horario...



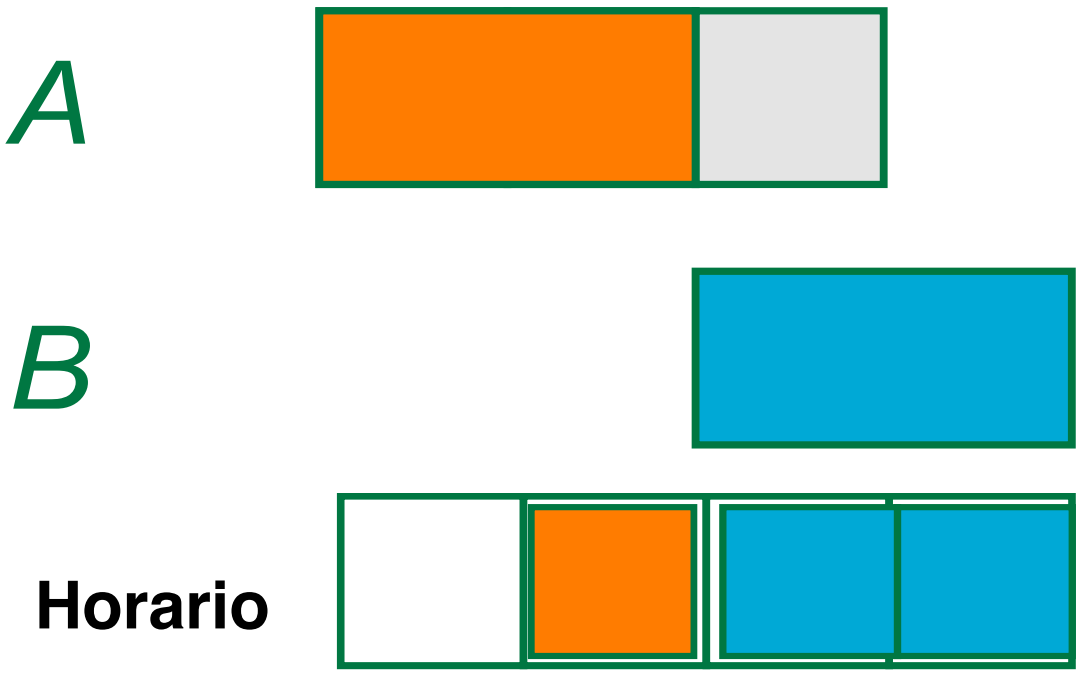
- $start(A) \in \{0, 1\}$ $dur(A) = 2$
- $start(B) \in \{0, 1, 2\}$ $dur(B) = 2$

Hacia tareas...



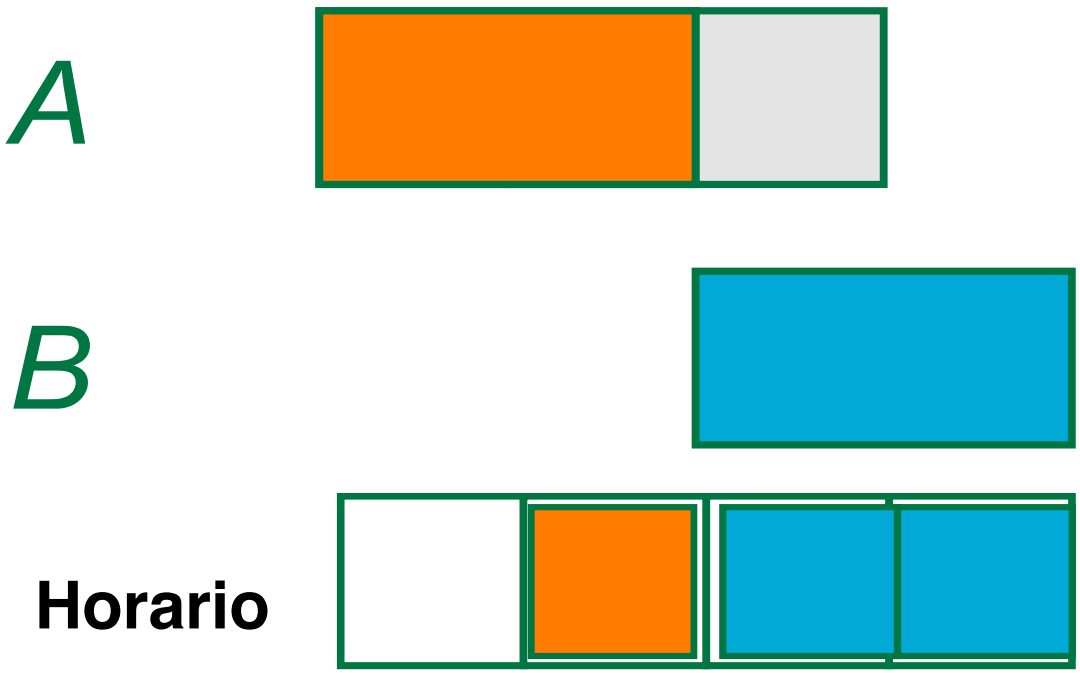
- $start(A) \in \{0, 1\}$ $dur(A) = 2$
- $start(B) \in \{2\}$ $dur(B) = 2$

Hacia horario...



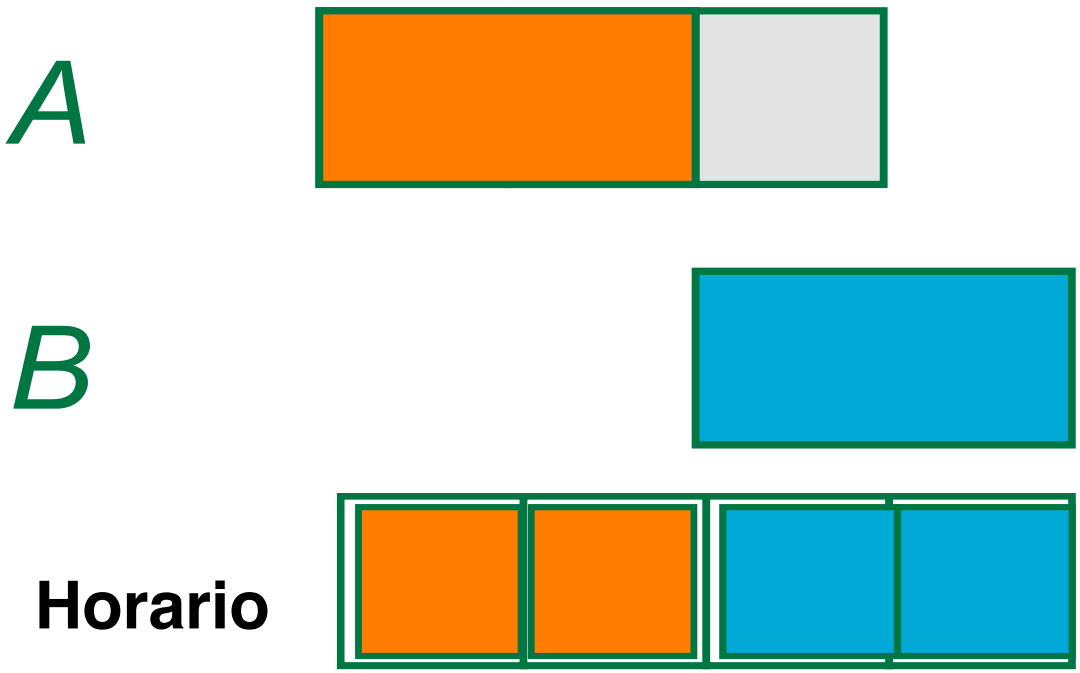
- $start(A) \in \{0, 1\}$ $dur(A) = 2$
- $start(B) \in \{2\}$ $dur(B) = 2$

Hacia tareas...



- $start(A) \in \{0\}$ $dur(A) = 2$
- $start(B) \in \{2\}$ $dur(B) = 2$

Hacia horario...(inútil)



- $start(A) \in \{0, 1\}$ $dur(A) = 2$
- $start(B) \in \{2\}$ $dur(B) = 2$

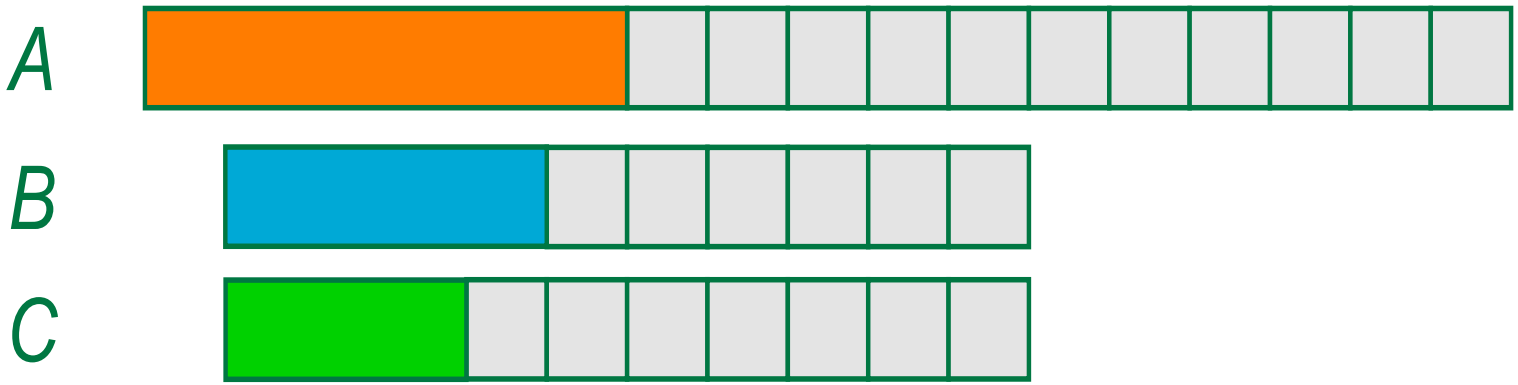
Búsqueda de arcos

- tempotabulación es débil
  - propagación reificada es mejor...
- Pero hay idea importante
  - tener en cuenta cuándo se usa un recurso
  - propagar esto hacia las tareas
- Búsqueda de arcos es un esquema más general para propagar el orden (arcos) entre tareas

Búsqueda de arcos: idea

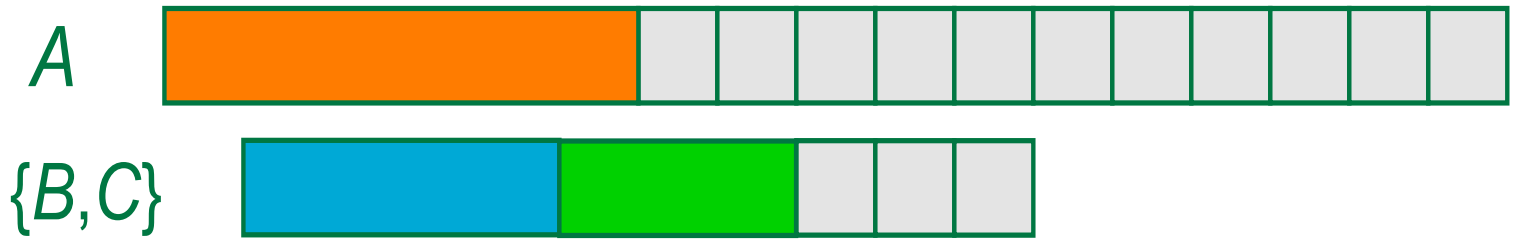
- Asumir un subconjunto O de tareas y $T \in O$
  - restringir T para que ejecute primero
  - averigue si las tareas en $O \backslash \{T\}$
    deben,pueden, o no pueden
    ejecutar de primero o de último
  - de manera simétrica para última
- Puede hacerse en $O(n^2)$ para n tareas

Ejemplo



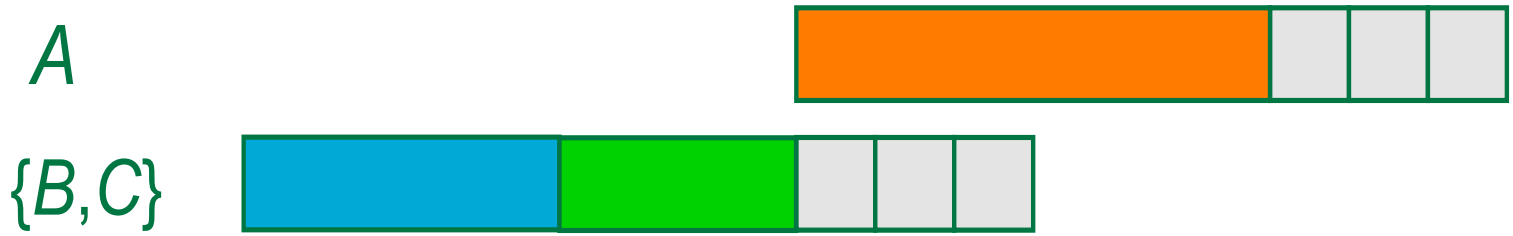
- start(A) $\in \{0,\ldots,11\}$ dur(A) = 6
- start(B) $\in \{1,\ldots,7\}$ dur(B) = 4
- start(C) $\in \{1,\ldots,8\}$ dur(C) = 3

Ejemplo



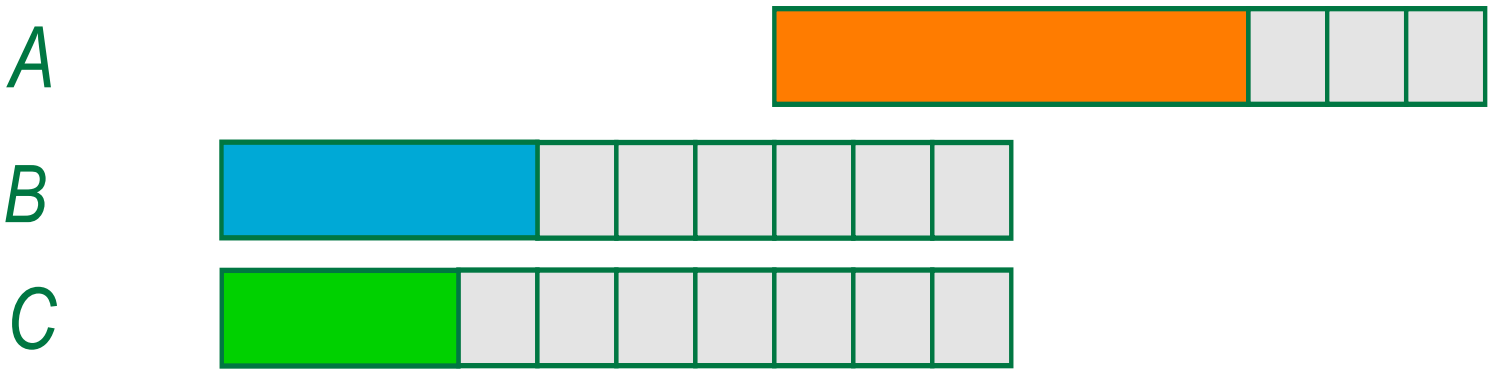
- Considere $\{B, C\}$
  - A no puede ir antes que $\{B, C\}$

Ejemplo



- Considere $\{B, C\}$
  - A no puede ir antes que $\{B, C\}$
  - propague que A va después de $\{B.C\}$

Ejemplo



- Resultado de la propagación
  - $start(A) \in \{8, ..., 11\}$
  - tempotabulación y reificación no propagarían nada!