

Inteligencia Artificial

Ayudantía 2

Fernanda Weiss
fernanda.weiss.13@sansano.usm.cl

2017-2

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' simplificado por reducción. Ambos son equivalentes.
- Permite detectar ausencia de solución.

1.2 Consistencia

- Grado de compatibilidad entre los valores del domino 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. 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 instancia de k-1 variables, que satisfagan todas las restricciones entre ellas, existe al menos una instancia 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 del 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 permite cumplir la restricción.
- Un problema es arco-consistente si todas sus variables son arco-consistentes, y una variable es arco-consistente si todos sus valores tienen soporte en todos los dominios de las variables conectadas.

1.4 Algoritmos de Arco-Consistencia

Algoritmo AC-1

Algorithm 1 Función revise. Notar que la función es direccional:
 $revise(x_i, x_j) \neq revise(x_j, x_i)$

```

1: function REVISE( $i, j$ )
2:    $delete \leftarrow FALSE$ 
3:   for cada  $a \in D_i$  do
4:     if no hay  $b \in D_j$  tal que  $(a, b) \in R_{ij}$  then
5:        $D_i \leftarrow D_i - a$ 
6:        $delete \leftarrow TRUE$ 
7:     end if
8:   end for return delete
9: end function

```

- Dado un CSP, AC-1 retorna un CSP equivalente y arco-consistente.
- Va revisando cada arco. La función revise(i, j) retorna cierto si ha eliminado algún valor no viable. En caso contrario, el arco de entrada ya es AC y retorna falso.
- Cada vez que haya una eliminación, se tienen que volver a chequear todas las restricciones (se parte de nuevo, hasta llegar al “punto fijo” en el que, pasando por todas las restricciones ya no hay más cambios).

- Se revisan todos los pares y se arrastra el valor de cambio hasta el final (poco eficiente).
- ¿Cómo hacer más eficiente el algoritmo? Sólo basta con revisar las variables que se basen en una variable cuyo dominio ha sido modificado.

Algorithm 2 Algoritmo de Arco Consistencia AC-1

```

1: procedure AC-1( $G$ )
2:    $Q \leftarrow \{(i, j) \mid (i, j) \in \text{arcos}(G), i \neq j\}$ 
3:   repeat
4:      $\text{cambio} \leftarrow \text{FALSE}$ 
5:     for cada  $(i, j) \in Q$  do
6:        $\text{cambio} \leftarrow \text{revise}(i, j) \vee \text{cambio}$ 
7:     end for
8:   until !cambio
9: end procedure

```

Algoritmo AC-3

Algorithm 3 Algoritmo de Arco Consistencia AC-3

```

1: procedure AC-3( $G$ )
2:    $Q \leftarrow \{(i, j) \mid (i, j) \in \text{arcos}(G), i \neq j\}$ 
3:   while  $Q \neq \emptyset$  do
4:     seleccionar y borrar arco  $(k, m)$  de  $Q$ 
5:      $\text{cambio} \leftarrow \text{FALSE}$ 
6:     for  $\text{revise}(k, m)$  do
7:        $Q \leftarrow Q \cup \{(i, k) \mid (i, k) \in \text{arcos}(G), i \neq k, i \neq m\}$ 
8:     end for
9:   end while
10: end procedure

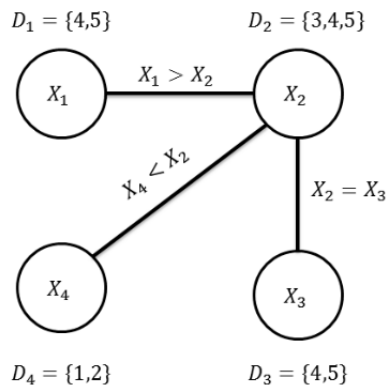
```

- Trabaja con una cola.
- Si borra un valor de un dominio, reintroduce a la cola los arcos asociados con variables que buscan soporte en la variable alterada.
- Con AC-3 se realizan menos revisiones que con AC-1.
- No es caro y es simple de implementar.
- Vuelve a chequear solo los arcos involucrados. Es decir, aquellos que podrían haber dejado de ser AC por la reducción de dominio de X_k .
- Luego de aplicar $\text{revise}(k, m)$, no se requiere agregar (m, k) porque ese arco (m, k) corresponde a la misma relación y no se ve afectado.

1.5 Ejercicio

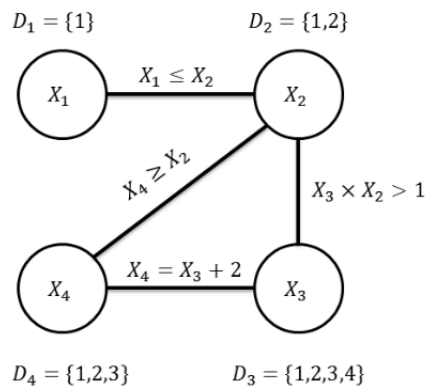
Dado los siguientes grafos de restricciones, aplique AC-1 y AC-3 para obtener dominio reducido:

1.5.1



Respuesta: D 1 = 5; D 2 = 4; D 3 = 4; D 4 = 1, 2

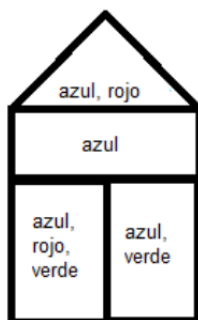
1.5.2



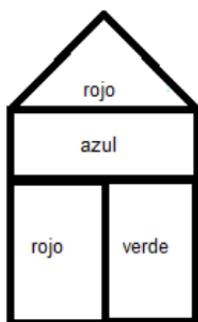
Respuesta: D 1 = 1; D 2 = 2; D 3 = 1; D 4 = 3

1.6 Ejercicio

Dado el siguiente CSP de coloreado de mapas, construya el grafo de restricciones (identificando variables, dominios, restricciones y relaciones) y aplique los algoritmos AC-1 y AC-3 para obtener un problema que sea arco-consistente. Evalúe



que algoritmo es más costoso. Respuesta: AC-1 lleva a cabo 16 revisiones de arcos. AC-3 lleva a cabo 9 revisiones de arcos. Al realizar arco consistencia queda el siguiente resultado:



1.7 Ejercicio

Responda Verdadero o Falso según la aseveración y justifique:

1. Un problema es más difícil si tiene más restricciones.
Falso. Un problema es más difícil cuando se encuentra en la zona de transición, es decir, con un número medio de restricciones. Un problema con muchas restricciones se encuentra más acotado por lo que puede ser más fácil detectar que en realidad no tiene solución.
2. El tamaño del espacio de búsqueda de un problema depende del modelo y no de la técnica a utilizar.
Verdadero. El tamaño de búsqueda lo definen las variables y sus dominios en un modelo.
3. Las técnicas de filtrado son capaces de encontrar una solución a un CSP.
Verdadero. Las técnicas de filtrado aún cuando su objetivo es reducir el espacio de búsqueda asociado a un problema, pueden llegar a encontrar una solución en ciertos casos (por ejemplo, en el caso en que cada variable,

luego del filtro, quede con un tamaño de dominio igual a 1). También pueden detectar que el problema no tiene solución.

4. El uso de técnicas de arco consistencia, puede eliminar soluciones en un CSP.

Falso. Sólo elimina valores que se sabe con seguridad que no serán parte de la solución final.

5. Si un problema es arco consistente, entonces, siempre tiene solución.

Falso. La arco-consistencia de ninguna forma garantiza que existan soluciones, a menos que tenga solo 2 variables.

1.8 Ejercicio

1. Explique por qué AC-3 es mejor que AC-1.

Es más eficiente ya que AC-3 solo vuelve a revisar los arcos que buscan soporte en la variable cuyo dominio ha sido filtrado, mientras que AC-1 vuelve a revisar todos los arcos.

2. Explique por qué se vuelven a revisar todos los arcos/restricciones cuando se realiza AC-1.

En AC-1 cada vez que se borra un valor de un dominio, se deben revisar otras variables que dependieran de ese valor debido a otras restricciones para así realizar la propagación de filtros.

3. Explique por qué se agregan arcos a la cola Q cuando se realiza AC-3 y cuál es el criterio para incorporarlos.

En AC-3 se agregan arcos a la cola cuando se eliminan valores del dominio de alguna de las variables. Estos arcos deben ser revisados nuevamente debido a que es posible que se haya eliminado el soporte de alguno de los valores de las variables ya revisadas restringidas con la variable en cuestión. Al eliminar valores de la variable j se agregan todos los arcos ya revisados de la forma i,j tal que j,i no es el arco que provoca la eliminación actual.

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 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 de búsqueda 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 qué punto saltar de vuelta. Algunos ejemplos:

- Backtracking cronológico: Retorno a la variable recientemente instanciada
- BT + GBJ: Retorno a la variable conectada en el grafo de restricciones más recientemente instanciada.
- BT + CBJ: Retorno a la variable en conflicto más reciente. Pasos para usar la técnica:
 1. Instanciar.
 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.

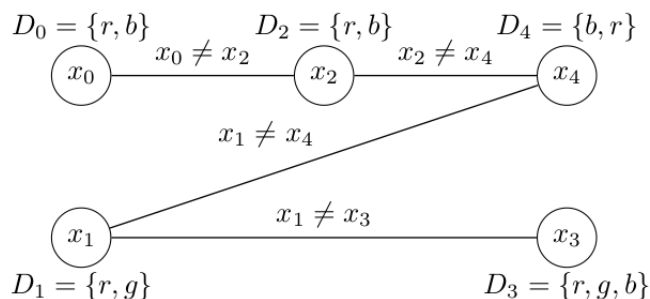
2.3 Técnicas Look-Ahead

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 (arco consistencia).

2.4 Ejercicio

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.



2.5 Ejercicio

Responda Verdadero o Falso según la aseveración 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.

2.6 Ejercicio

1. ¿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.
2. ¿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.