

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 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. 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 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.

Ejemplo

Considerar el problema con variables $X_1 \in \{a, b\}$ y $X_2 \in \{b\}$, $X_1 \neq X_2$

Si asignamos

$$X_1 = a \rightarrow X_2 = b$$

$$X_1 = b \rightarrow \nexists X_2$$

Por tanto la variable X_1 no es arco-consistente, ya que tiene un valor en su dominio que no tiene soporte en el dominio de X_2 . Decimos que $X_1 = a$ es un valor viable y que $X_1 = b$ es un valor no viable y debemos eliminarlo del dominio.

$$D_1 = \{a, b\} - \{b\} = \{a\}$$

Para establecer la arco-consistencia, se propagan las reducciones de los dominios hasta obtener un punto fijo. Es decir, al reducir el dominio de una variable, debo verificar la AC del resto de las variables hasta no obtener más reducciones producto de ese cambio.

1.4. Algoritmos de Arco-Consistencia

Algoritmo AC-1

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

```

function REVISE(i, j)
  delete  $\leftarrow$  FALSO
  for cada  $a \in D_i$  do
    if no hay  $b \in D_j$  tal que  $(a, b) \in R_{ij}$  then
       $D_i \leftarrow D_i - \{a\}$ 
      delete  $\leftarrow$  VERDADERO
    end if
  end for
  return delete
end function

```

- Dado un CSP, AC-1 retorna un CSP equivalente y arco-consistente.
 - Va revisando cada arco. La función $\text{revise}(i, j)$ 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

```

procedure AC-1( $G$ )
   $Q \leftarrow \{(i, j) \mid (i, j) \in \text{arcos}(G), i \neq j\}$   $\triangleright G$  es el grafo de restricciones
 $\triangleright G$  aporta dos arcos por restricción a  $Q$ :  $(i, j)$  y  $(j, i)$ 

  repeat
     $\text{cambio} \leftarrow \text{FALSO}$ 
    for cada  $(i, j) \in Q$  do
       $\text{cambio} \leftarrow \text{revise}(i, j) \vee \text{cambio}$ 
    end for
  until  $\neg \text{cambio}$ 
end procedure

```

Algoritmo AC-3**Algorithm 3** Algoritmo de Arco Consistencia AC-3

```

procedure AC-3
   $Q \leftarrow \{(i, j) \mid (i, j) \in \text{arcos}(G), i \neq j\}$   $\triangleright G$  es el grafo de restricciones
 $\triangleright G$  aporta dos arcos por restricción a  $Q$ :  $(i, j)$  y  $(j, i)$ 

  while  $Q \neq \emptyset$  do
    seleccionar y borrar arco  $(k, m)$  de  $Q$ 
     $\text{cambio} \leftarrow \text{FALSO}$ 
    if  $\text{revise}(k, m)$  then
       $Q \leftarrow Q \cup \{(i, k) \mid (i, k) \in \text{arcos}(G), i \neq k, i \neq m\}$ 
    end if
  end while
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. Verdades

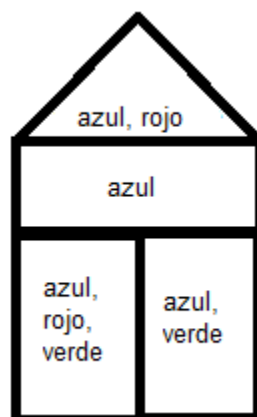
- Si un problema tiene solución, entonces existe una red arco-consistente para dicho problema. Pero si un problema es arco-consistente (o existe un problema equivalente AC) no necesariamente tiene solución.
- Si un problema es AC, solo tiene restricciones binarias y éstas no forman un ciclo, entonces tiene solución.
- Si es k -consistente y tiene k variables, entonces tiene solución.

Algoritmos de AC no permiten decidir si un CSP tiene solución, excepto en los siguientes casos:

- Si algún dominio queda vacío, implica que no hay solución
- Si todos los dominios contienen un único elemento se tiene una solución: la que se obtiene al asignar a cada variable el valor de su dominio.
- Si una variable tiene un dominio con m elementos y el resto de las variables tienen un dominio con un único elemento, entonces hay m soluciones.
- Un CSP con n variables se puede solucionar sin necesidad de realizar búsqueda consiguiendo n -consistencia.

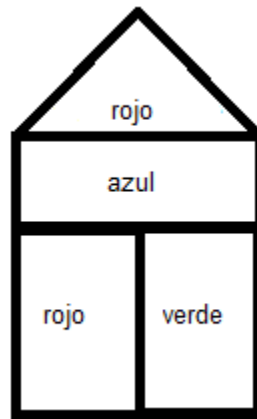
Ejercicio 1

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:



Ejercicio 2

Dados los siguientes problemas, dibuje los grafos y aplique los algoritmos AC-1 y AC-3. Comente sobre el resultado final. Evalúe que algoritmo es más costoso.

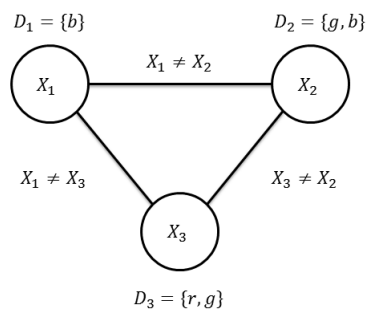
Problema a

Variables: X_1, X_2 y X_3

Dominios: $D_1 = \{b\}$, $D_2 = \{g, b\}$, $D_3 = \{r, g\}$

Restricciones: $X_1 \neq X_2$; $X_1 \neq X_3$; $X_2 \neq X_3$

Respuesta:



Luego de aplicar alguno de los algoritmos quedan los siguientes dominios:

$$D_1 = \{b\}$$

$$D_2 = \{g\}$$

$$D_3 = \{r\}$$

Como cada dominio tiene un solo valor posible, el problema tiene solución $X_1 = b$; $X_2 = g$; $X_3 = r$. El algoritmo AC-1 realiza 12 revisiones, mientras que AC-3 realiza 7 revisiones. Este resultado puede variar según el orden en que se revisan los arcos.

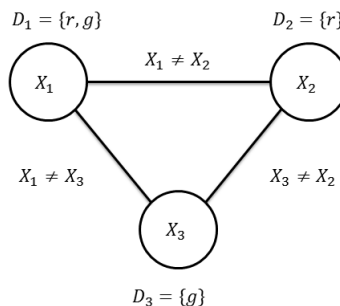
Problema b

Variables: X_1, X_2 y X_3

Dominios: $D_1 = \{r, g\}$, $D_2 = \{r\}$, $D_3 = \{g\}$

Restricciones: $X_1 \neq X_2$; $X_1 \neq X_3$; $X_2 \neq X_3$

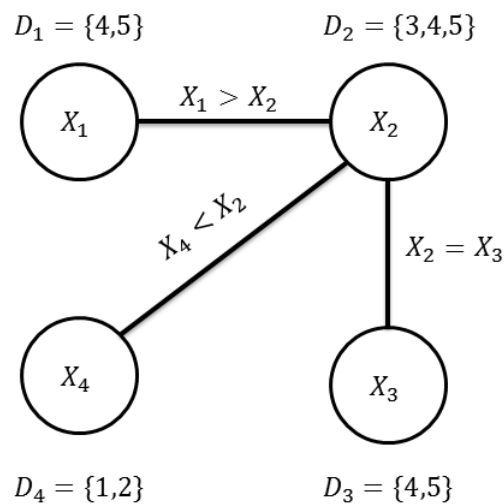
Respuesta:



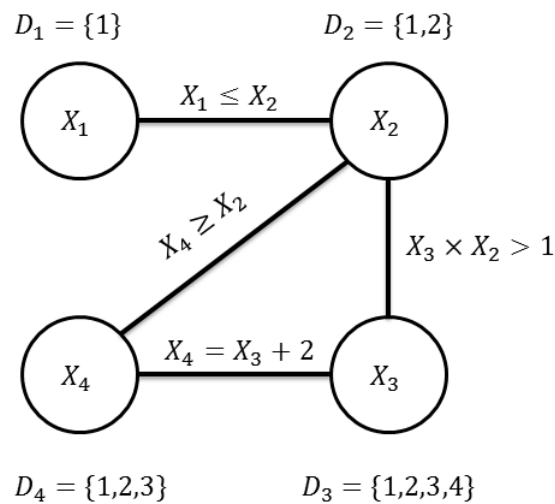
Al aplicar el algoritmo, durante una de las iteraciones uno de los dominios queda vacío con lo que se concluye que el problema no tiene solución. Ambos algoritmos realizan solo 3 revisiones.

Ejercicio 3

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



Respuesta: $D_1 = \{5\}; D_2 = \{4\}; D_3 = \{4\}; D_4 = \{1, 2\}$



Respuesta: $D_1 = \{1\}; D_2 = \{2\}; D_3 = \{1\}; D_4 = \{3\}$

Ejercicio 4

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.

Ejercicio 5

Responda con un breve desarrollo:

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. **El uso de técnicas de arco consistencia, ¿puede eliminar soluciones en un CSP?**

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

3. **Si un problema es arco consistente, entonces, ¿siempre tiene solución?**

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

4. **A medida que aumenta la cantidad de restricciones, ¿Aumenta la dificultad del problema?**

No, los problemas de satisfacción de restricciones poseen una fase de transición que demuestra que los problemas de mayor dificultad se dan con una cantidad media de restricciones. Cuando un problema tiene pocas restricciones es fácil encontrar una solución, mientras que cuando un problema tiene muchas restricciones puede ser fácil determinar que el problema no tiene solución.

5. **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.

6. **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.