

## Unidad 3

# Resolución de problemas mediante búsquedas y heurísticas

**Hugo Sanjurjo González**

**Sistemas Inteligentes**

Grado Dual en Industria Digital

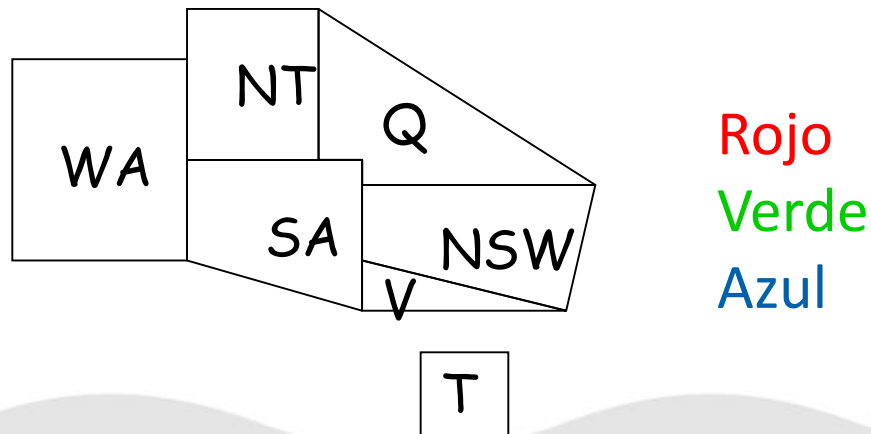
Curso 23/24



## 3.4 Problemas de Satisfacción de Restricciones

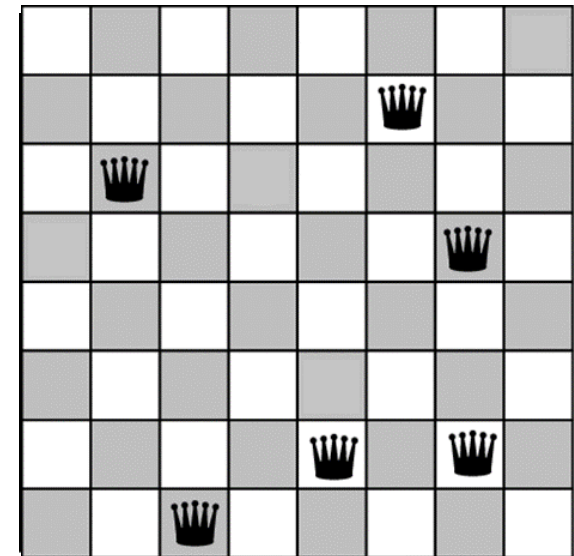
## Introducción (1)

- Problemas de satisfacción de restricciones.
  - *Constraint Satisfaction Problems (CSP)*
- Idea básica:
  - Retrasar las decisiones difíciles hasta que sean más sencillas.



## Introducción (2) – Formulación #1

- Estados: Reinas posicionadas correctamente (0, 1, 2, ... 8).
- Estado inicial: 0 reinas en el tablero
- Siguiendo estado: Añadir una reina en el tablero.
- Objetivo: 8 reinas colocadas sin atacarse las unas a las otras.



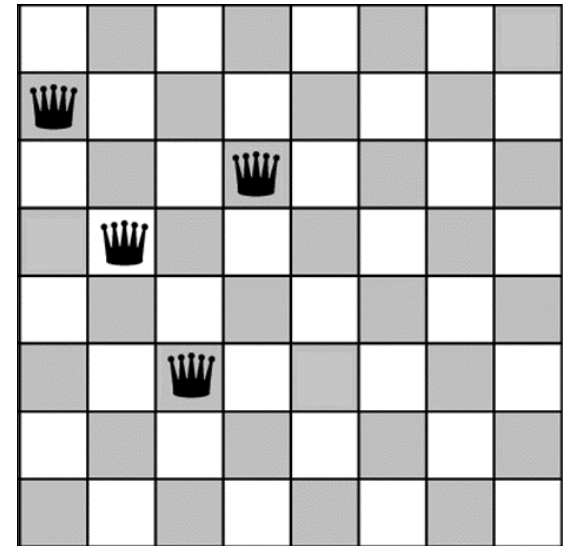
Para 8-reinas: 16.777.216 estados

Para 16-Reinas:

18.446.744.073.709.551.616

## Introducción (2) – Formulación #2

- Estados: Reinas posicionadas (0, 1, 2, ... 8) en la fila más a la izquierda sin atacarse las unas a las otras.
- Estado inicial: 0 reinas en el tablero
- Siguiendo estado: Añadiendo una reina en el tablero en la columna vacía más a la izquierda.
- Objetivo: 8 reinas colocadas sin atacarse las unas a las otras.



2057 estados

## Introducción (3)

- Algunos problemas tienen estados y objetivos que se conforman en base a un **representación simple y estructurada**.
- Este tipo de representación ve el problema como un **conjunto de variables** que necesitan unos **valores** para cumplir ciertas **restricciones**.



Fuente:  
<https://grupogreenmarket.com/catalogo/ninos-y-juegos/juego-de-figuras-geometricas.html>

## Observación reflexiva #1

- ¿Podemos resolver estos problemas escogiendo el orden apropiadamente?
- ¿Podemos incluso evitar tomar la decisión de escoger ese orden?

## Componentes del CSP (1)

- Cada problema tiene un patrón estándar:
  - Un conjunto de variables a los que hay que asignar valores que respeten una serie de restricciones.
- La función de los sucesores y la función objetivo pueden desarrollarse para cualquiera de estos problemas.
- Se trata de situaciones de propósito general que no requieren conocimiento concreto en un dominio específico.

## Componentes del CSP (2)

1. Conjunto de variables  $\{X_1, X_2, \dots, X_n\}$ .
  - Cada variable  $X_i$  tiene un dominio  $D_i$ , normalmente finito.
2. Conjunto de restricciones  $\{C_1, C_2, \dots, C_n\}$ .
  - Cada restricción  $C_i$  se relaciona con un subconjunto de variables especificando las combinaciones válidas de sus valores.
3. Objetivo:
  - Asignar un valor a cada variable de tal forma que las restricciones se cumplan.

## CSP finito VS CSP Infinito

- Finito:
  - Cada variable tiene un dominio finito de valores.
- Infinito:
  - Algunas o todas las variables tienen un dominio infinito.
  - Ej. Problemas de programación lineal sobre números reales.

$$\text{for } i = 1, 2, \dots, p : a_{i,1}x_1 + a_{i,2}x_2 + \dots + a_{i,n}x_n = a_{i,0}$$

$$\text{for } j = 1, 2, \dots, q : b_{j,1}x_1 + b_{j,2}x_2 + \dots + b_{j,n}x_n \leq b_{j,0}$$

## Conmutación en CSP

El orden en el cual se asignan valores a las variables no tiene impacto en la asignación alcanzada.

1. Los sucesores pueden escogerse seleccionando una variable y luego asignando cada valor en el dominio de esa variable.
  - Reducción en el factor de ramificación.
2. No es necesario guardar el camino a un nodo.
  - Algoritmo de búsqueda de vuelta atrás (*backtracking algorithm*)

## Observación reflexiva #2

- 3 variables  $X_1, X_2, X_3$
- 5 valores  $V_1, V_2, V_3, V_4, V_5$
- Una asignación puede ser
$$A = \{X_1 \leftarrow v_1, X_3 \leftarrow v_3\}$$
- ¿Cuáles son los sucesores de A?

## Observación reflexiva #2 - Solución

- 3 variables  $X_1, X_2, X_3$
- 5 valores  $V_1, V_2, V_3, V_4, V_5$
- Una asignación puede ser  
$$A = \{X_1 \leftarrow v_1, X_3 \leftarrow v_3\}$$
- ¿Cuáles son los sucesores de A?

$$\{X_1 \leftarrow v_1, X_3 \leftarrow v_3, X_4 \leftarrow v_2\}$$

$$\{X_1 \leftarrow v_1, X_3 \leftarrow v_3, X_4 \leftarrow v_4\}$$

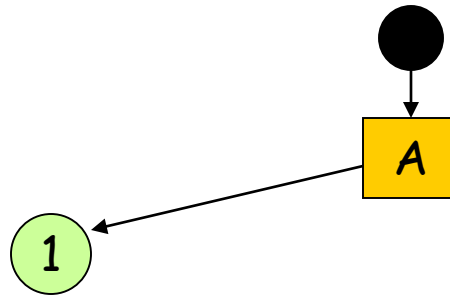
$$\{X_1 \leftarrow v_1, X_3 \leftarrow v_3, X_4 \leftarrow v_5\}$$

## Backtracking search (1)

- Esencialmente es un algoritmo de búsqueda en anchura simplificado usando recursión.
- Ej.
  - Variables: A, B y C
  - Dominio: {1,2,3,4}
  - Restricciones:
    - $A \leftarrow \{1, 4\}$
    - $B \leftarrow \{1\}$
    - $C \leftarrow \{2,3\}$

## Backtracking search (2)

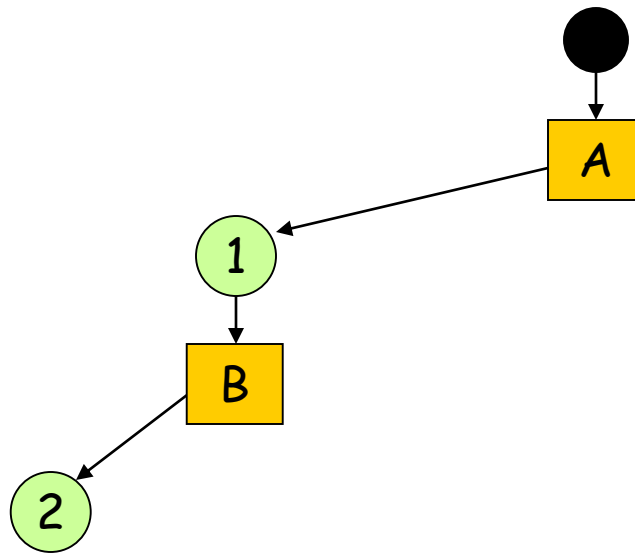
- Ej.



Asignación =  $\{(A,1)\}$

## Backtracking search (2)

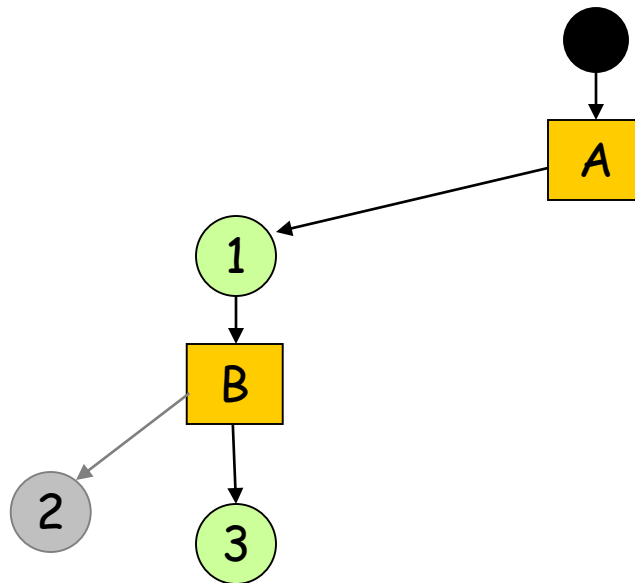
- Ej.



Asignación =  $\{(A,1), (B,2)\}$

## Backtracking search (2)

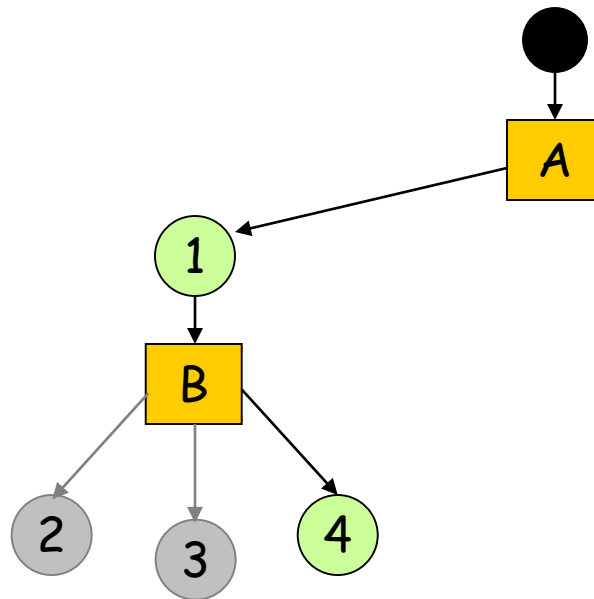
- Ej.



Asignación =  $\{(A,1), (B,3)\}$

## Backtracking search (2)

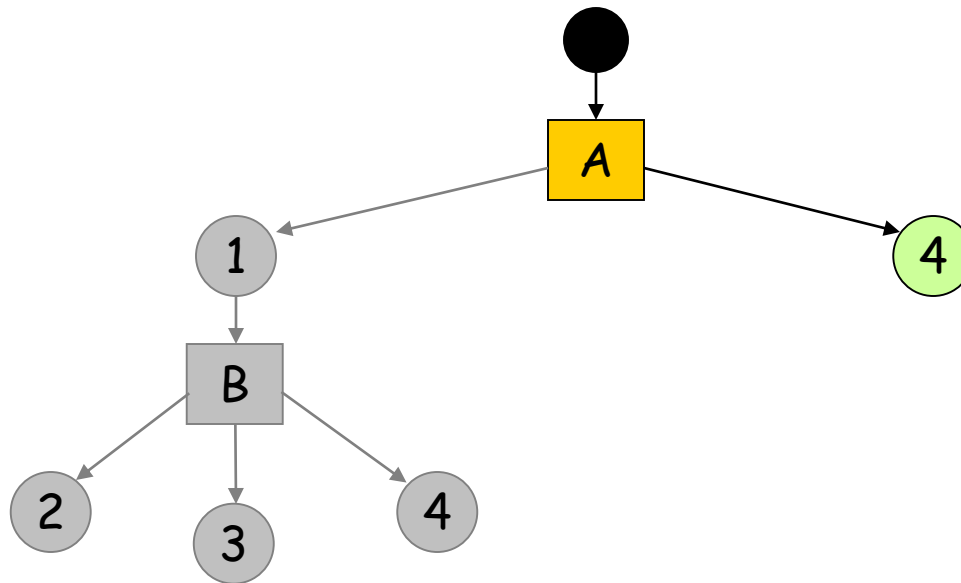
- Ej.



Asignación =  $\{(A,1), (B,4)\}$

## Backtracking search (2)

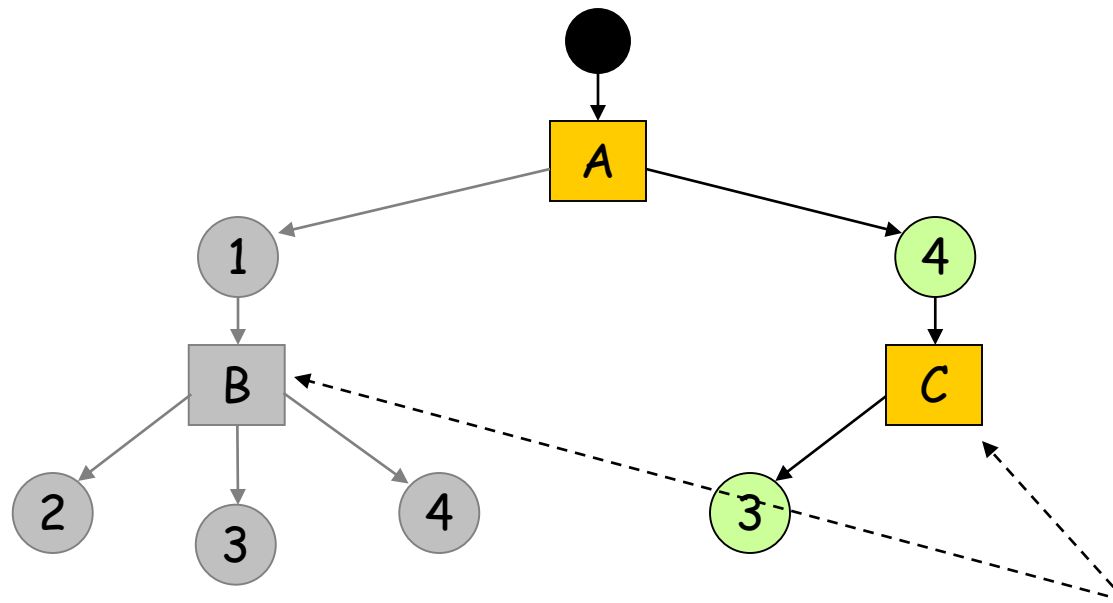
- Ej.



Asignación =  $\{(A,4)\}$

## Backtracking search (2)

- Ej.

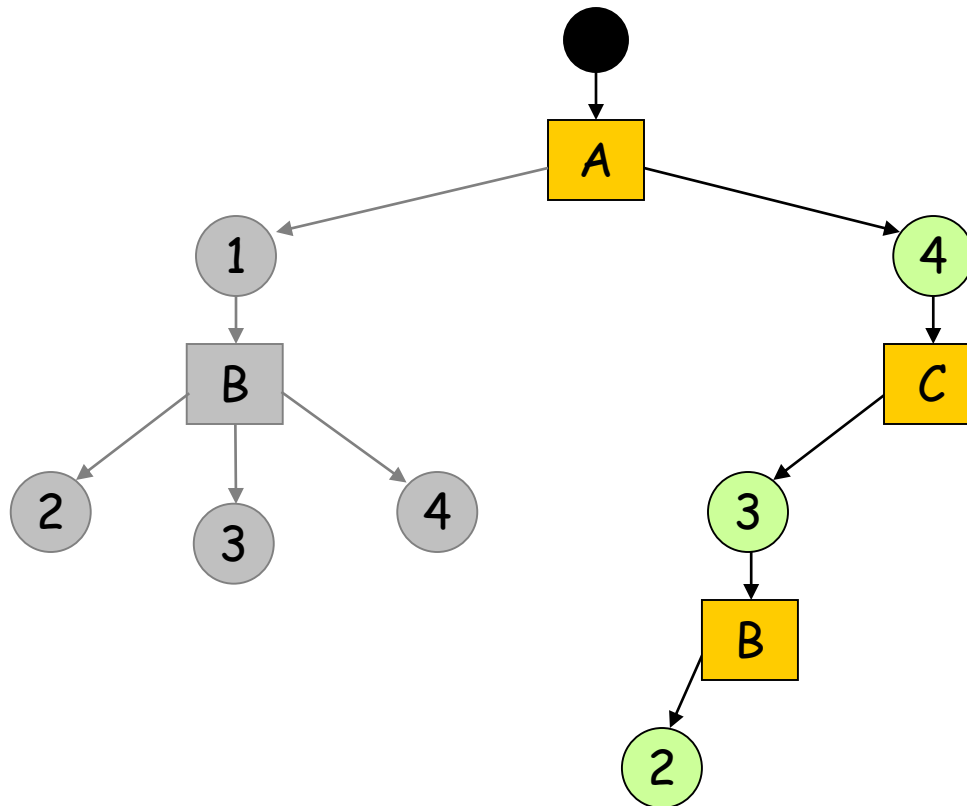


El algoritmo no tiene que considerar las variables en el mismo orden

Asignación =  $\{(A,4),(C,3)\}$

## Backtracking search (2)

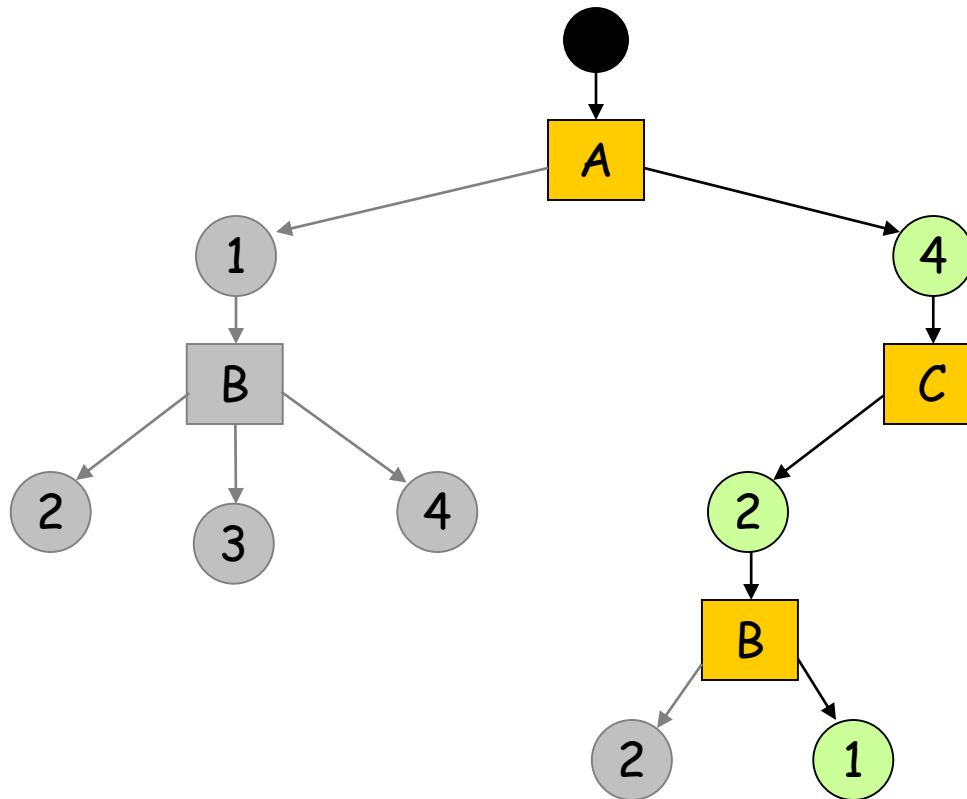
- Ej.



Asignación =  $\{(A,4), (C,3), (B,2)\}$

## Backtracking search (2)

- Ej.



Asignación =  $\{(A,4), (C,3), (B,1)\}$

## Backtracking algorithm – Pseudocódigo

CSP-BACKTRACKING(**A**)

1. If assignment **A** is complete then return **A**
2. **X**  $\leftarrow$  select a variable not in **A**
3. **D**  $\leftarrow$  select an ordering on the domain of **X**
4. For each value **v** in **D** do
  - a. Add (**X** $\leftarrow$ **v**) to **A**
  - b. If **A** is valid then
    - i. **result**  $\leftarrow$  CSP-BACKTRACKING(**A**)
    - ii. If **result**  $\neq$  failure then return **result**
5. Return failure

## Backtracking – Eficiencia (1)

- ¿A qué variable X se le debería de asignar un valor en el siguiente paso?
  - La **siguiente asignación** puede llevar a **ninguna solución**, pero el algoritmo aún no lo sabe.
  - Seleccionar la **variable correcta** a la que asignar un valor puede ayudar a descubrir la contradicción más rápidamente.
- ¿En qué orden deberían asignarse los valores de X?
  - La asignación puede ser parte de una solución. Seleccionar el valor correcto puede ayudar a descubrir esta solución de forma más rápida.

## Backtracking – Eficiencia (2)

### CSP-BACKTRACKING(*A*)

1. If assignment *A* is complete then return *A*
2. *X* ← **select** a variable not in *A*
3. *D* ← **select** an ordering on the domain of *X*
4. For each value *v* in *D* do
  - a. Add (*X*←*v*) to *A*
  - b. If *A* is valid then
    - i. *result* ← CSP-BACKTRACKING(*A*)
    - ii. If *result* ≠ failure then return *result*
5. Return failure

## Propagación de restricciones (1)

- El algoritmo de búsqueda solo considera las restricciones de una variable cuando esta variable es seleccionada para darle un valor.
- Si se pudiera mirar a las restricciones antes, seríamos capaces de reducir el espacio de búsqueda de forma drástica.
- ¿Cómo?
  - Por ejemplo: forward checking (comprobación hacia delante).

## Propagación de restricciones (2)

*Comunicar la reducción del dominio de una variable de decisión a todas las restricciones que se establecen sobre dicha variable.*

[[Fuente](#)]

## Propagación de restricciones – Pseudocódigo

When  
    “un par( $X \leftarrow v$ ) es añadido a una asignación  $A$ ”  
do:

For each variable  $Y$  not in  $A$  do:

    For every constraint  $C$  relating  $Y$  to  $X$  do:

        Remove all values from  $Y$ 's domain that do  
        not satisfy  $C$

## Backtracking algorithm mejorado – Pseudocódigo

CSP-BACKTRACKING(A, var-domains)

1. If assignment A is complete then return A
2.  $X \leftarrow$  select a variable not in A
3.  $D \leftarrow$  select an ordering on the domain of X
4. For each value v in D do
  - a. Add ( $X \leftarrow v$ ) to A
  - b.  $\text{var-domains} \leftarrow$  forward checking(var-domains, X, v, A)
  - c. If A is valid then
    - i.  $\text{result} \leftarrow$  CSP-BACKTRACKING(A, var-domains)
    - ii. If result  $\neq$  failure then return result
5. Return failure

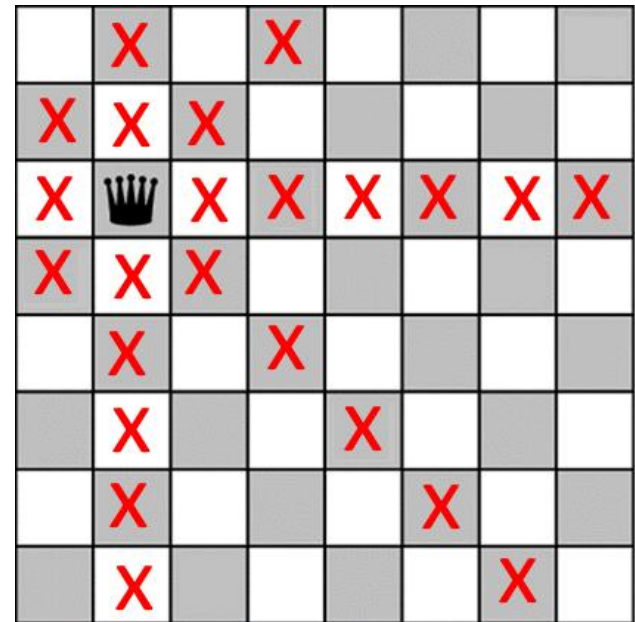
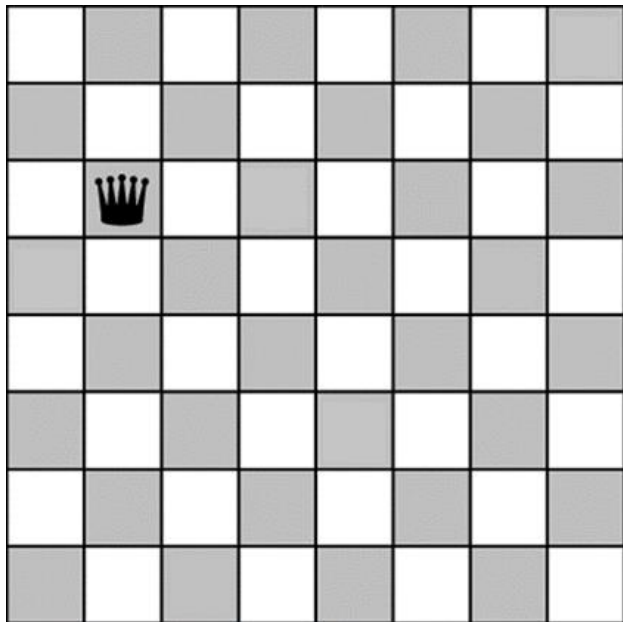
## Observación reflexiva #3

### 8-Reinas: Formulación CSP

- 64 variables  $X_{ij}$ , siendo  $i = 1$  hasta 8;  $j = 1$  hasta 8.
- El dominio de cada variable es  $\{1,0\}$
- Las limitaciones se pueden representar como:
  - Si  $X_{ij} = 1$  entonces  $X_{ik} = 0$  para todo  $k = 1$  to 8,  $k \neq j$
  - Si  $X_{ij} = 1$  entonces  $X_{kj} = 0$  para todo  $k = 1$  to 8,  $k \neq i$
  - ¿Diagonales?
    - Si  $X_{ij} = 1$ , entonces  $X_{k,l} = 0$ , para todo  $k, l$ , donde  $k \neq i$  y  $l \neq j$ , y donde  $|k - i| = |l - j|$
    - Si  $X_{ij} = 1$ , entonces  $X_{k,l} = 0$ , para todo  $k, l$ , donde  $k \neq i$  y  $l \neq j$ , y donde  $|k - i| = |l - j|$
    - Si  $X_{ij} = 1$ , entonces  $X_{k,l} = 0$ , para todo  $k, l$ , donde  $k \neq i$  y  $l \neq j$ , y donde  $|k - i| = |l - j|$
    - Si  $X_{ij} = 1$ , entonces  $X_{k,l} = 0$ , para todo  $k, l$ , donde  $k \neq i$  y  $l \neq j$ , y donde  $|k - i| = |l - j|$

## Observación reflexiva #4

### Forward checking



## Forward checking - Pseudocódigo

When

“un par  $(X \leftarrow v)$  es añadido a la asignación  $A$ ”  
do:

For each variable  $Y$  not in  $A$  do:

For every constraint  $C$  relating  $Y$  to  
variables in  $A$  do:

Remove all values from  $Y$ 's domain  
that do not satisfy  $C$

## CSP - Eficiencia (3)

1. ¿A qué variable X se le debería de asignar un valor en el siguiente paso?
  - a. **Most-constrained-variable heuristic (Heurística de variable más restringida)**
  - b. **Most-constraining-variable heuristic (Heurística de la variable más restrictiva)**

Si vas a fallar, hazlo lo antes posible.

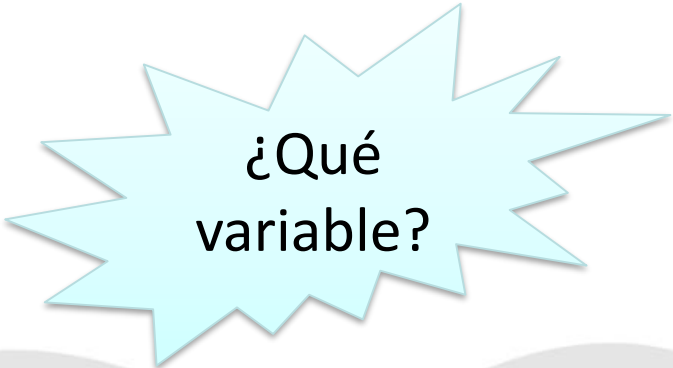
2. ¿En qué orden deberían asignarse los valores de X?
  - **Least-constraining-value heuristic (Heurística de valor menos restrictivo)**

Intenta conseguir la mayor probabilidad de éxito

Todas las variables deben tomar un valor (y solo un valor del dominio)  
eventualmente.

### 1- a) Most-constrained-variable heuristic (Heurística de variable más restringida)

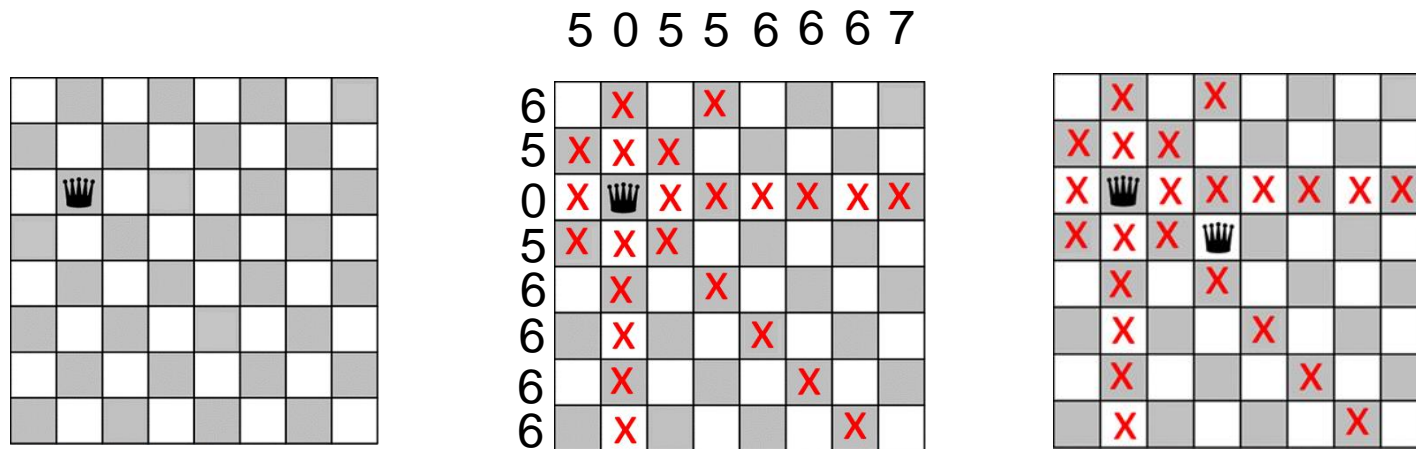
- Seleccionar la **variable** con el **dominio más pequeño**.
- Se minimiza el factor de ramificación.



¿Qué  
variable?

## Observación reflexiva #4

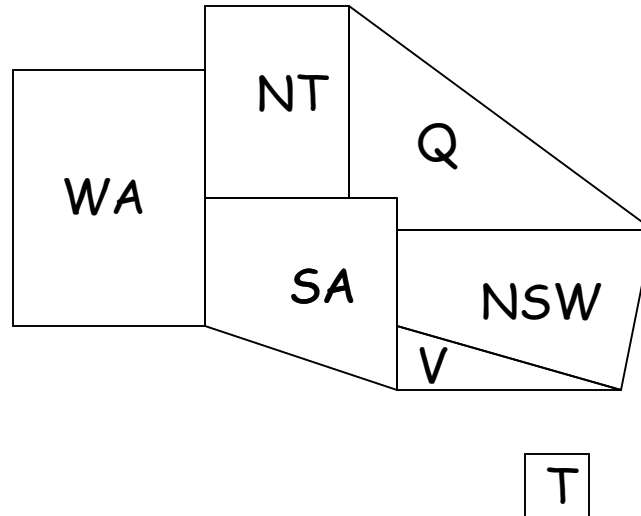
¿Cómo se aplica al caso de 8-Reinas?



- Contar el número de casillas no atacadas en cada fila y columna.
- Poner una reina en un fila o columna con nº de casillas no atacadas más pequeño.
- Eliminar las celdas susceptibles de ser atacadas como en el caso anterior.
- Repetir.

## Observación reflexiva #5

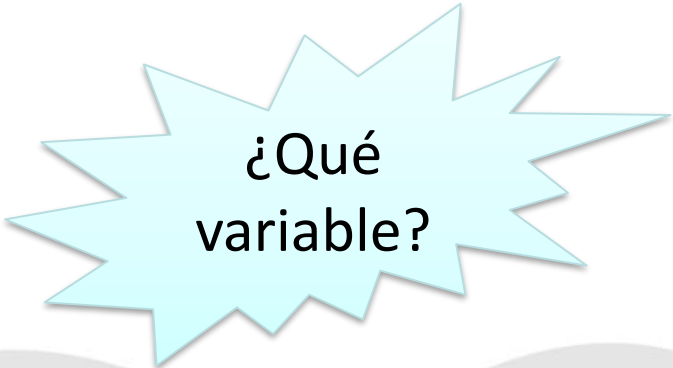
En el caso del mapa:



- Todas las variables tienen el mismo dominio.

### 1 b) Most-constraining-variable heuristic (heurística de la variable más restrictiva)

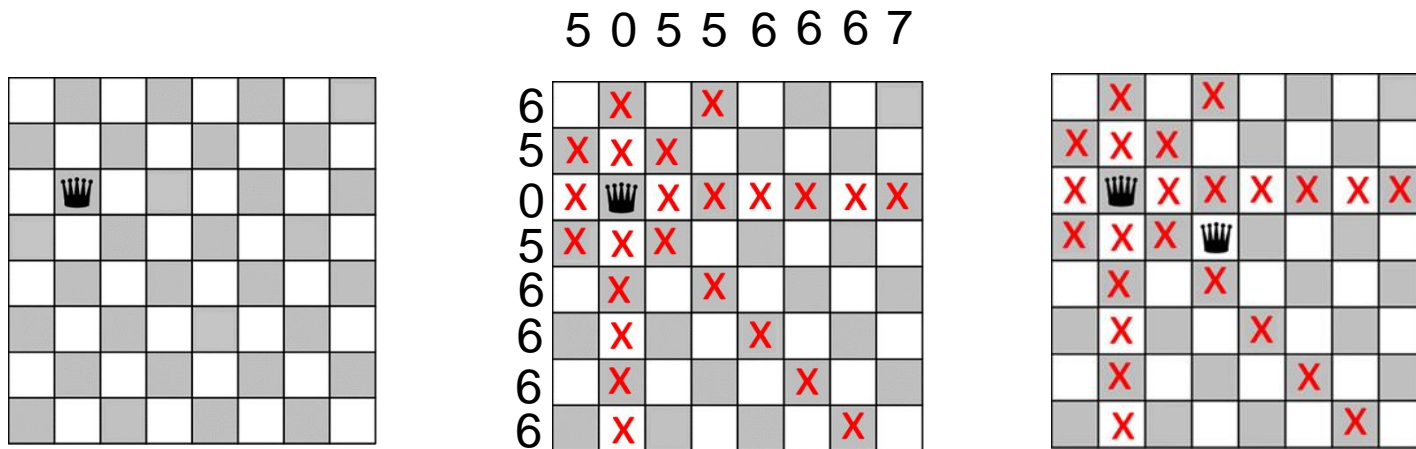
- Dentro de las variables con menor dominio restante (**Most-constrained-variable heuristic**), selecciona la que aparece **con el mayor número de restricciones** de las variables que no se encuentran en la asignación actual.
- Se incrementa la futura eliminación de valores para reducir el factor de ramificación.



¿Qué  
variable?

## Observación reflexiva #6

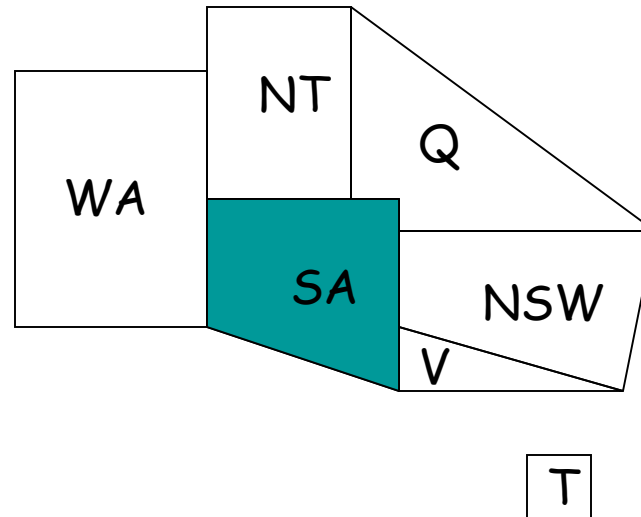
¿Cómo se aplica al caso de 8-Reinas?



- Todas aparecen en el mismo número de restricciones

## Observación reflexiva #7

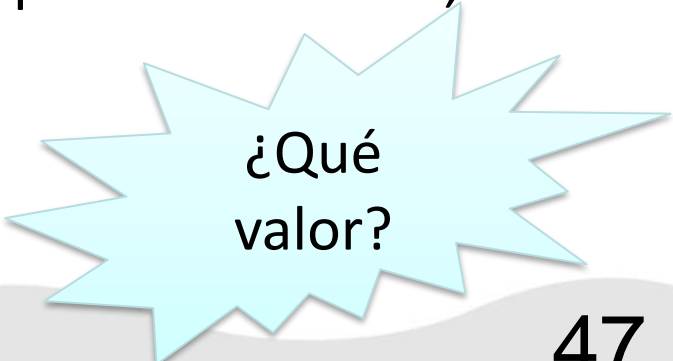
En el caso del mapa:



- Antes de que un valor sea asignado, todas las variables tiene un dominio de tamaño 3, pero SA tiene más restricciones porque está al lado de más territorios.
- Le asignamos un valor, por ejemplo azul.

## 2 Least-Constraining-Value Heuristic (heurística de valor menos restrictivo)

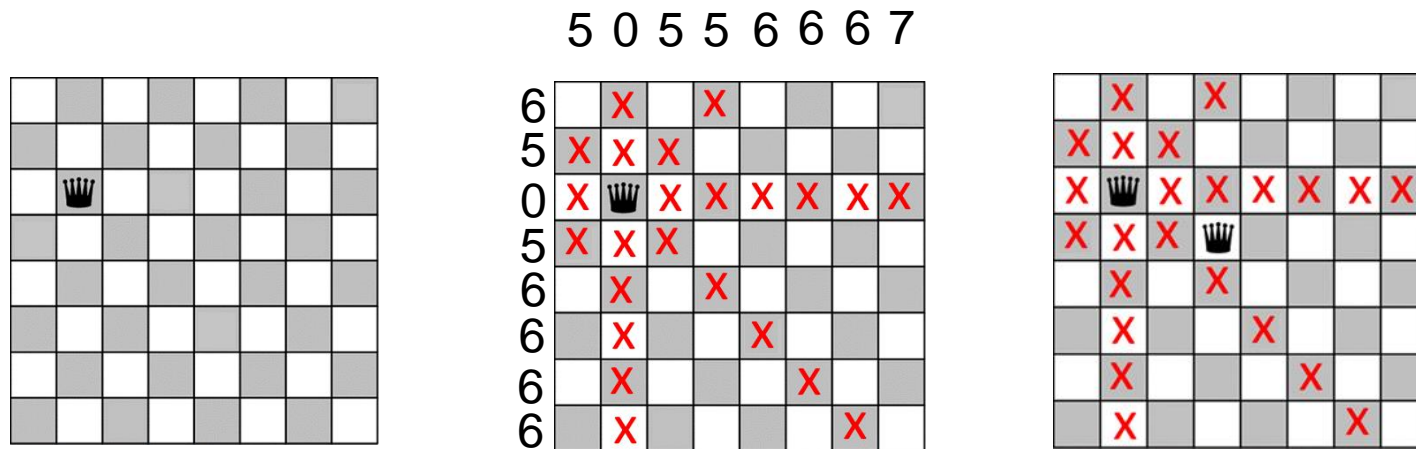
- **Selecciona el valor que elimina el menor número de valores del dominio** de aquellas variables que no están en la asignación actual.
- Ya que solo un valor será asignado a cada variable, escoge el menos restrictivo primero, ya que es el que menos probable nos lleve a una asignación equivocada.
  - Seleccionando esta heurística es necesario **comprobar hacia delante** (forward checking) para cada valor, no solo para el valor seleccionado.



¿Qué  
valor?

## Observación reflexiva #8

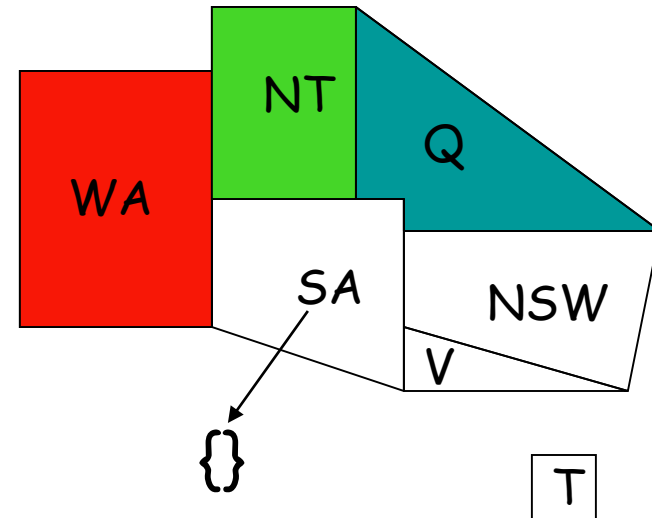
¿Cómo se aplica al caso de 8-Reinas?



- Cualquier valor elimina el mismo número de restricciones.

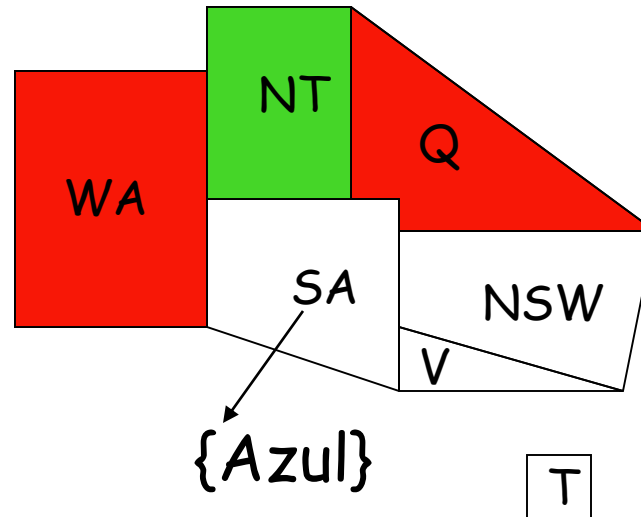
## Observación reflexiva #9 (1)

En el caso del mapa en esta situación.



- El dominio de Q tiene dos valores restantes (Rojo y Azul)
- Si le asignamos el valor azul a Q, entonces SA no tendría ningún valor en su dominio (no puede ser rojo ni verde ni azul).
- Si le asignamos rojo le quedaría uno (Azul).

## Observación reflexiva #9 (2)



- Le asignamos el valor rojo.

## Forward checking - Eficiencia

- La comprobación hacia delante incrementa la complejidad temporal.
  - Una buena técnica es combinar backtracking y forward checking.
  - Intercambio entre propagación hacia delante y atrás y propagación de restricciones.

## Resumen CSP - ¿Qué necesitamos? (1)

- No solo necesitamos comprobar el sucesor y el objetivo:
  - Propagar las restricciones.
  - Comprobar errores.
- Para ello:
  - Representación explícita de las restricciones.
  - Algoritmos de propagación de restricciones.

## Resumen CSP - ¿Qué necesitamos? (2)

- Tener en cuenta:
  1. ¿A que variable se le debería asignar un valor en cada paso?
  2. ¿En qué orden asignamos los valores?
- Para ello:
  - 1 a) Most-constrained-variable heuristic (Heurística de variable más restringida)
  - 1 b) Most-constraining-variable heuristic (Heurística de la variable más restrictiva)
  - 2 Least-constraining-value heuristic (Heurística de valor menos restrictivo)

## Aplicaciones de CSP

- Asignación de tripulaciones a vuelo.
- Flota de transporte.
- Horario de trenes y vuelos.
- Horario de tiendas.
- Operaciones en puertos.
- Diseño.
- Procedimientos radio quirúrgicos.

## Bibliografía

Esta presentación se basa principalmente en información recogida en las siguientes fuentes:

- McCoy, K. F. (2010). <https://www.eecis.udel.edu/~mccoy/>
- Russell, S. & Norvig, P. (2010). *Artificial Intelligence: A modern approach*. 3ª Ed. Prentice-Hall.

## Lecturas

- Recomendadas:
  - Russell, S. & Norvig, P. (2010). *Artificial Intelligence: A modern approach*. 3ª Ed. Prentice-Hall.
    - Chapter 6 – Constraint Satisfaction Problems.