

6. PROBLEMAS DE SATISFACCIÓN DE RESTRICCIONES

IA 3.2 - Programación III

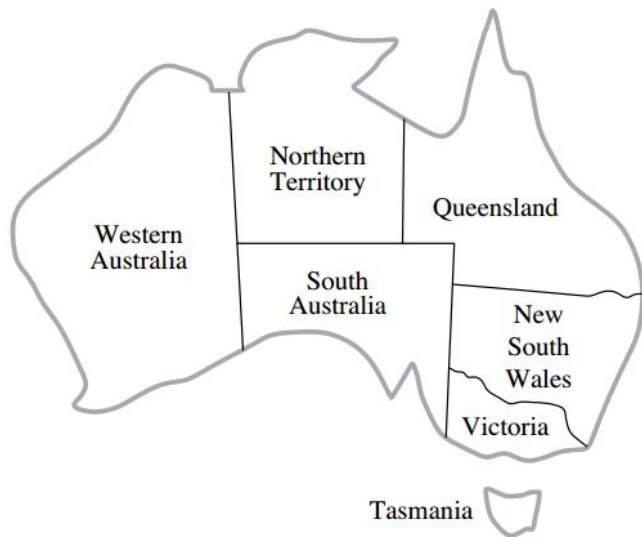
1º C - 2023

Lic. Mauro Lucci

Problema de coloreo de mapas

— — —

Objetivo. Colorear cada región de Australia con rojo, verde o azul de manera que ningún par de regiones limítrofes tenga un mismo color.





Desafío

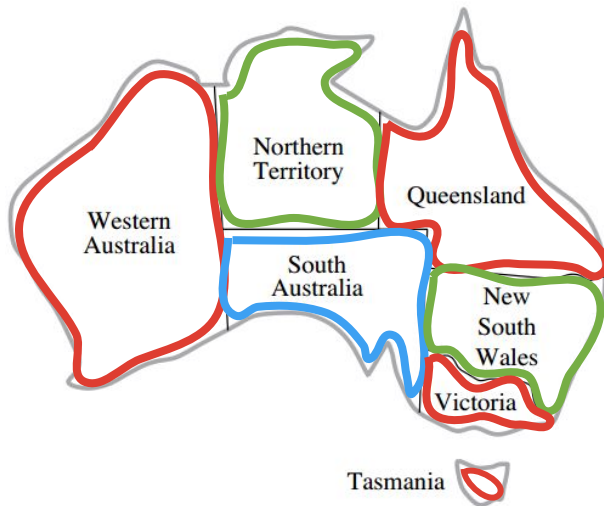
— — —

¿Es posible resolver este problema?



Respuesta

Existen muchas soluciones para este problema. Una de ellas:



Formulación incremental

— — —

- **Estados.** Tupla $s = (s_1, \dots, s_j)$ con $0 \leq j \leq 7$, donde $s_i \in \{\text{red}, \text{green}, \text{blue}\}$ es el color de la i -ésima región. En total hay $1 + 3 + 3^2 + \dots + 3^7 = 3280$ estados.
- **Estado inicial.** $()$.
- **Acciones.**

$i \leftarrow x$: representa pintar la i -ésima región de color x .

$\text{ACCIONES}(s_1, \dots, s_j) = \{j+1 \leftarrow \text{red}, j+1 \leftarrow \text{green}, j+1 \leftarrow \text{blue}\}$.

- **Modelo transicional.**

$\text{RESULTADO}((s_1, \dots, s_j), j+1 \leftarrow x) = (s_1, \dots, s_j, x)$.

- **Test objetivo.**

Para todo $1 \leq i < j \leq 7$, si las regiones i y j son limítrofes, entonces $s_i \neq s_j$.

- **Costo de camino.** El costo individual es 1.

Formulación de estado completo

— — —

- **Estados.** Tupla $s = (s_1, \dots, s_7)$, donde $s_i \in \{\text{red}, \text{green}, \text{blue}\}$ es el color de la i -ésima región. En total hay $3^7 = 2187$ estados.
- **Estado inicial.** Cualquier estado, por ejemplo $(\text{red}, \text{red}, \text{red}, \text{red}, \text{red}, \text{red}, \text{red})$.
- **Acciones.**

$i \leftarrow x$: representar pintar la i -ésima región de color x .

$$\text{ACCIONES}(s) = \{i \leftarrow \text{red}, i \leftarrow \text{green}, i \leftarrow \text{blue} : 1 \leq i \leq 7\}.$$

- **Modelo transicional.**

$$\text{RESULTADO}(s, i \leftarrow x) = (s_1, \dots, s_{i-1}, x, s_{i+1}, \dots, s_7).$$

- **Test objetivo.**

Para todo $1 \leq i < j \leq 7$, si las regiones i y j son limítrofes, entonces $s_i \neq s_j$.

- **Función objetivo.** Número de pares de regiones en conflicto.

Inferencia

— — —

Una vez pintado South Australia con ●, podemos **inferir** que todas las regiones vecinas ya no se pueden pintar con ●.



¿Cómo aprovechamos esta información durante la búsqueda?

Problema de Satisfacción de Restricciones

Constraint Satisfaction Problem (CSP)

Un **Problema de Satisfacción de Restricciones** (CSP) se compone de:

1. Un conjunto de **variables**

$$X = \{X_1, \dots, X_n\}.$$

2. Un conjunto de **dominios**

$$D = \{D_1, \dots, D_n\}.$$

Cada dominio $D_i = \{v_1, \dots, v_k\}$ es el conjunto de valores permitidos para la variable X_i .


3. Un conjunto de **restricciones** C que especifica las combinaciones de valores permitidas.

— — —

Restricciones

Cada restricción C_i de C es un par $\langle Y, R \rangle$ donde:

- Y es una tupla de **variables participantes** en la restricción, y
- R es una **relación** que define los valores que esas variables pueden tomar.

 **Ejemplo.** Si las variables X_1 y X_2 tienen dominio $D_1 = D_2 = \{0,1\}$, entonces una restricción que diga que X_1 y X_2 deben tener valores diferentes se escribe:

$\langle (X_1, X_2), X_1 \neq X_2 \rangle$ o bien $\langle (X_1, X_2), \{(0,1), (1,0)\} \rangle$



Ejemplo – Coloreo de Australia

Variables.

$\{W, N, Q, NS, V, S, T\}$.

Dominios.

$D_i = \{\text{red}, \text{green}, \text{blue}\}$ para cada variable.

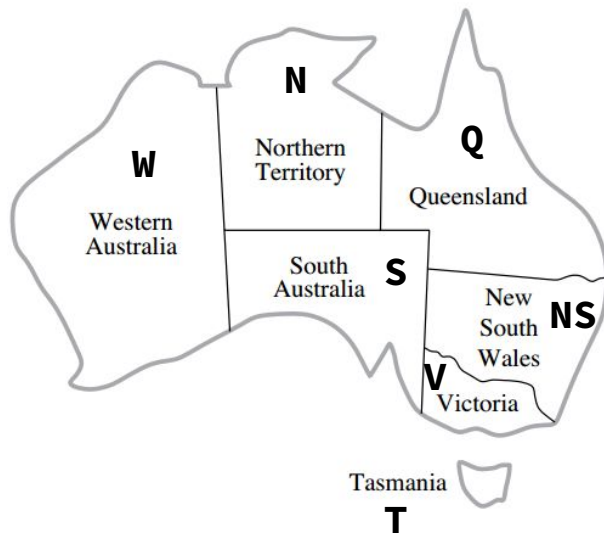
Restricciones.

$C = \{S \neq W, S \neq N, S \neq Q, S \neq NS, S \neq V, W \neq N, N \neq Q, Q \neq NS, NS \neq V\}$

donde una restricción $S \neq W$ es una abreviatura para

$\langle (S, W), S \neq W \rangle$ o equivalentemente

$\langle (S, W), \{(\text{red}, \text{green}), (\text{red}, \text{blue}), (\text{green}, \text{red}), (\text{green}, \text{blue}), (\text{blue}, \text{red}), (\text{blue}, \text{green})\} \rangle$



Más definiciones...

— — —

En un CSP,

- Cada **estado** se define por una **asignación** de valores a algunas o todas las variables

$$\{X_i = v_i, X_j = v_j, \dots\}.$$

- Una asignación es **completa** si asigna un valor a todas las variables, de lo contrario es **parcial**.
- La asignación es **consistente** (o legal) si no viola ninguna de las restricciones.



Ejemplo - Coloreo de Australia. La asignación parcial $\{S=\text{red}, N=\text{red}\}$ no es consistente (es inconsistente) porque viola la restricción $S \neq N$.

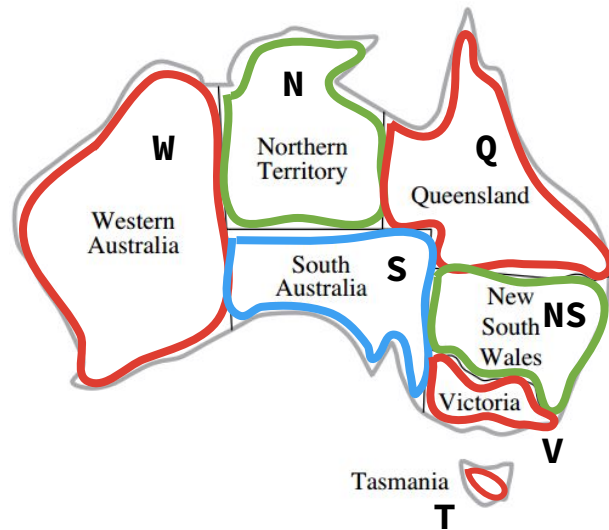
- Una **solución** es una asignación completa y consistente.



Ejemplo – Coloreo de Australia

Existen muchas soluciones para este problema. Una de ellas:

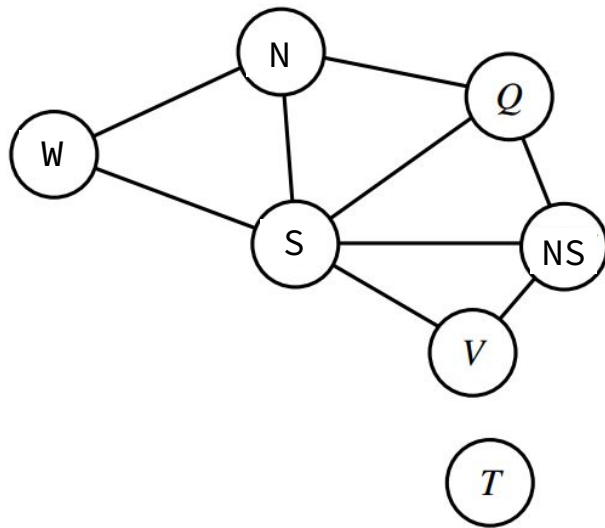
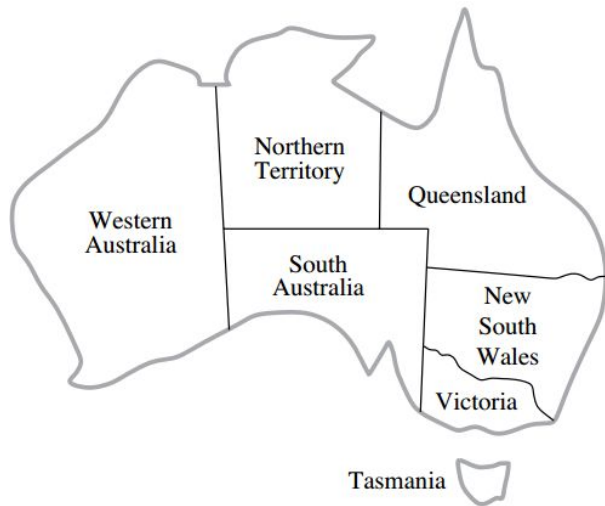
$\{W=\text{red}, N=\text{green}, Q=\text{red}, NS=\text{green},$
 $V=\text{red}, S=\text{blue}, T=\text{red}\}$



Grafo de restricciones

— — —

Los CSP se suelen visualizar como **grafos de restricciones** – donde los nodos son variables y los arcos conectan todo par de variables que participan en una restricción.





Ejemplo – Scheduling

— — —



Una parte del proceso de ensamblaje de un auto requiere de la siguiente secuencia de tareas:

1. Instalar tren delantero y trasero (duración: 10 minutos por tren).
2. Colocar ruedas (duración: 1 minuto por rueda).
3. Ajustar las tuercas de cada rueda (duración: 2 minutos por rueda).
4. Colocar la taza de cada rueda (duración: 1 minuto por rueda).
5. Inspeccionar el ensamblaje final (duración: 3 minutos).

Objetivo. Programar la hora de inicio de cada tarea, respetando el orden de las mismas.



Ejemplo – Scheduling

— — —

Variables. Una por tarea:

$X = \{\text{TREN}\uparrow, \text{TREN}\downarrow, \text{RUEDA}\kappa, \text{RUEDA}\pi, \text{RUEDA}\psi, \text{RUEDA}\nu, \text{TUERCA}\kappa, \text{TUERCA}\pi, \text{TUERCA}\psi, \text{TUERCA}\nu, \text{TAZA}\kappa, \text{TAZA}\pi, \text{TAZA}\psi, \text{TAZA}\nu, \text{INSPEC}\}.$

Dominios. El valor de cada variable indica el tiempo en el que comienza dicha tarea, expresado como un número natural de minutos.

Suponiendo que el ensamblaje debe estar listo en 30 minutos, el dominio de todas las variables es:

$$D_i = \{0, 1, \dots, 27\}$$



Ejemplo – Scheduling

— — —

Restricciones de **precedencia**:

$$\text{TREN}\uparrow + 10 \leq \text{RUEDA}\kappa$$

$$\text{TREN}\downarrow + 10 \leq \text{RUEDA}\kappa$$

$$\text{RUEDA}\kappa + 1 \leq \text{TUERCA}\kappa$$

$$\text{RUEDA}\nearrow + 1 \leq \text{TUERCA}\nearrow$$

$$\text{RUEDA}\searrow + 1 \leq \text{TUERCA}\searrow$$

$$\text{RUEDA}\swarrow + 1 \leq \text{TUERCA}\swarrow$$

$$X + d_x \leq \text{INSPEC}$$

$$\text{TREN}\uparrow + 10 \leq \text{RUEDA}\nearrow$$

$$\text{TREN}\downarrow + 10 \leq \text{RUEDA}\searrow$$

$$\text{TUERCA}\kappa + 2 \leq \text{TAZA}\kappa$$

$$\text{TUERCA}\nearrow + 2 \leq \text{TAZA}\nearrow$$

$$\text{TUERCA}\searrow + 2 \leq \text{TAZA}\searrow$$

$$\text{TUERCA}\swarrow + 2 \leq \text{TAZA}\swarrow$$

para toda variable X distinta de
INSPEC con duración d_x



Ejemplo – Scheduling

— — —

Restricciones de **precedencia**:

$$\text{TREN}\uparrow + 10 \leq \text{RUEDA}\kappa$$

$$\text{TREN}\downarrow + 10 \leq \text{RUEDA}\kappa$$

$$\text{RUEDA}\kappa + 1 \leq \text{TUERCA}\kappa$$

$$\text{RUEDA}\nearrow + 1 \leq \text{TUERCA}\nearrow$$

$$\text{RUEDA}\searrow + 1 \leq \text{TUERCA}\searrow$$

$$\text{RUEDA}\swarrow + 1 \leq \text{TUERCA}\swarrow$$

$$X + d_x \leq \text{INSPEC}$$

Nuevamente,

$\text{TREN}\uparrow + 10 \leq \text{RUEDA}\kappa$
es una abreviatura para
 $\langle (\text{TREN}\uparrow, \text{RUEDA}\kappa),$
 $\text{TREN}\uparrow + 10 \leq \text{RUEDA}\kappa \rangle$

$$\text{TREN}\uparrow + 10 \leq \text{RUEDA}\nearrow$$

$$\text{TREN}\downarrow + 10 \leq \text{RUEDA}\searrow$$

$$\text{TUERCA}\kappa + 2 \leq \text{TAZA}\kappa$$

$$\text{TUERCA}\nearrow + 2 \leq \text{TAZA}\nearrow$$

$$\text{TUERCA}\searrow + 2 \leq \text{TAZA}\searrow$$

$$\text{TUERCA}\swarrow + 2 \leq \text{TAZA}\swarrow$$

para toda variable X distinta de
INSPEC con duración d_x



Ejemplo – Scheduling

Restricciones **disyuntivas**:

Suponer ahora que los operarios necesitan compartir una máquina para instalar los ejes, entonces $TREN_{\uparrow}$ y $TREN_{\downarrow}$ ya no se deben superponer:

$$(TREN_{\uparrow} + 10 \leq TREN_{\downarrow}) \textbf{ or } (TREN_{\downarrow} + 10 \leq TREN_{\uparrow}).$$

Estas restricciones pueden parecer más complicadas, pero en esencia son lo mismo: reducen a pares de valores que pueden asumir las variables.

Por ejemplo, la asignación parcial $\{TREN_{\uparrow}: 3, TREN_{\downarrow}: 13\}$ la verifica, pero $\{TREN_{\uparrow}: 3, TREN_{\downarrow}: 5\}$ no.

Tipos de variables

— — —

El dominio de una variable puede ser:

- **Discreto finito.**

$$D_i = \{v_1, \dots, v_n\}$$

- **Discreto infinito.**

$$D_i = \text{Naturales}$$

- **Continuo.**

$$D_i = [a, b] \text{ con } a, b \in \text{Reales}$$

Nos limitaremos a variables discretas finitas.

Tipos de restricciones

— — —

Las restricciones pueden ser:

- **Unarias**. Restringen el valor de una única variable. Por ejemplo:

$$\langle (S), S \neq \bullet \rangle$$

- **Binarias**. Relacionan dos variables. Por ejemplo:

$$\langle (S, N), S \neq N \rangle$$

- **Globales**. Relacionan un número arbitrario de variables. Por ejemplo:

$$\langle (W, N, S), Alldiff(W, N, S) \rangle$$

donde $Alldiff(W, N, S)$ dice que las variables W, N, S deben tener valores diferentes.

Tipos de restricciones

— — —

- En general, es posible transformar restricciones globales en un conjunto de restricciones binarias.



Ejemplo. La restricción

$$\langle (W, N, S), \text{Alldiff}(W, N, S) \rangle$$

es equivalente a

$$\langle (W, N), W \neq N \rangle, \langle (W, S), W \neq S \rangle, \langle (N, S), N \neq S \rangle$$

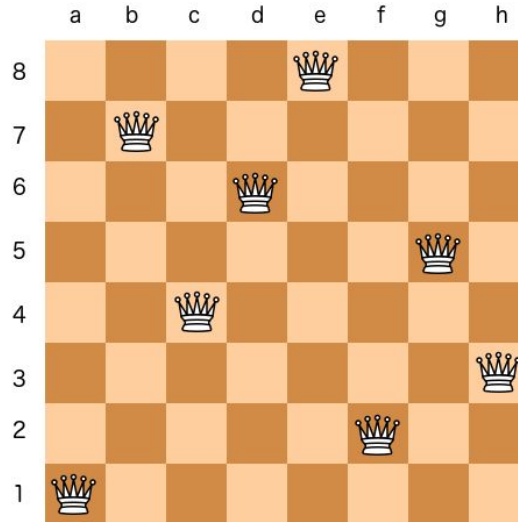
- Usualmente, se prefieren restricciones globales porque:
 - a. Son más fáciles de escribir.
 - b. Permiten mayor poder de inferencia (ya lo veremos).



Ejercicio

— — —


Formular como CSP el problema de las 8-reinas.





Respuesta

— — —

- **Variables.** $\{F_1, \dots, F_8\}$ donde F_i es el número de fila donde se ubica la reina de la columna i .
- **Dominios.** $D_i = \{1, \dots, 8\}$ para toda variable F_i .
- **Restricciones.**
 - No hay dos reinas en una misma columna.  (gratis)
 - No hay dos reinas en una misma fila:
 $\langle (F_i, F_j), F_i \neq F_j \rangle$ para todo $1 \leq i < j \leq 8$.
O equivalentemente: $\langle (F_1, \dots, F_8), \text{Alldiff}(F_1, \dots, F_8) \rangle$
 - No hay dos reinas en una misma diagonal (más difícil):
 $\langle (F_i, F_j), j - i \neq |F_i - F_j| \rangle$ para todo $1 \leq i < j \leq 8$.

Ventajas

— — —

- CSP es una representación natural para una amplia gama de problemas.
- Existen solvers para CSPs.
- Suele ser más sencillo resolver un problema con un solver de CSP que diseñar una solución personalizada usando otra técnica de búsqueda.
- Los solvers de CSP pueden ser más rápidos que otros algoritmos de búsqueda debido a que **pueden eliminar rápidamente grandes porciones del espacio de búsqueda**.

Ejemplo - Coloreo de mapas.

Una vez elegida la asignación $\{S=\text{blue}\}$, ninguna de las 5 regiones vecinas puede usar ese color. Pasamos de $3^5 = 243$ asignaciones posibles para las variables vecinas a $2^5 = 32$, una reducción del 87%.

Ventajas

— — —

- CSP puede desechar asignaciones parciales inconsistentes, sin perder tiempo en completarlas
- CSP puede detectar cuáles variables violan una restricción y focalizarse en ellas.
- Muchos problemas intratables con otros algoritmos de búsqueda pueden ser resueltos rápidamente con CSP.

Inferencia en CSPs

Un algoritmo que resuelve CSP puede:

1. **Buscar**. Asignar a una variable algún valor de su dominio.
2. **Inferir** por **propagación de restricciones**. Usar las restricciones para reducir el dominio de una variable, que a su vez puede permitir reducir el dominio de otra variable, y así.

La inferencia puede intercalarse con la búsqueda o como preprocesamiento. A veces este preprocesamiento ya resuelve el problema.

— — —

Consistencia local


- Recordar que una asignación (de valores a algunas o todas las variables) es consistente si no viola ninguna restricción.
- **Consistencia local**. Una asignación es consistente con una restricción R si no la viola.
- La propagación de restricciones funciona con la idea de consistencia local en el grafo de restricciones.

Consistencia de nodos

— — —

Una variable (nodo) es **nodo-consistente** si los valores de su dominio son consistentes con las restricciones unarias sobre esa variable.

📖 Ejemplo - Coloreo de Australia (*).

Considerar una variante del problema donde South Australia no puede pintarse de , es decir, se agrega la restricción unaria:

$\langle (S), S \neq \text{green} \rangle$ o equivalentemente $\langle (S), \{\text{green}\} \rangle$.

La variable S empieza con dominio $\{\text{red}, \text{green}, \text{blue}\}$ y para hacerla nodo-consistente su dominio se debe reducir a $\{\text{red}, \text{blue}\}$.

Consistencia de arcos

Una variable X_i es **arco-consistente** con otra variable X_j si para todo valor de D_i existe algún valor de D_j consistente con la restricción sobre el arco (X_i, X_j) .



Ejemplos

— — —

Ejemplo - Restricción no lineal.

Considerar un problema con dos variables X e Y con dominio $\{0,1,2,3,4,5,6,7,8,9\}$ y la restricción binaria:

$\langle (X,Y), Y = X^2 \rangle$ o equivalentemente $\langle (X,Y), \{(0,0), (1,1), (2,4), (3,9)\} \rangle$

¿Es X arco-consistente con Y ? Por ejemplo, el par $(0,y)$ pertenece a la relación tomando $y = 0$. Por el contrario, el par $(4,y)$ no pertenece a la relación para ningún valor y del dominio de Y .

Para que X sea arco-consistente con Y , su dominio se debe reducir a $\{0,1,2,3\}$.

Para que Y sea arco-consistente con X , su dominio se debe reducir a $\{0,1,4,9\}$.


Ejemplos

📖 Ejemplo - Coloreo de Australia (**).

Considerar una variante donde South Australia debe pintarse de , es decir, el dominio de S es ahora $\{\text{green}\}$. El dominio de las demás variables sigue siendo $\{\text{red}, \text{green}, \text{blue}\}$.

Las variables S y W están relacionadas con la restricción binaria:

$\langle (W, S), W \neq S \rangle$ o equivalentemente $\langle (W, S), \{(\text{red}, \text{green}), (\text{blue}, \text{green})\} \rangle$

Para que W sea arco-consistente con S, su dominio $\{\text{red}, \text{green}, \text{blue}\}$ debe reducirse a $\{\text{red}, \text{blue}\}$. Es decir, se saca  de su dominio porque el par (green, s) no pertenece a la relación para ningún valor s del dominio de S.

Algoritmo AC-3 para arco consistencia

— — —

```
# Después de aplicar AC-WITH, la variable  $X_i$  es arco consistente con  $X_j$ .  
1 function AC-WITH( $csp, X_i, X_j$ ) return verdadero si se reduce el dominio de  $X_i$ , sino falso  
2     reducido  $\leftarrow$  falso  
3     for each  $x$  in  $csp.D_i$  do  
4         if ( $x, y$ ) no verifica la restricción entre  $X_i$  y  $X_j$   
           para ningún valor  $y$  en  $csp.D_j$  then  
5              $csp.D_i.remove(x)$   
6             reducido  $\leftarrow$  verdadero  
7     return reducido
```


Algoritmo AC-3 para arco consistencia

— — —

Después de aplicar AC-3, o bien todas las variables son arco-consistente o bien alguna variable tiene dominio vacío, indicando que el CSP no tiene solución.

```
1 function AC-3(csp) return fallo si se encuentra inconsistencia, sino verdadero
2   cola ← cola con todos los arcos de csp
3   while cola no es vacía do
4     (Xi, Xj) ← cola.desencolar()
5     if AC-WITH(csp, Xi, Xj) then
6       if csp.Di es vacío then return fallo
7       for each Xk in csp.vecinos(Xi) do
8         cola.encolar(Xk, Xi)
9   return verdadero
```

Algoritmo AC-3 para arco consistencia

— — —

Después de aplicar AC-3, o bien todas las variables son arco-consistente o bien alguna variable tiene dominio vacío, indicando que el CSP no tiene solución.

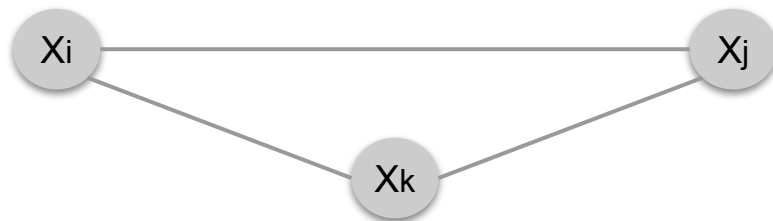
```
1 function AC-3(csp) return falso si se encuentra inconsistencia, sino verdadero
2   cola ← cola con todos los arcos de csp
3   while cola no es vacia do
4     (Xi,Xj) ← cola.desencolar()
5     if AC-WITH(csp,Xi,Xj) then
6       if csp.Di es vacio then return falso
7       for each Xk in csp.vecinos(Xi) do
8         cola.encolar(Xk,Xi)
9   return verdadero
```

Si AC-WITH redujo el dominio de X_i , entonces:

- Si el dominio queda vacío, entonces el CSP no tiene solución.
- Sino, el cambio puede habilitar más reducciones en los dominios de todo vecino X_k de X_i , incluso si X_k ya ha sido considerado anteriormente.

Consistencia de caminos



Un conjunto de dos variables distintas $\{X_i, X_j\}$ es **camino-consistente** con otra variable X_k si, para toda asignación $\{X_i = a, X_j = b\}$ consistente con la restricción sobre el arco (X_i, X_j) , existe una asignación para X_k consistente con la restricción sobre el arco (X_i, X_k) y (X_k, X_j) .



Ejemplos

— — —

📖 Ejemplo - Coloreo de Australia (***)

Considerar una variante donde podemos usar únicamente los colores  y . Es decir, el dominio de todas las variables es $\{\text{red}, \text{blue}\}$.

La consistencia de arco no hace nada porque todas las variables ya son arco-consistentes.

Para hacer algún progreso, hay que ver la consistencia de caminos. En este sentido, $\{W, S\}$ no es camino consistente con N , pues las únicas asignaciones para $\{W, S\}$ consistentes con la restricción $W \neq S$ son:

- A. $\{W = \text{red}, S = \text{blue}\}$
- B. $\{W = \text{blue}, S = \text{red}\}$

y entonces no hay asignación posible para N consistente con $W \neq N$ y $N \neq S$:

- Si $N = \text{red}$, entonces viola la restricción $W \neq N$ para A y $N \neq S$ para B.
- Si $N = \text{blue}$, entonces viola la restricción $N \neq S$ para A y $W \neq N$ para B.

Luego no hay asignación posible para $\{W, S\}$ y el CSP no tiene solución.

Consistencia de caminos

Para imponer consistencia de caminos se utiliza un algoritmo llamado **PC-2** (Mackworth, 1977).

Su funcionamiento es muy similar a AC-3, así que no lo vamos a ver.

k -consistencia

Todo esto se puede generalizar 🤖 ...

Un CSP es **k -consistente** si para todo conjunto de $k-1$ variables distintas $\{X_1, \dots, X_{k-1}\}$ y para toda asignación consistente con las restricciones involucradas en esas variables, se puede asignar un valor consistente a otra variable X_k .

- 1-consistencia es consistencia de nodos.
- 2-consistencia es consistencia de arcos.
- 3-consistencia es consistencia de caminos.

k -consistencia fuerte

— — —

Un CSP es **fuertemente k -consistente** si es k -consistente, $(k-1)$ -consistente, ..., 2-consistente y 1-consistente.

Observación. Supongamos que un CSP con n nodos es fuertemente n -consistente. Entonces podemos resolverlo fácilmente de la siguiente forma:

- Elegir un valor consistente para X_1 .
- Por 2-consistencia, debe existir un valor para X_2 consistente con la restricción (X_1, X_2) .
- Por 3-consistencia, debe existir un valor para X_3 consistente con las restricciones (X_1, X_3) y (X_2, X_3) . Y así con el resto de las variables...

Es decir, para cada X_i , tenemos que buscar en su dominio un valor consistente con todas las restricciones que relacionan a X_i con las variables consideradas anteriormente.

k -consistencia fuerte

— — —

- Lamentablemente, **todo algoritmo para imponer k -consistencia requiere en el peor caso un consumo de memoria y de tiempo que es exponencial en k .**
- En la práctica, determinar el nivel apropiado de consistencia es una ciencia experimental.
- Comúnmente, se usa 2-consistencia y con menor frecuencia 3-consistencia.

Restricciones globales

— — —

Las restricciones globales pueden ser manejadas por algoritmos específicos que son más eficientes que los métodos generales presentados hasta ahora.

- La restricción

$$\langle (X_1, \dots, X_m), \textit{Alldiff}(X_1, \dots, X_m) \rangle$$

representa que las variables X_1, \dots, X_m deben tener valores distintos.

- La restricción

$$\langle (X_1, \dots, X_m), \textit{Atmost}(K, X_1, \dots, X_m) \rangle$$

representa que los valores de X_1, \dots, X_m suman a lo sumo K , siendo K alguna constante numérica. Es decir,

$$\langle (X_1, \dots, X_m), X_1 + \dots + X_m \leq K \rangle$$

Ejemplo

— — —

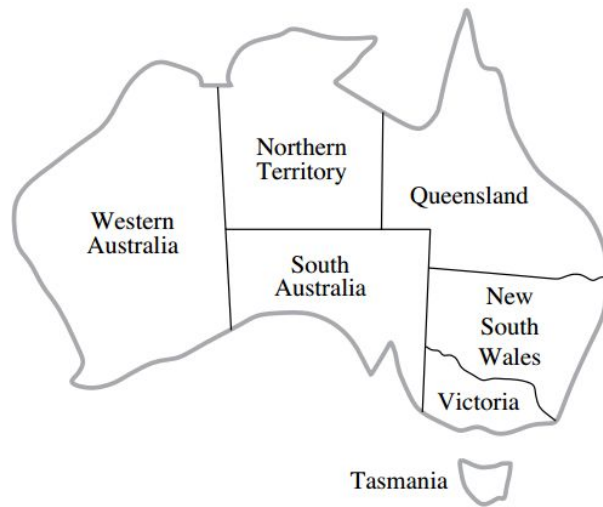
📖 Ejemplo - Coloreo de Australia (***).

Considerar la variante donde hay dos colores disponibles: ● y ●.

Las regiones Western Australia, Northern Territory y South Australia son mutuamente fronterizas, es decir, vale la restricción:

$$\langle (W, N, S), \text{Alldiff}(W, N, S) \rangle$$

Dado que hay 3 variables involucradas y tan solo 2 colores disponibles, podemos inferir que el CSP no tiene solución.



Restricciones globales – *Alldiff*

— — —

$$\langle (X_1, \dots, X_m), \textcolor{red}{Alldiff}(X_1, \dots, X_m) \rangle$$

Test de inconsistencia. Si en los dominios hay n posibles valores distintos para X_1, \dots, X_m y $n < m$, entonces la restricción no se puede verificar.



Ejemplo

— — —

Consideremos un CSP con tres variables X , Y y Z con dominio $\{5,6,7,8,9\}$, relacionados por medio de la siguiente restricción:

$$\langle (X,Y,Z), \textit{Atmost}(14,X,Y,Z) \rangle$$

De los dominios, sabemos que el menor valor que puede asumir cada una de las variables es 5. Es decir, podemos inferir que las variables verifican la desigualdad

$$X + Y + Z > 14$$

Luego, la desigualdad

$$X + Y + Z \leq 14$$

no se puede verificar y el CSP no tiene solución.

Restricciones globales – *Atmost*

— — —

$$\langle (X_1, \dots, X_m), \textbf{Atmost}(K, X_1, \dots, X_m) \rangle$$

- **Test de inconsistencia.** Si la suma del menor valor del dominio de cada una de las variables X_1, \dots, X_m es mayor a K , entonces la restricción no se puede verificar.



Resumen

— — —

- ❑ Un CSP representa un estado como una **asignación** de valores a un conjunto de **variables**, donde cada variable tiene un **dominio** y las condiciones que tienen que cumplir las soluciones se representan con un conjunto de **restricciones**.
- ❑ Muchos problemas importantes del mundo real se pueden escribir como CSPs.
- ❑ Vimos diversas técnicas de **inferencia** que usan a las restricciones para reducir el dominio de las variables, por ejemplo: consistencia de nodos, de arcos, de caminos y k-consistencia.



Próximamente

— — —

Muchos CSPs no se pueden resolver únicamente con inferencia...
Llega un momento en donde tenemos que buscar.

Veremos **algoritmos de búsqueda** para CSPs.

Formulación para CSPs

— — —

- **Estados.** Asignaciones parciales.
- **Estado inicial.** Asignación vacía {}.
- **Acciones.**

$X_i = val$: representa agregar a la asignación parcial la variable X_i con valor $val \in D_i$, siempre que la asignación sea consistente y la variable X_i no haya sido asignada previamente.

- **Modelo transicional.**

$RESULTADO(\{X_{i1}=v_1, \dots, X_{ij}=v_r\}, X_{ik}=v_k) = \{X_{i1}=v_1, \dots, X_{ij}=v_r, X_{ik}=v_k\}$

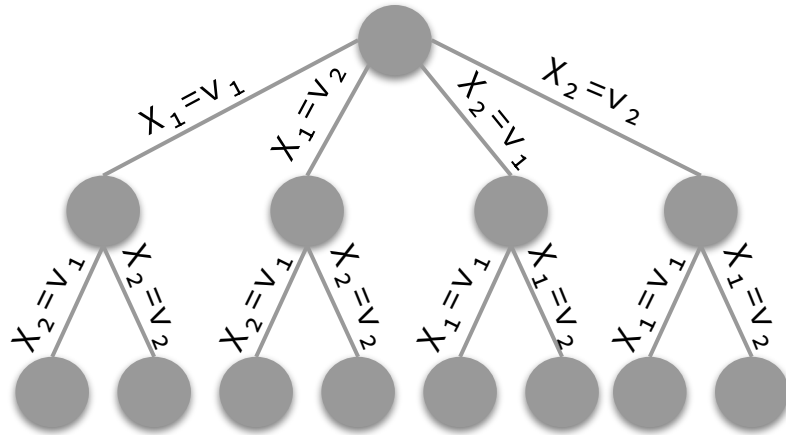
- **Test objetivo.** ¿La asignación es completa?
- **Costo de camino.** Irrelevante.



Ejemplo

— — —

Para un CSP con 2 variables, cada una con 2 valores en su dominio, el número de asignaciones completas es $2^2 = 4$ pero el árbol de búsqueda tendría $2!.2^2 = 8$ hojas.



Tamaño del árbol de búsqueda

- Considerar un CSP con n variables de dominio con tamaño a lo sumo d .
- Podríamos aplicar cualquier algoritmo de búsqueda no informada, pero...
- El factor de ramificación en la raíz es $n.d$, en el siguiente nivel es $(n-1).d$ y así...
- Este árbol de búsqueda tendría $n!.d^n$ hojas, mientras que la cantidad de asignaciones completas distintas es d^n .

Conmutatividad

- Los CSPs tienen la propiedad de **conmutatividad**: el orden de aplicación de las acciones no tiene efecto en la salida.
- Entonces, basta con **ordenar** las variables y en cada nivel del árbol asignar un valor a la siguiente variable.

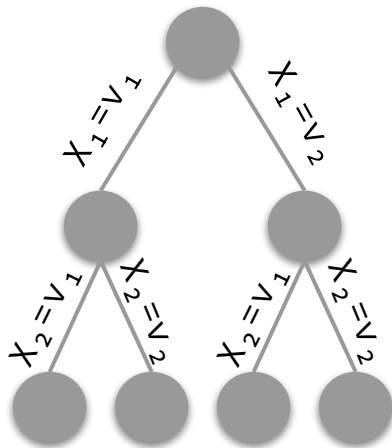


Ejemplo

— — —

Considerar un CSP con 2 variables, cada una con 2 valores en su dominio, donde primero se asigna la variable X_1 y luego X_2 .

El número de asignaciones completas es $2^2 = 4$ y el árbol de búsqueda tiene $2^2 = 4$ hojas.



Conmutatividad

- 📖 **Ejemplo - Coloreo de Australia.** En el nodo raíz podemos elegir entre las acciones

$$S = \text{red}, S = \text{green} \text{ y } S = \text{blue},$$

pero nunca elegimos entre

$$S = \text{red} \text{ y } N = \text{blue}.$$

- Teniendo en cuenta la conmutatividad, el número de hojas es d^n y coincide con el número de asignaciones completas.

Búsqueda vuelta atrás

Backtracking search

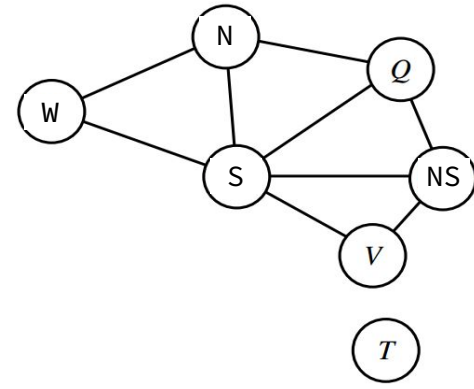
La **búsqueda vuelta atrás** es una búsqueda primero en profundidad que asigna valores para **una variable a la vez** y vuelve hacia atrás cuando a una variable no le quedan valores consistentes para asignar.

— — —



Ejemplo – Coloreo de Australia

{}

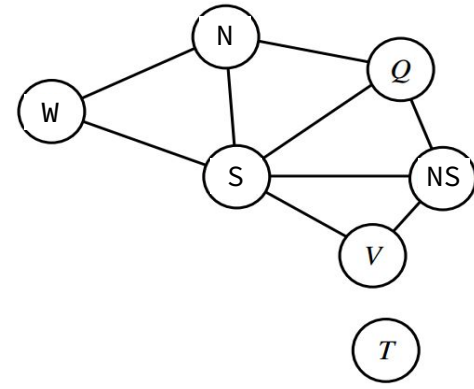




Ejemplo – Coloreo de Australia

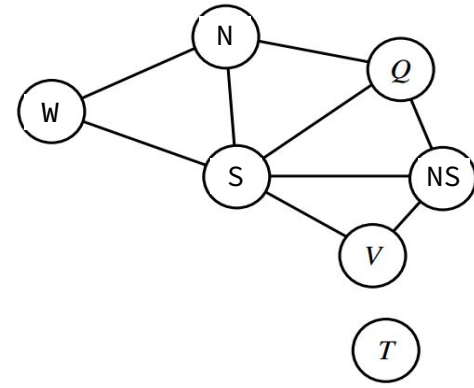
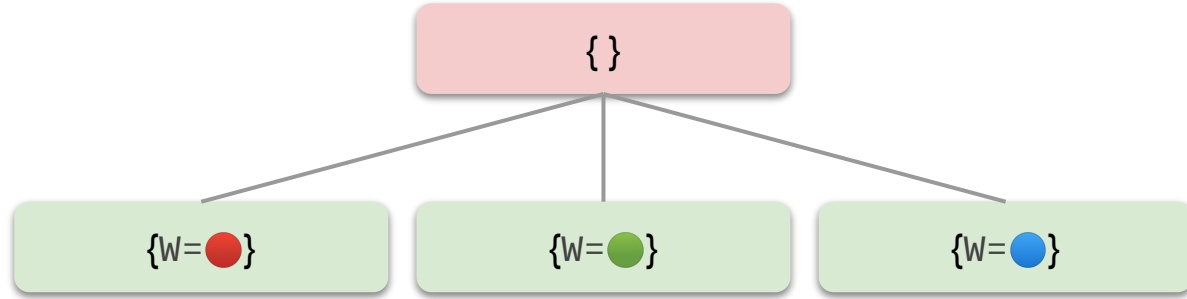
— — —

{ }



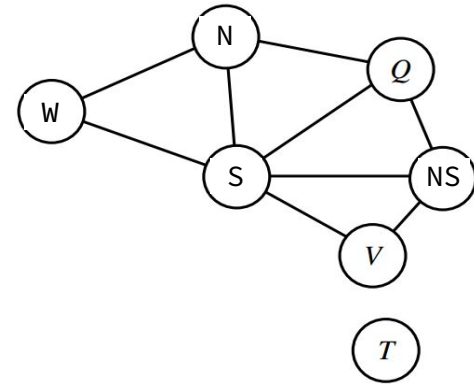
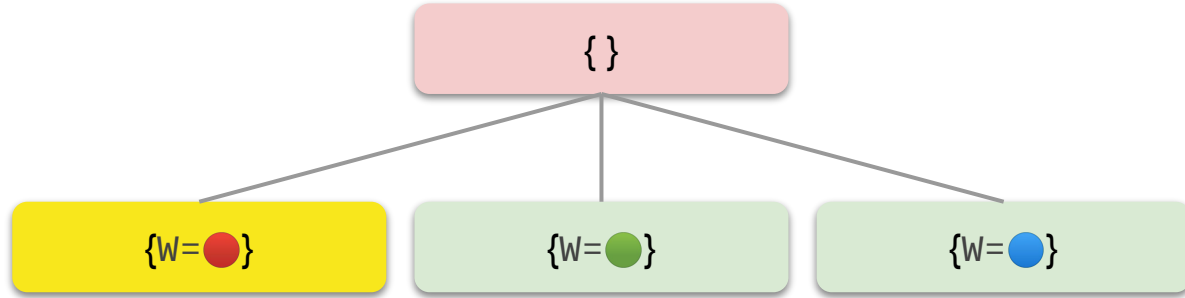


Ejemplo – Coloreo de Australia



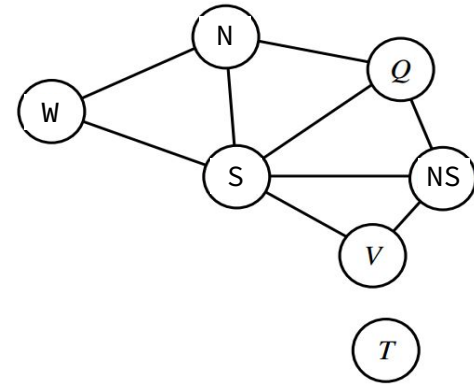
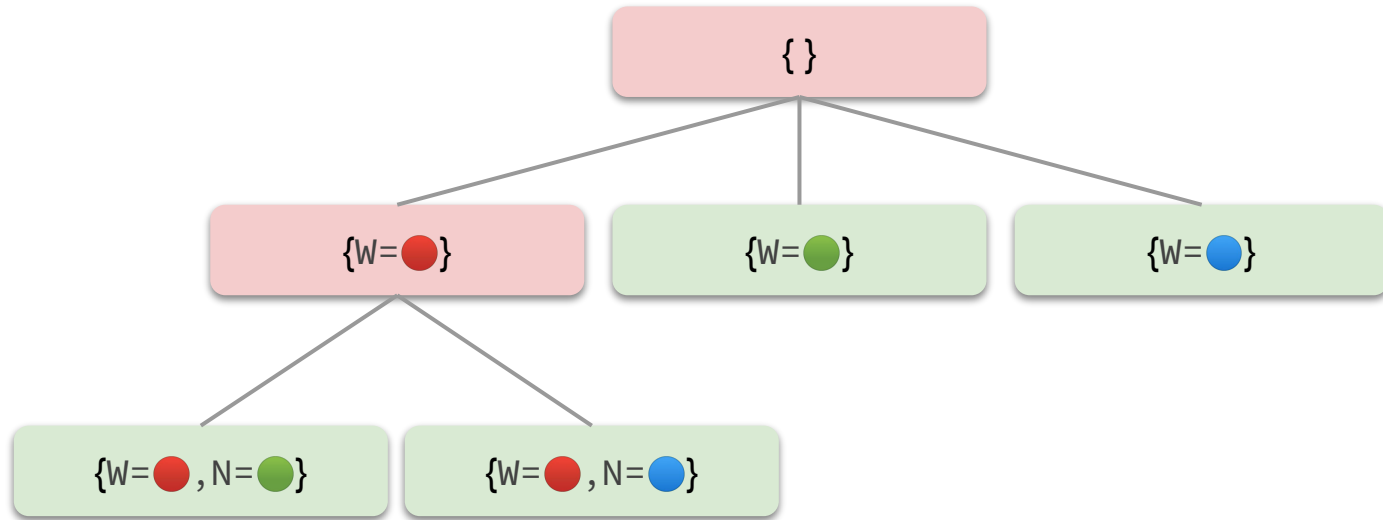


Ejemplo – Coloreo de Australia



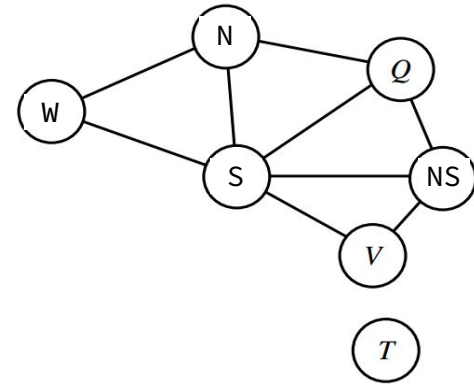
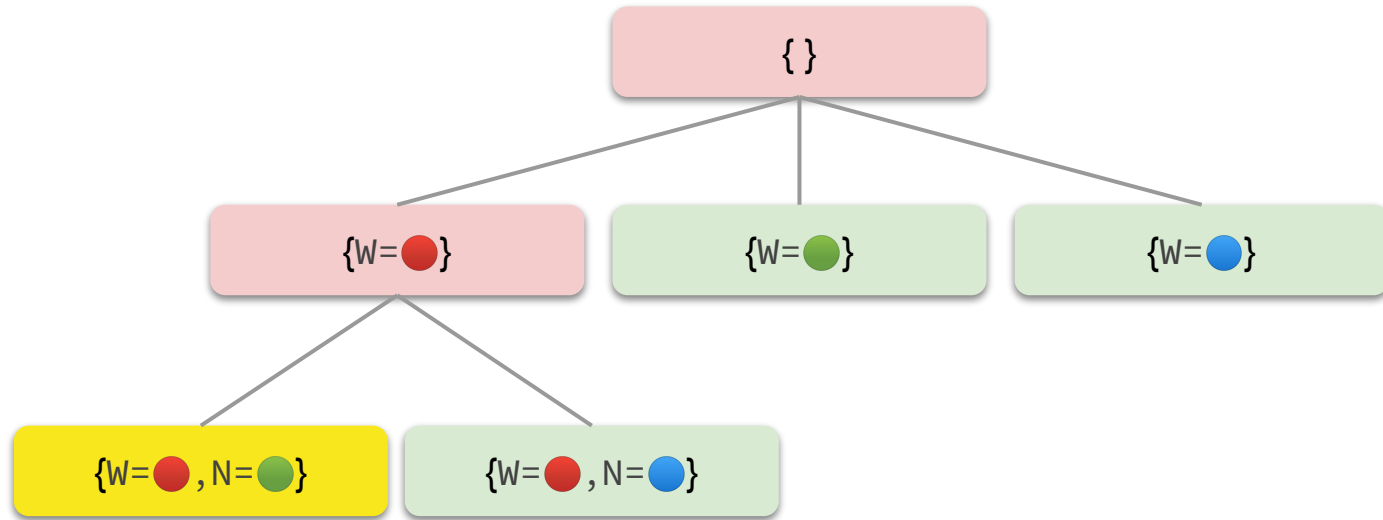


Ejemplo – Coloreo de Australia



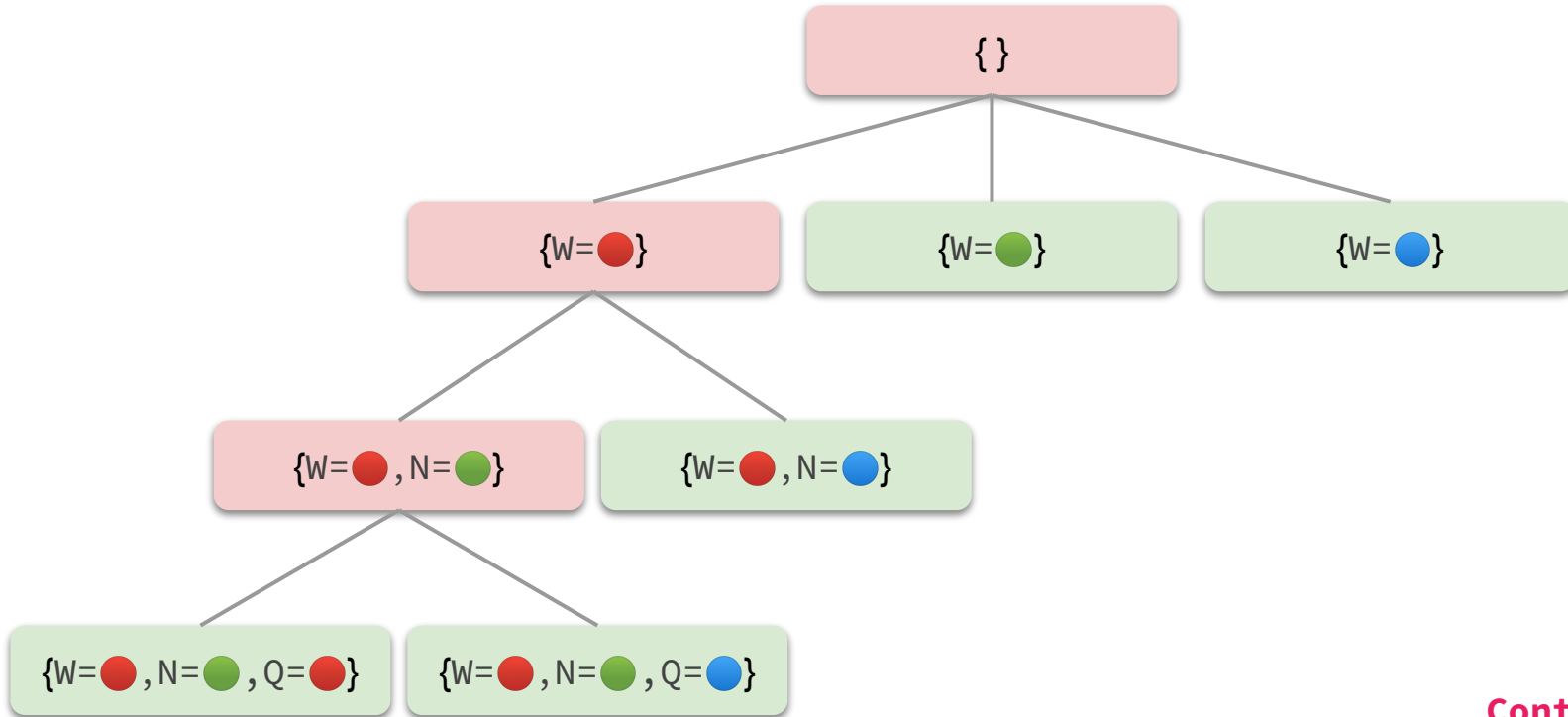
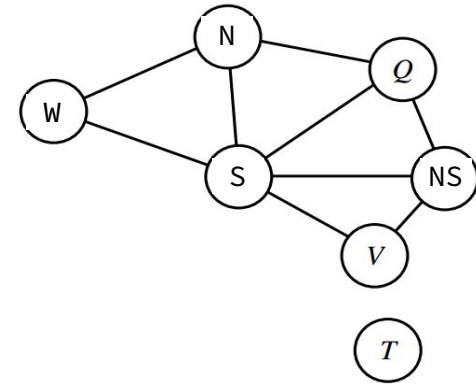


Ejemplo – Coloreo de Australia





Ejemplo – Coloreo de Australia



Continúa...

Algoritmo general de búsqueda vuelta atrás (sin inferencia)

— — —

```
1 function BACKTRACK(csp, assign) return solución o fallo
2   if csp.es_completa(assign) then return assign
3    $X_i \leftarrow \text{ELEGIR\_VAR}(\textit{csp}, \textit{assign})$ 
4   for each x in ORDENAR_DOM(csp, assign,  $X_i$ ) do
5     assign[ $X_i$ ]  $\leftarrow x$ 
6     if csp.es_consistente(assign) then
7       res  $\leftarrow$  BACKTRACK(csp, assign)
8       if res  $\neq$  fallo then return res
9     assign.pop( $X_i$ )
10  return fallo
```

```
1 function BACKTRACKING-SEARCH(csp) return solución o fallo
2   return BACKTRACK(csp, {})
```

Algoritmo general de búsqueda vuelta atrás (sin inferencia)

— — —

```
1 function BACKTRACK(csp, assign) return solución o fallo
2   if csp.es_completa(assign) then return assign
3    $X_i \leftarrow \text{ELEGIR\_VAR}(\text{csp}, \text{assign})$ 
4   for each x in ORDENAR_DOM(csp, assign,  $X_i$ ) do
5     assign[ $X_i$ ]  $\leftarrow x$ 
6     if csp.es_consistente(assign) then
7       res  $\leftarrow \text{BACKTRACK}(\text{csp}, \text{assign})$ 
8       if res  $\neq$  fallo then return res
9     assign.pop( $X_i$ )
10  return fallo
```

ELEGIR_VAR elige una variable aún no asignada, con algún criterio.

ORDENAR_DOM ordena los valores del dominio de X_i , con algún criterio.

Llamada recursiva. Si devuelve un fallo, la llamada anterior intentará otro valor para la variable elegida.

```
1 function BACKTRACKING-SEARCH(csp) return solución o fallo
2   return BACKTRACK(csp, {})
```

Algoritmo general de búsqueda vuelta atrás (con inferencia)

— — —

```
1 function BACKTRACK(csp, asign) return solución o fallo
2   if csp.es_completa(asign) then return asign
3    $X_i \leftarrow \text{ELEGIR\_VAR}(\textit{csp}, \textit{asign})$ 
4   for each x in ORDENAR_DOM(csp, asign,  $X_i$ ) do
5     asign[ $X_i$ ]  $\leftarrow x$ 
6     if csp.es_consistente(asign) then
7       infers  $\leftarrow \text{INFERIR}(\textit{csp}, \textit{asign}, X_i)$ 
8       if infers  $\neq$  fallo then
9         agregar infers a csp
10        res  $\leftarrow \text{BACKTRACK}(\textit{csp}, \textit{asign})$ 
11        if res  $\neq$  fallo then return res
12        eliminar infers de csp
13    asign.pop( $X_i$ )
14  return fallo
```


Algoritmo general de búsqueda vuelta atrás (con inferencia)

— — —

```
1 function BACKTRACK(csp, asign) return solución o fallo
2   if csp.es_completa(asign) then return asign
3   Xi ← ELEGIR_VAR(csp, asign)
4   for each x in ORDENAR_DOM(csp, asign, Xi) do
5     asign[Xi] ← x
6     if csp.es_consistente(asign) then
7       infers ← INFERIR(csp, asign, Xi)
8       if infers ≠ fallo then
9         agregar infers a csp
10        res ← BACKTRACK(csp, asign)
11        if res ≠ fallo then return res
12        eliminar infers de csp
13    asign.pop(Xi)
14  return fallo
```

INFERIR intenta reducir el dominio de las variables, por ejemplo imponiendo consistencia de nodos, arcos o caminos.

Si la inferencia detecta que la asignación parcial no lleva a una solución, entonces devuelve un fallo.

De lo contrario, devuelve un objeto que guarda las inferencias hechas para luego agregarlas al CSP.

Observación

— — —

BACKTRACKING-SEARCH utiliza implícitamente un estado inicial, una función de acciones, un modelo transicional y un test objetivo que **no dependen del problema específico**.

Esto se debe a que la representación de los CSPs está **estandarizada**.

Backtracking mejorado

— — —

Recordar que los algoritmos de búsqueda no informada se podían mejorar con funciones heurísticas derivadas de nuestro conocimiento del problema.

La ventaja del backtracking es que podemos mejorarlo **sin tener conocimiento específico del dominio**.

1. ¿Cuál variable elegir a continuación (**ELEGIR_VAR**)?
2. ¿En qué orden se deberían intentar sus valores (**ORDENAR_DOM**)?
3. ¿Qué inferencias hacer (**INFERIR**)?

1. Elección de variable

— — —

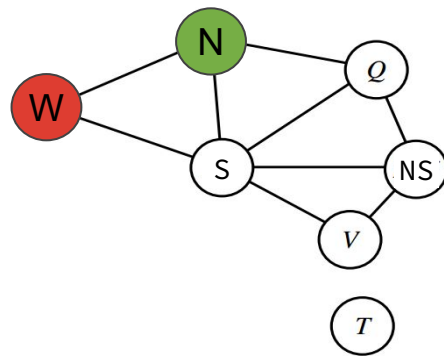
- **Predeterminado** o **aleatorio**. 🙅
- **Heurística de mínimos valores restantes**. Elegir la variable con la menor cantidad de valores legales.

📖 Ejemplo - Coloreo de Australia.

Luego de la asignación $\{W=\text{rojo}, N=\text{verde}\}$, queda un único valor posible para S. Tiene sentido asignar $S=\text{azul}$ antes que las demás variables.

De hecho, luego de asignar a S, las elecciones para Q, NS y V quedan determinadas.

Pero no ayuda a elegir la primera región a colorear...



1. Elección de variable

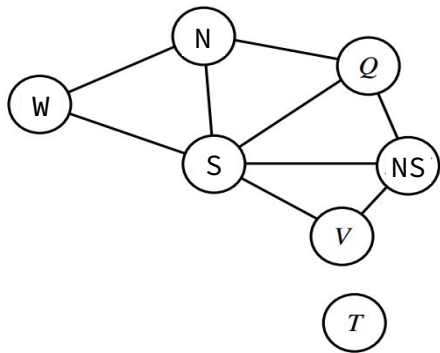
— — —

- **Heurística de grado.** Elegir la variable involucrada en el mayor número de restricciones sobre otras variables no asignadas.

📖 Ejemplo - Coloreo de Australia.

Inicialmente,

- S tiene grado 5,
- N, Q y NS tienen grado 3,
- W y V tienen grado 2,
- T tiene grado 0.



Intenta reducir el factor de ramificación en elecciones futuras y suele usarse como **regla de desempate**.

2. Ordenamiento de valores

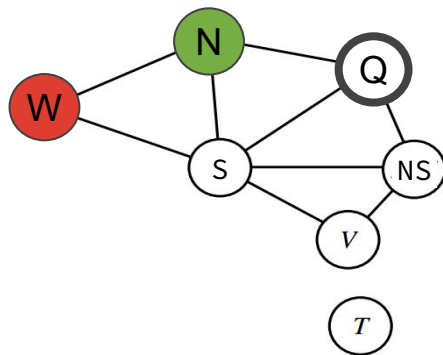
— — —

- **Heurística de valor menos restringido.** Prefiere el valor que descarta la menor cantidad de opciones para las variables vecinas en el grafo de restricciones.

Ejemplo - Coloreo de Australia.

Suponer la asignación $\{W=\text{red}, N=\text{green}\}$ y que la siguiente variable elegida es Q. Hay dos opciones para Q:

- $Q=\text{blue}$ descarta el último valor legal que le queda a S.
- $Q=\text{red}$ no descarta otros valores legales de S, luego es preferible probar este color en primer lugar.



Esta heurística intenta dejar la mayor flexibilidad para las asignaciones posteriores.

Observación

En general,

- Elegir la variable con el menor número de valores restantes ayuda a **minimizar el número de nodos en el árbol de búsqueda** al podar tempranamente grandes partes del árbol, y
- Dado que **solo se necesita una solución**, tiene sentido intentar primeramente con los valores más probables para las variables.

3. Inferencia

Inferencia por comprobación hacia adelante.

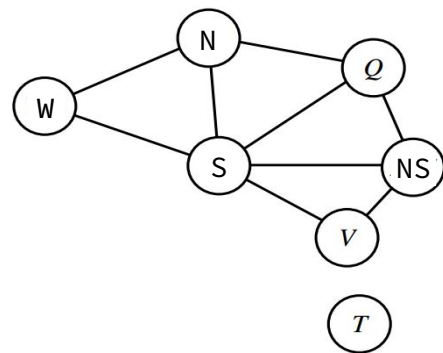
Cada vez que se asigna un valor v para una variable X_i , se debe imponer la arco-consistencia de X_i :

Para toda variable no asignada X_j vecina de X_i , se debe eliminar del dominio de X_j todo valor que sea inconsistente con v .



Ejemplo – Coloreo de Australia

— — —



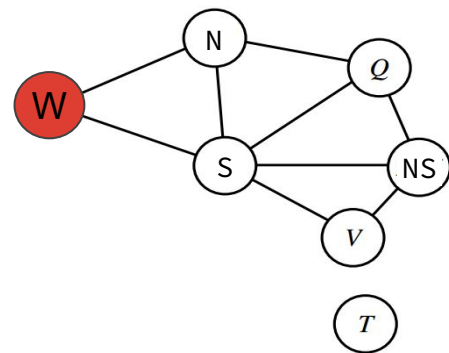
Progreso de la búsqueda hacia atrás con inferencia por comprobación hacia adelante:

	W	N	Q	NS	V	S	T	
Dominios iniciales	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>



Ejemplo – Coloreo de Australia

— — —



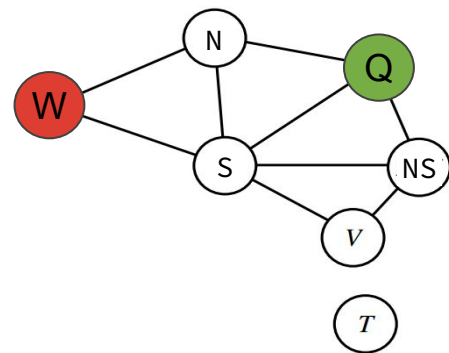
Progreso de la búsqueda hacia atrás con inferencia por comprobación hacia adelante:

	W	N	Q	NS	V	S	T
Dominios iniciales							
Luego de W=							



Ejemplo – Coloreo de Australia

— — —



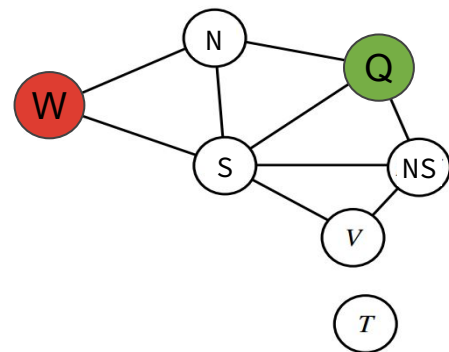
Progreso de la búsqueda hacia atrás con inferencia por comprobación hacia adelante:

	W	N	Q	NS	V	S	T
Dominios iniciales	●●●	●●●	●●●	●●●	●●●	●●●	●●●
Luego de W=●	●	●●	●●●	●●●	●●●	●●	●●●
Luego de Q=●	●	●	●	●●	●●●	●	●●●



Ejemplo – Coloreo de Australia

— — —



Progreso de la búsqueda hacia atrás con inferencia por comprobación hacia adelante:

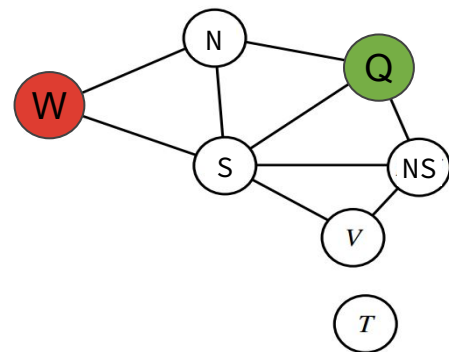
	W	N	Q	NS	V	S	T
Dominios iniciales	●●●	●●●	●●●	●●●	●●●	●●●	●●●
Luego de W=●	●	●●	●●●	●●●	●●●	●●	●●●
Luego de Q=●	●	●	●	●●	●●●	●	●●●

Los dominios de N y S se redujeron a un único valor. La comprobación hacia adelante eliminó la ramificación sobre estas variables.



Ejemplo – Coloreo de Australia

— — —



Progreso de la búsqueda hacia atrás con inferencia por comprobación hacia adelante:

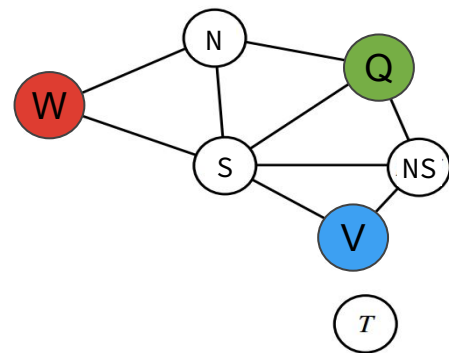
	W	N	Q	NS	V	S	T
Dominios iniciales	●●●	●●●	●●●	●●●	●●●	●●●	●●●
Luego de W=●	●	●●	●●●	●●●	●●●	●●	●●●
Luego de Q=●	●	●	●	●●	●●●	●	●●●

La asignación $\{W=\text{red}, Q=\text{green}\}$ fuerza a que N y S sean ●, pero N y S son adyacentes. La comprobación hacia adelante no es capaz de detectar esta inconsistencia.



Ejemplo – Coloreo de Australia

— — —



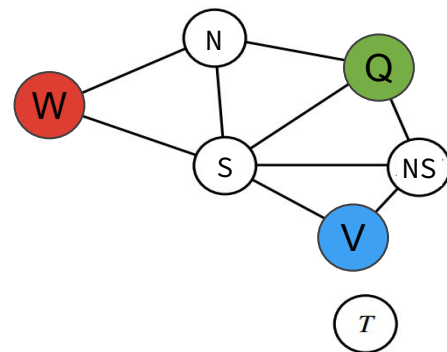
Progreso de la búsqueda hacia atrás con inferencia por comprobación hacia adelante:

	W	N	Q	NS	V	S	T
Dominios iniciales	●●●	●●●	●●●	●●●	●●●	●●●	●●●
Luego de W=●	●	●●	●●●	●●●	●●●	●●	●●●
Luego de Q=●	●	●	●	●●	●●●	●	●●●
Luego de V=●	●	●	●	●	●		●●●



Ejemplo – Coloreo de Australia

— — —



Progreso de la búsqueda hacia atrás con inferencia por comprobación hacia adelante:

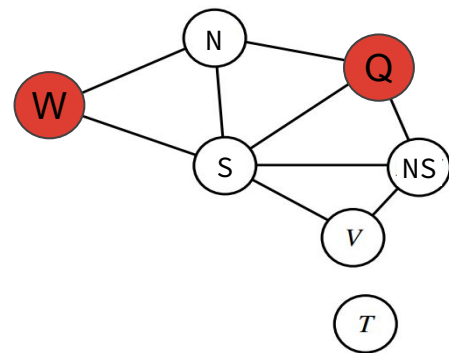
	W	N	Q	NS	V	S	T
Dominios iniciales	●●●	●●●	●●●	●●●	●●●	●●●	●●●
Luego de W=●	●	●●	●●●	●●●	●●●	●●	●●●
Luego de Q=●	●	●	●	●●	●●●	●	●●●
Luego de V=●	●	●	●	●	●		●●●

El dominio de S quedó vacío. La comprobación hacia adelante detectó que la asignación $\{W=●, Q=●, V=●\}$ es inconsistente y el algoritmo retrocede inmediatamente.



Ejemplo – Coloreo de Australia

— — —



Progreso de la búsqueda hacia atrás con inferencia por comprobación hacia adelante:

	W	N	Q	NS	V	S	T	
Dominios iniciales	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
Luego de W= <div></div>	<div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
Luego de Q= <div></div>	<div><div></div></div>	<div><div></div><div></div></div>	<div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
Continúa...								

3. Inferencia

La inferencia por comprobación hacia adelante detecta muchas inconsistencias, pero no todas. El problema es que fuerza la arco-consistencia únicamente sobre la variable elegida.

Mantenimiento de Arco-Consistencia (MAC).

Cada vez que se asigna un valor a una variable X_i , se llama al algoritmo AC-3, pero con una cola que contiene un arco (X_i, X_j) para toda variable no asignada X_j que sea vecina de X_i .

Es estrictamente más poderoso que la comprobación hacia adelante, pero más costosa.



Ejemplo – Coloreo de Australia

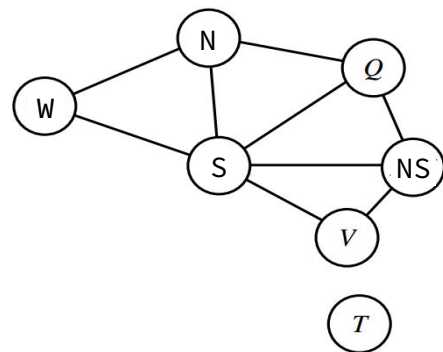
Vamos a aplicar la búsqueda hacia atrás con las siguientes mejoras:

1. La variable se elige con la heurística de mínimos valores restantes y el desempate es por la heurística de grados.
2. Los valores se ordenan con la heurística de valor menos restringido.
3. La inferencia es por comprobación hacia adelante.



Ejemplo – Coloreo de Australia

— — —



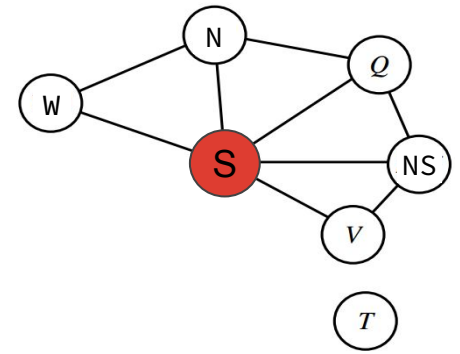
	W	N	Q	NS	V	S	T	
Dominios iniciales	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>





































1. Como todas las variables tienen el mismo dominio, por la regla de desempate elegimos la variable S de grado 5. Cualquier color para S descarta de sus vecinos la misma cantidad de valores, luego comenzamos por un color arbitrario, digamos ●.




Ejemplo – Coloreo de Australia

— — —



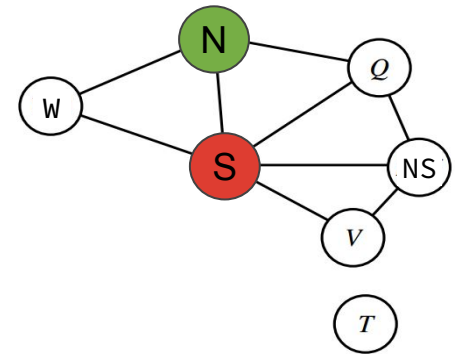
	W	N	Q	NS	V	S	T
Dominios iniciales	  	  	  	  	  	  	  
Luego de S= 	 	 	 	 	 		  

2. Como todas las variables no asignadas, salvo T, tienen el mismo dominio, por la regla de desempate elegimos alguna de las variables N, Q o NS de grado 2, digamos N. Cualquier color para N descarta de sus vecinos la misma cantidad de colores, podemos comenzar por un color arbitrario, digamos .



Ejemplo – Coloreo de Australia

— — —



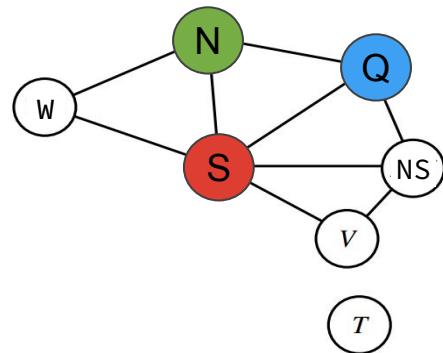
	W	N	Q	NS	V	S	T	
Dominios iniciales	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
Luego de S= <div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	
Luego de N= <div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	

3. Como las variables no asignadas W y Q tienen un único valor en su dominio, por la regla de desempate elegimos Q de grado 1.



Ejemplo – Coloreo de Australia

— — —



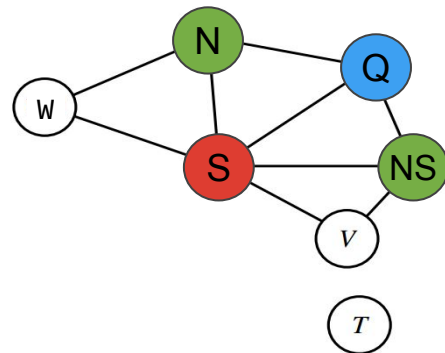
	W	N	Q	NS	V	S	T	
Dominios iniciales	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
Luego de S= <div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
Luego de N= <div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
Luego de Q= <div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>

4. Como las variables no asignadas W y NS tienen un único valor en su dominio, por la regla de desempate elegimos NS de grado 1.



Ejemplo – Coloreo de Australia

— — —



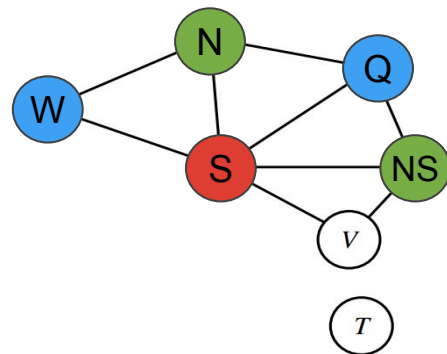
	W	N	Q	NS	V	S	T	
Dominios iniciales	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
Luego de S= <div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	
Luego de N= <div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	
Luego de Q= <div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	
Luego de NS= <div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	

5. Como las variables no asignadas W y V tienen un único valor en su dominio y ambas tienen grado 0, elegimos cualquiera de ella, digamos W.



Ejemplo – Coloreo de Australia

— — —



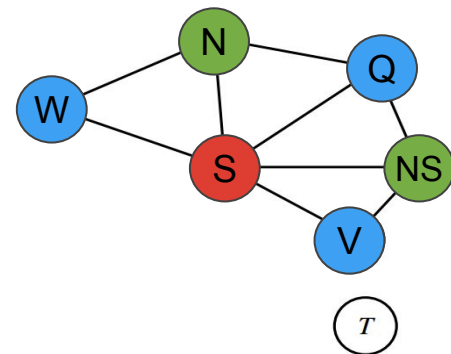
	W	N	Q	NS	V	S	T	
Dominios iniciales	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
Luego de S= <div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	
Luego de N= <div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	
Luego de Q= <div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	
Luego de NS= <div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	
Luego de W= <div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	

6. Elegimos la variable no asignada V que tiene un único valor en su dominio.



Ejemplo – Coloreo de Australia

— — —



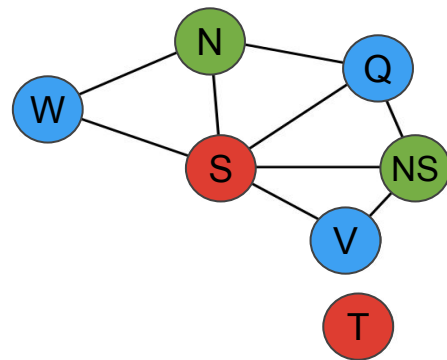
	W	N	Q	NS	V	S	T	
Dominios iniciales	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
Luego de S= <div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	
Luego de N= <div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	
Luego de Q= <div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	
Luego de NS= <div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	
Luego de W= <div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	
Luego de V= <div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	

7. Elegimos la variable T pues es la última variable sin asignar. Como no tiene vecinos, podemos comenzar por un color arbitrario, digamos ●.



Ejemplo – Coloreo de Australia

— — —



	W	N	Q	NS	V	S	T	
Dominios iniciales	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
Luego de S= <div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	
Luego de N= <div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	
Luego de Q= <div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	
Luego de NS= <div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	
Luego de W= <div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	
Luego de V= <div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div><div></div><div></div></div>	
Luego de T= <div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	

¡Encontramos una solución sin fallar, nunca volvimos hacia atrás!

Búsqueda local para CSPs

Para aplicar **búsqueda local** en CSPs, se necesita una **formulación de estado completo**.

- Los estados son asignaciones completas (tanto consistentes como inconsistentes).
- El estado inicial puede ser cualquiera.
- Las acciones cambian el valor de una de las variables.

Típicamente, el estado inicial viola muchas restricciones, pero **el objetivo de la búsqueda es minimizar el número de restricciones violadas**.

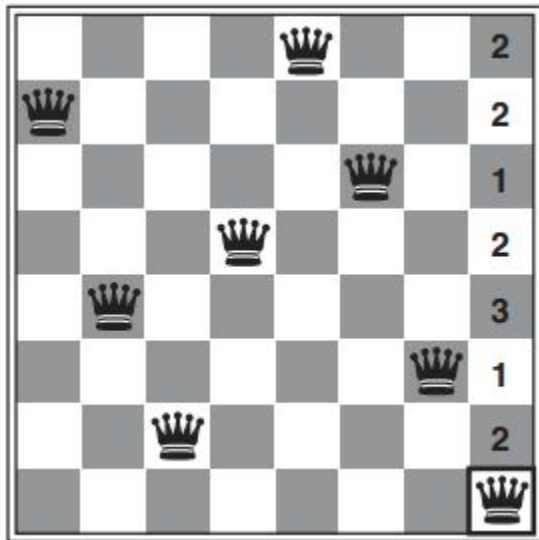
— — —

Mínimos-conflictos

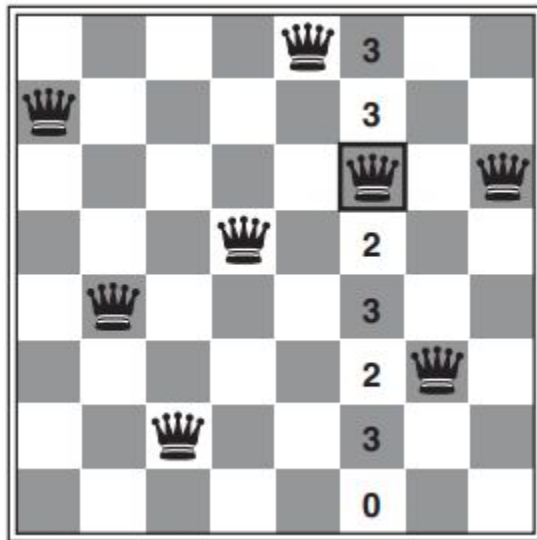
- Siendo dos variables vecinas X_i e X_j ya asignadas, decimos que X_i está en **conflicto** con X_j si sus valores violan la restricción sobre el arco (X_i, X_j) .
- Al elegir un nuevo valor para una variable, la heurística más obvia es aquella que elige el valor que resulte en el menor número de conflictos con las otras variables.
- La **heurística mínimos-conflictos** es un algoritmo de búsqueda local, que en cada iteración elige una variable en conflicto y le asigna aquel valor que resulte en el mínimo número de conflictos con las otras variables.



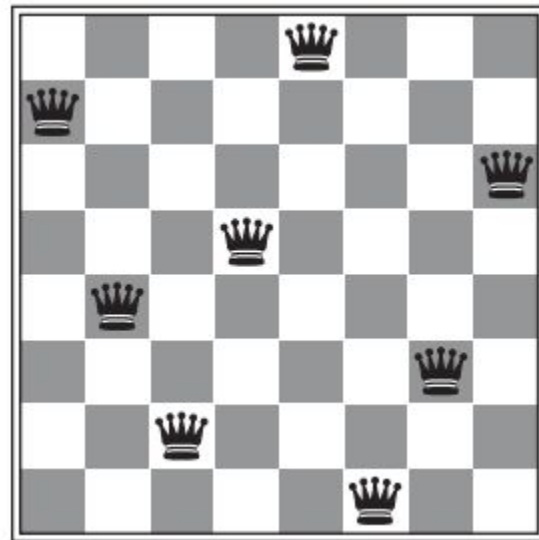
Ejemplo – 8 Reinas



1. Estado inicial. Elegimos la 8va reina y la movemos a la 3er fila, pues allí tiene un único conflicto (con la 6ta reina).



2. Elegimos la 6ta reina y la movemos a la 8va fila, pues allí no está en conflicto con ninguna otra reina.



3. Llegamos a un estado sin reinas en conflicto. ¡Encontramos una solución!

Algoritmo de mínimos-conflictos

— — —

```
1 function MIN-CONFLICTS(csp) return solución o fallo
2   actual ← una asignación inicial completa para csp
3   while no se cumple criterio de parada do
4     if actual es consistente then return actual
5      $X_i$  ← una variable de csp. $X$  en conflicto elegida al azar
6      $x$  ← el valor de  $X_i$  que minimiza NCONFLICTOS(csp, actual,  $X_i$ ,  $x$ )
7     actual[ $X_i$ ] ←  $x$ 
8   return fallo
```

Algoritmo de mínimos-conflictos

— — —

Puede ser al azar o a partir de un procedimiento greedy: en cada iteración elegir una variable nueva y asignarle el valor de mínimo conflicto respecto a esa asignación parcial.

```
1 function MIN-CONFLICTS(csp) return solución o fallo
2   actual ← una asignación inicial completa para csp
3   while no se cumple criterio de parada do
4     if actual es consistente then return actual
5     Xi ← una variable de csp.X en conflicto elegida al azar
6     x ← el valor x de Xi que minimiza NCONFLICTOS(csp, actual, Xi, x)
7     actual[Xi] ← x
8   return fallo
```

NCONFLICTOS es una función que cuenta el número de restricciones violadas por un valor particular, dado el resto de la asignación actual. Los valores que empatan en el número mínimo de conflictos se resuelven al azar.

Performance de MIN-CONFLICTS

- Es sorprendentemente efectivo para muchos CSPs.
- Por ejemplo, resuelve el problema del millón de reinas en un promedio de 50 pasos (luego de la asignación inicial).
- El paisaje de un CSP con la heurística de conflictos mínimos tiene usualmente muchas mesetas.
- Las técnicas vistas anteriormente para búsqueda local pueden ayudar a escapar de estas mesetas: movimientos laterales y lista tabú.

Estructura de los problemas

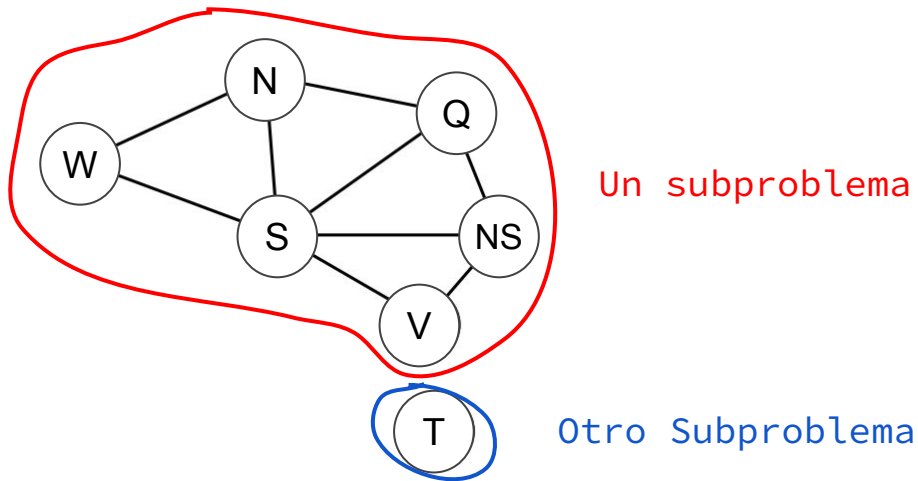
La dificultad para resolver un CSP está fuertemente relacionada con la **estructura** del grafo de restricciones.

Una de las estructuras más importantes son las **componentes conexas** del grafo.

— — —

📖 Ejemplo – Coloreo de Australia

Colorear Tasmania y colorear el continente son **subproblemas independientes** – cualquier solución para Tasmania combinada con cualquier solución para el continente conduce a una solución para el mapa completo.



Ventajas


Suponer que un CSP tiene n variables con d posibles valores y que se divide en k subproblemas independientes, cada uno de ellos con n/k variables.

La cantidad de asignaciones completas en el problema original es:

$$d^n$$

La cantidad de asignaciones completas en los subproblemas es:

$$k \cdot d^{n/k}$$

 **Ejemplo.** Para $n=80$, $d=2$ y $k=4$, pasamos de $1,2 \times 10^{24}$ a 4.194.304 asignaciones completas.

Teoría de grafos

La **teoría de grafos** se ocupa de estudiar la estructura de los grafos, en particular de los grafos de restricciones que representan a un CSP.

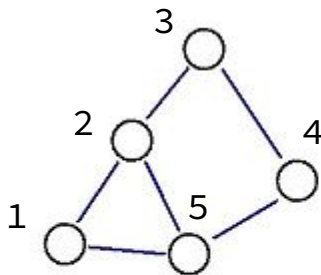
Para definir formalmente qué son las componentes conexas, debemos conocer algunos conceptos de teoría de grafos.

Subgrafo inducido

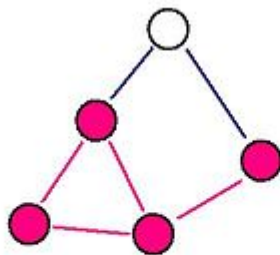
— — —

Dado un grafo $G=(V,E)$ y un subconjunto $U \subseteq V$, se define el **subgrafo inducido** $G[U]$ como aquél que tiene a U como conjunto de vértices y toda arista de G con extremos en U es una arista de $G[U]$.

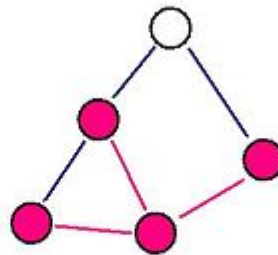
Ejemplos.



Grafo original



Subgrafo inducido por $\{1,2,5,4\}$.



No es subgrafo inducido porque falta la arista $(1,2)$.

Componente conexa

— — —

- Una **componente conexa** de un grafo es un subgrafo inducido que es:
 - **Conexo**. Todo par de vértices están conectados por un camino.
 - **Maximal**. Se desconecta al agregar cualquier otro vértice del grafo.
- **Cada componente conexa del grafo se corresponde con un subproblema independiente.**
- Existen algoritmos para encontrar las componentes conexas de un grafo, que están basados en DFS: algoritmo de Kosaraju, algoritmo de Tarjan, entre otros.



Resumen

— — —

- ❑ La **búsqueda hacia atrás**, basada en una búsqueda primero en profundidad, se usa habitualmente para resolver CSPs.
- ❑ La inferencia se puede intercalar con la búsqueda, mediante **comprobación hacia adelante** o con mecanismos más sofisticados como el **mantenimiento de arco-consistencia**.
- ❑ Las heurísticas de **mínimos valores restantes** y de **grados** se usan para elegir la próxima variable a asignar.
- ❑ La heurística de **valor menos restringido** se usa para decidir qué valor elegir primero para una variable dada.



Resumen

- ❑ La búsqueda local que usa la **heurística de mínimos-conflictos** ha tenido gran éxito para resolver CSPs.
- ❑ La dificultad para resolver un CSP depende fuertemente de la **estructura** del grafo de restricciones. Por ejemplo, las **componentes conexas** permiten dividir un CSP en **subproblemas independientes**.
- ❑ El formalismo CSP es muy potente. Para cada nuevo problema, sólo tenemos que definirlo en término de restricciones y luego podemos resolverlo con cualquier solver de CSPs.



Próximamente

— — —

Volveremos a hablar de algoritmos de búsqueda local. Nos ocuparemos de una familia de algoritmos que han demostrado ser muy efectivos y que tienen la particularidad de estar basados en leyes naturales.