

# 1. Técnicas de Filtrado y Consistencia

## 1.1. Filtrado

- Elimina elementos que con seguridad no pueden ser parte de la solución, reduciendo el espacio de búsqueda.
- Reduce el problema  $P$  a un problema  $P'$  reducido, simplificado por reducción. Ambos son equivalentes.
- Permite detectar ausencia de solución.

## 1.2. Consistencia

- Grado de compatibilidad entre los valores del dominio y restricciones.
- 1-consistencia o nodo-consistencia: consistencia de nodos. Quitar elementos del dominio que no cumplen con las restricciones unarias.
- 2-consistencia o arco-consistencia: considera las restricciones binarias. Quitar elementos del dominio de una variable que no satisfacen restricciones respecto de otra variable. Algoritmos: AC-1, AC-3.
- consistencia de caminos: dos variables son camino consistentes si para todos los pares de valores de las variables, satisfacen las restricciones con las otras variables del problema
- k-consistencia: Una red es k-consistente, si y solo si, dada cualquier instanciación de k-1 variables, que satisfagan todas las restricciones entre ellas, existe al menos una instanciación de una variable k tal que se satisfacen las restricciones entre las k variables.

## 1.3. Arco-Consistencia

- Eliminar todos los valores que no cumplan con las restricciones.
- Valor viable: posee un valor compatible dentro de los dominios de las variables unidas por una restricción
- Valor no viable: no tiene un valor compatible y será eliminado de dominio de una variable
- Cada valor del dominio debe tener al menos un soporte en el dominio de la otra variable.
- Soporte: Para cada valor en su dominio, para toda otra variable conectada a  $X_i$ , existe un valor  $b$  en  $X_j$  (al menos uno), tal que  $(a, b)$  pertenezcan a  $R_{ij}$ ; es decir, se cumple una restricción y dichos valores son compatibles (por ejemplo,  $X_i \neq X_j$ ). En términos simples, *Soporte* es si se cumple la restricción.
- Un problema es arco-consistente si todas sus variables son arco-consistentes, y una variable es arco-consistente si tiene soporte en todas las variables conectadas.

## 2. Técnicas de Búsqueda completa

Las técnicas de búsqueda completa son técnicas que trabajan construyendo soluciones candidatas de manera incremental y son capaces de descartar estas soluciones parciales tan pronto como determinan que no se producirá una solución válida a partir de ellas. Determinan todas las soluciones al problema o si el problema en realidad no tiene solución. Debido a que revisan “todo” o gran parte del espacio de búsqueda pueden tener un alto tiempo de ejecución en problemas complejos con grandes espacios de búsqueda.

### 2.1. Árboles de Búsqueda

- Estructura que permite construir *instanciaciones completas* en un problema de búsqueda
- El número de niveles del árbol es el número de variables más uno
- A lo más tantos descendientes por nivel como valores tenga el dominio de cada variable
- Dependiendo de la técnica, es más o menos frondoso, las técnicas descartan ramas donde no encontraran soluciones, sin embargo, algunas descartaran más que otras al usar más información del problema.
- Un árbol NO es lo mismo que espacio de búsqueda

### 2.2. Backtracking y Técnicas Look Back

Backtracking va armando un árbol de búsqueda instanciando valores para las diferentes variables. A medida que avanza revisa si se cumplen las restricciones de variables previamente instanciadas. Las técnicas Look Back indican a que punto saltar. Algunos ejemplos:

- Backtracking cronológico: vuelve a la variable asignada anteriormente.
- BT+GBJ: vuelve a la variable conectada en el grafo de restricciones más recientemente instanciada
- BT+CBJ: vuelve a la variable en conflicto más reciente que se encuentre en el conjunto de conflictos.

#### 2.2.1. Backtracking + Conflict-directed Back Jumping

Pasos para usar la técnica:

1. Instanciar y asignar un valor.
2. Si entra en conflicto con una variable ya instanciada, la agrego en el conjunto conflicto de la variable actual y pruebo con otro valor. Si entra en conflicto con más de una, se elije la variable instanciada más prematuramente, la más antigua.

3. Repetir los pasos anteriores
4. Si no hay mas valores a instanciar, entonces del conjunto conflicto se elije la variable que haya sido instanciada más recientemente. Este sera el punto de backtracking.
5. Al llegar a la variable del punto anterior, se le agrega a su conjunto conflicto las variables del conjunto conflicto de la variable desde donde se *viene* (con excepción de la misma variable) y de ahí se borra su conjunto conflicto

## 2.3. Técnicas Look-Ahead

Van armando un árbol de búsqueda instanciando valores para las diferentes variables. A medida que avanzan revisan si se cumplen las restricciones de variables que aún no se encuentran instanciadas. Los valores de las futuras variables que son inconsistentes con la asignación actual son temporalmente eliminados de su dominio. Si el dominio de una variable queda vacío, se deshace la instanciación de la variable actual y se prueba con otra.

- Forward Checking: revisa solo las restricciones de la variable que se esta instanciando.
- Real Full Look-Ahead: se revisan todas las restricciones de las variables involucradas, es decir, se revisan las restricciones que involucren a la variable instanciada, y a su vez las restricciones que involucren las variables involucradas en esas restricciones.

## Tablas a utilizar

Para ordenar el desarrollo en los ejercicios se deben utilizar las siguientes tablas:

### AC-1 y AC-3

Arco	Dominio	Cola

### BT y BT+GBJ

Variable	Instanciación	Punto de retorno	Chequeos

### BT+CBJ

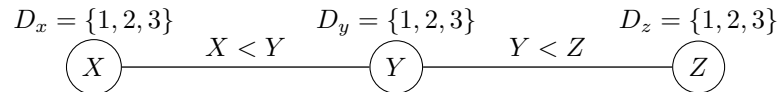
Variable	Instanciación	Punto de retorno	Conjunto de Conflictos	Chequeos

## FC y Real Full Look Ahead

Variable	Instanciación	Dominios Filtrados	Punto de retorno	Chequeos

## Ejercicio 1: Arco Consistencia

Para el siguiente modelo, representado mediante su grafo de restricciones, aplique AC-1 y AC-3.



### SOLUCIÓN

Arco	Dominio	Cola
(X, Y)	$D_x - \{3\}$	$Q = Q' = \{(X, Y)(Y, X)(Y, Z)(Z, Y)\}$
(Y, X)	$D_y - \{1\}$	$Q = \{(Y, X)(Y, Z)(Z, Y)\} \cup Q'$
(Y, Z)	$D_y - \{3\}$	$Q = \{(Y, Z)(Z, Y)\} \cup Q'$
(Z, Y)	$D_z - \{1, 2\}$	$Q = \{(Z, Y)\} \cup Q'$
(X, Y)	$D_x - \{2\}$	$Q = \{(X, Y)(Y, X)(Y, Z)(Z, Y)\}$
(Y, X)		$Q = \{(Y, X)(Y, Z)(Z, Y)\} \cup Q'$
(Y, Z)		$Q = \{(Y, Z)(Z, Y)\} \cup Q'$
(Z, Y)		$Q = \{(Z, Y)\} \cup Q'$
(X, Y)		$Q = \{(X, Y)(Y, X)(Y, Z)(Z, Y)\}$
(Y, X)		$Q = \{(Y, X)(Y, Z)(Z, Y)\}$
(Y, Z)		$Q = \{(Y, Z)(Z, Y)\}$
(Z, Y)		$Q = \{(Z, Y)\}$
(Z, Y)		$Q = \{\emptyset\}$

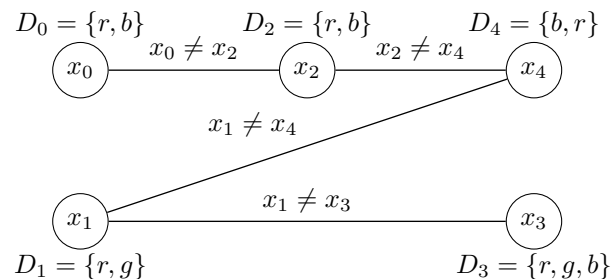
Cuadro 1: Solución con AC-1, con 12 arcos revisados

Arco	Dominio	Cola
(X, Y)	$D_x - \{3\}$	$Q = \{(X, Y)(Y, X)(Y, Z)(Z, Y)\}$
(Y, X)	$D_y - \{1\}$	$Q = \{(Y, X)(Y, Z)(Z, Y)\}$
(Y, Z)	$D_y - \{3\}$	$Q = \{(Y, Z)(Z, Y)(X, Y)\}$
(Z, Y)	$D_z - \{1, 2\}$	$Q = \{(Z, Y)(X, Y)\}$
(X, Y)	$D_x - \{2\}$	$Q = \{(X, Y)\}$
(X, Y)		$Q = \{\emptyset\}$

Cuadro 2: Solución con AC-3, con 5 arcos revisados

## Ejercicio 2, Técnicas de Búsqueda Completa

Para el siguiente CSP, encontrar una solución con BT, BT+GBJ y BT+CBJ, FC y Real Full Look Ahead, construir los árboles de búsqueda y concluir



### SOLUCIÓN

**Respuesta Parcial:** La solución encontrada debería ser  $X_0 = r; X_1 = g; X_2 = b; X_3 = r; X_4 = r$ . BT debería dar una rama más de trashing y más chequeos que BT+GBJ y BT+CBJ para llegar a esta solución. Hasta este punto BT+GBJ y BT+CBJ se comportan igual. A continuación se muestra la tabla de BT+CBJ.

BT realizó 2 saltos mientras que BT+CBJ realizó solo uno. Esto es sin contar aquellos saltos hacia variables con dominios vacíos, en este caso son 4 y 2 respectivamente. Para encontrar la primera solución BT realiza 20 chequeos, mientras que BT+CBJ realiza 15.

**Back Tracking Cronológico**

Variable	Instancia	PT BT	Chequeos
$X_0$	$r$		0
$X_1$	$r$		0
$X_2$	$r$		1
	$b$		1
$X_3$	$r$		1
	$g$		1
$X_4$	$b$		2
	$r$	$X_3$	2
$X_3$	$b$		1
$X_4$	$b$	$X_3$	2
$X_3$		$X_2$	0
$X_2$		$X_1$	0
$X_1$	$g$		0
$X_2$	$r$		1
	$b$		1
$X_3$	$r$		1
$X_4$	$b$		2
	$r$		2
Total	16	4	18

**Back Tracking + Conflict-directed Back Jumping**

Variable	Instancia	PT BT	Conjunto Conflicto	Chequeos
$X_0$	$r$			0
$X_1$	$r$			0
$X_2$	$r$			1
	$b$			1
$X_3$	$r$			1
	$g$			1
$X_4$	$b$			2
	$r$	$x_1$	$Cx_4 = \{X_1\}$	2
$X_1$	$g$			0
$X_2$	$r$			1
	$b$			1
$X_3$	$r$			1
$X_4$	$b$			2
	$r$			2
Total	14	1	1	15

**Forward Checking**

Variable	Instancia	PT BT	Dominio Filtrados	Chequeos
$X_0$	$r$	$X_1$	$D_2 = \{r, b\} - \{r\}$	2
$X_1$	$r$		$D_4 = \{b, r\} - \{r\}$	2
			$D_3 = \{r, g, b\} - \{r\}$	3
$X_2$	$b$		$D_4 = \{b\} - \{b\}$	1
$X_1$	$g$		$D_3 = \{r, g, b\} - \{g\}$	3
$X_2$	$b$		$D_4 = \{b, r\} - \{b\}$	2
$X_3$	$r$			0
$X_4$	$r$			0
Total	7	1	6	13

**Ejercicio 3. Verdadero o Falso**

Responda Verdadero o Falso según corresponda, y justifique:

1. GBJ puede experimentar el fenómeno de trashing.

**Verdadero.** GBJ también puede experimentar trashing. Por ejemplo, si existe problema de consistencia entre una variable  $X_1$  y  $X_3$ , y además  $X_3$  está conectada a  $X_2$ , se volverá a la variable  $X_2$  y se repetirán una o más instanciaciones de  $X_3$ .

2. Espacio de búsqueda y árbol de búsqueda son términos equivalentes.

**Falso.** El espacio de búsqueda depende del modelo formulado, mientras que el árbol de la técnica utilizada. Por otro lado, espacio de búsqueda son las posibles combinaciones de valores de variables mientras que el árbol de búsqueda es una estructura que nos permite la instanciación.

3. La heurística del ordenamiento de variables hace que cualquier algoritmo que resuelve CSP sea más eficiente.

**Falso.** Depende de la técnica a utilizar y de cuán conectado sea el CSP.

4. Si un problema no tiene solución usando FC, puede ser que usando BT se encuentre la solución.

**Falso.** Las técnicas no inventan soluciones. Si el problema efectivamente tiene solución, tanto FC como BT la encontrarán. En lo que se diferencian es en el tiempo y número de chequeos que realizan.

5. La heurística de ordenar respecto de la variable más conectada y con dominio más pequeño ayuda a que GBJ mejore su búsqueda.

**Verdadero.** En general la técnica siempre sirve, porque se dispone de más información al comienzo y se puede profundizar menos en el árbol, permitiendo a GBJ dar saltos más "inteligentes".

6. El árbol de búsqueda solo depende de la técnica que se usa para resolver un CSP y no del modelo asociado.

**Falso.** Dos modelos que definen las variables de manera distinta van a tener árboles muy diferentes.

7. Para backtracking siempre es más fácil resolver un grafo de restricciones débilmente conectado que uno completamente conectado, cuando se desea encontrar todas las soluciones de un problema.

**Falso.** Un problema más conectado podría llevar a que se detecten inconsistencias mucho más rápidamente y que el árbol que se construya sea por tanto menos profundo.

8. Para backtracking siempre será más fácil resolver un grafo de restricciones completamente conectado que uno débilmente conectado, cuando se desea encontrar todas las soluciones de un problema.

**Falso.** Un grafo débilmente conectado significara menos restricciones para las variables conllevando a que se realicen menos chequeos en la búsqueda de las soluciones.

## Ejercicio 4: Materia

Otras preguntas:

- ¿Cuándo GBJ sería más eficiente que CBJ?

CBJ es menos eficiente en el sentido que tiene que ir construyendo y guardando información de conflictos, entonces suponiendo el caso en que el conflicto es causado por la variable, conectada por el grafo, más recientemente instanciada, los dos harían los mismos saltos, pero GBJ no tuvo que construir conjuntos de conflicto.

- ¿Cómo se evalúa la eficiencia de los algoritmos Look-Ahead?

Contando chequeos básicamente, no es sensato considerar el tamaño del árbol porque claramente mientras más chequeos de consistencia hacen en cada paso, más pequeño el árbol, pero más caro fue el proceso. La cantidad de saltos que hacen también es importante, pues después de cada paso es necesario restaurar dominios y rehacer filtros.