Inteligencia Artificial
Tema 6. Problemas de Satisfacción de Restricciones

José Manuel Fuertes García
jmf@ujaen.es

Departamento de Informática
Universidad de Jaén

27 de abril de 2017

Objetivos

- Entender cómo se pueden plantear determinados problemas considerando más información.
- Conocer el significado de los problemas de satisfacción de restricciones.
- Conocer los métodos de resolución de problemas de satisfacción de restricciones y su aplicación.

Índice

Introducción

Un problema de satisfacción de restricciones (PSR) está definido por:

- Un conjunto de variables $X_1, X_2, \ldots, X_n$.
- Un conjunto de restricciones $C_1, C_2, \ldots, C_m$.
- Cada variable X_i tiene un dominio no vacío D_i de valores posibles.
- Cada restricción implica un subconjunto de variables y especifica las combinaciones de valores aceptables para el mismo.

- Estado del problema: asignación de valores a una o todas las variables.
- Asignación consistente: no viola ninguna restricción.
- Solución a un PSR: asignación completa (que asigna valor a todas las variables) y consistente.

Algunos PSR requieren una solución que optimice una función objetivo.

Introducción

Satisfacción de restricciones *versus* búsqueda estándar

- **Búsqueda estándar:**
  - El estado es una *caja negra*: Cualquier estructura de datos que apoye la función sucesor, la función heurística y el test objetivo.
- **Satisfacción de restricciones:**
  - El estado se define mediante variables X_i con valores del dominio D_i.
  - El test objetivo es un conjunto de restricciones que especifican las combinaciones de valores permitidos para los subconjuntos de variables.
  - Un estado del problema está definido por una asignación de valores (total o parcial) a las variables $X_i = v_i, X_j = v_j, ...$

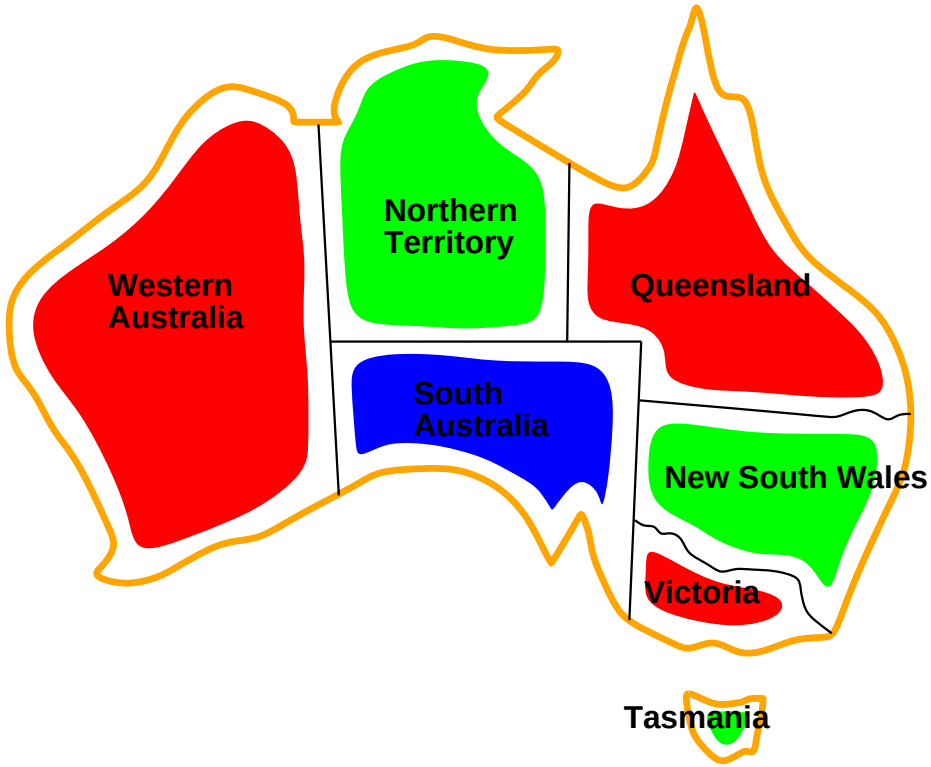
Pueden emplearse algoritmos que resultan más eficientes que los de búsqueda estándar.

Ejemplo de problema de satisfacción de restricciones

- Coloreado de un mapa:

- Variables: `AO, TN, Q, NGS, V, AS, T`.
- Dominios $D_i = \{\text{rojo, verde, azul}\}$.
- Restricciones: Regiones adyacentes deben tener colores diferentes.

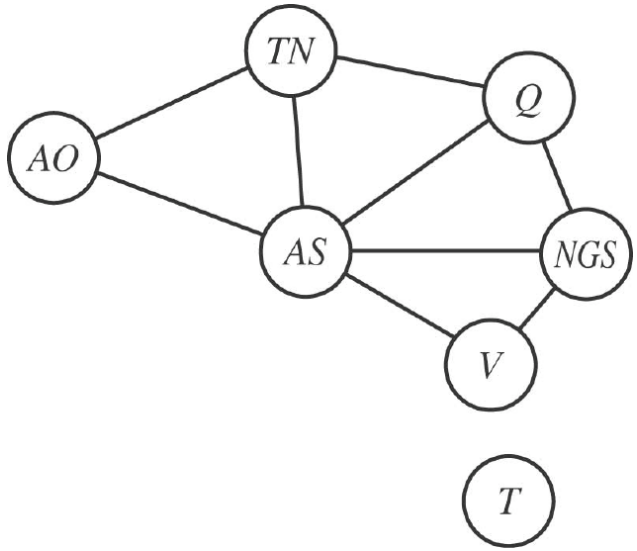
Ejemplo de problema de satisfacción de restricciones



Ejemplo de problema de satisfacción de restricciones

Grafo de restricciones

- Podemos visualizar el problema como un grafo de restricciones.
- En el caso del mapa, el grafo es binario (cada arco une dos nodos).
- Puede haber restricciones unarias, binarias, ternarias, etc.



Ventajas de los PSR

Ventajas:

- El modelo es estándar: la función sucesor y el test objetivo pueden escribirse de forma genérica.
- Se pueden desarrollar heurísticas eficaces y genéricas independientes del problema.
- La estructura del grafo de restricciones puede usarse para simplificar el proceso de solución.

Formulación incremental de los PSR

Formulación incremental:

- Estado inicial: Asignación vacía.
- Función sucesor: Instanciar una variable no asignada (sin provocar conflictos).
- Test objetivo: Tener instanciadas todas las variables.
- Costo: Constante por cada paso.

- La solución aparecerá a profundidad n. Podemos usar algoritmos primero en profundidad.
- Se puede utilizar la formulación completa de estados. Con ella podemos emplear métodos de búsqueda local.

Problemas de satisfacción de restricciones

Ejemplos de problemas reales:

- Problemas de asignación: Quién imparte qué clase.
- Problemas de horarios: Qué clase se ofrece, dónde y cuándo.
- Problemas de configuración de hardware.
- Problemas de organización de transporte.
- Problemas de organización de factorías.
- ...

Algunos problemas reales involucran variables con valores reales.

Índice

Tipos de problemas de satisfacción de restricciones

Con variables discretas:

- De dominios finitos.
  - n variables, tamaño del dominio $d \rightarrow O(d^n)$ asignaciones completas.
  - Ej. PSRs booleanos, que incluyen como casos especiales algunos problemas NP-completos.
- De dominios infinitos.
  - Números enteros, cadenas, etc.
  - Ej., en la construcción de un calendario, la fecha de comienzo de cada trabajo es una variable y los valores posibles son números enteros de días desde la fecha actual.
  - Necesidad de un lenguaje de restricción, p.e., `ComienzoTrabajo1 + 5` $\leq$ `ComienzoTrabajo3`.

Tipos de problemas de satisfacción de restricciones

Con variables continuas:

- Ej., tiempos de comienzo/fin para las observaciones del Telescopio Hubble.
- La categoría más conocida de PSRs en dominios continuos son los problemas de programación lineal, en los que las restricciones deben ser desigualdades lineales que forman una región convexa.
  - Se pueden resolver en tiempo polinomial.

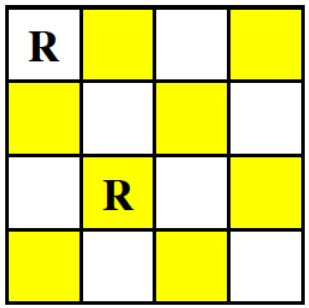
Tipos de problemas de satisfacción de restricciones

Tipos de restricciones

- Unarias: Involucran a una sola variable.
  Ejemplo: $\texttt{AS} \neq \textit{verde}$
- Binarias: Implican a un par de variables.
  Ejemplo: $\texttt{AS} \neq \texttt{AO}$
- De alto-orden: Afectan a tres o más variables.
  Ejemplo: restricciones de las columnas criptoaritméticas.
- Preferencias (o restricciones débiles): Frecuentemente se representan a través de un costo en la asignación.
  Ejemplo: para una variable, *rojo* puede ser mejor que *verde*.
  - Problemas de optimización restringida.

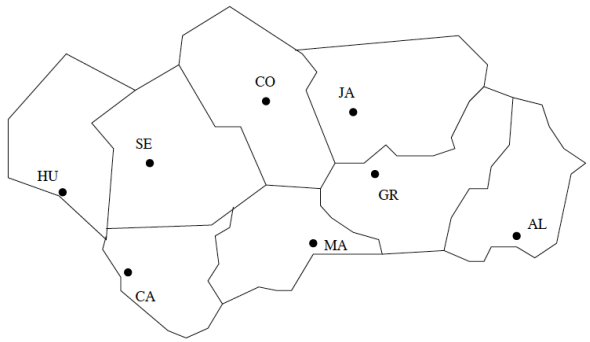
Ejemplo de problema de satisfacción de restricciones

- N-Reinas:
  - Situar **N** reinas en un tablero de ajedrez de tamaño $N \times N$ de forma que no se den jaque mutuamente.
  - Variables: $V_1, ..., V_N$
  - Dominios: $D_i = \{1, ..., N\}$
  - Restricciones:
    - Jaque horizontal: $V_i \neq V_j$
    - Jaque diagonal: $|V_i - V_j| \neq |i - j|$
  - Ejemplo: $(V_1 = 1, V_2 = 3)$:



Ejemplo de problema de satisfacción de restricciones

- Coloreado de mapas:
  - Usando tres colores (rojo,azul,verde) colorear el mapa de Andalucía:



  - Variables: Huelva, Cádiz, Sevilla, Córdoba, Málaga, Jaen, Granada, Almería.
  - Dominios: $\{rojo, azul, verde\}$
  - Restricciones:
    - $P \neq Q$, para cada par de provincias vecinas P y Q.

Ejemplo de problema de satisfacción de restricciones

- Satisfacibilidad proposicional:
  - Dado un conjunto de variables proposicionales $L = \{p_1, ..., p_n\}$ y un conjunto de cláusulas proposicionales $\{C_1, ..., C_m\}$ formadas con los símbolos de L, determinar si existe una asignación de valores de verdad a los símbolos de L de manera que se satisfagan todas las cláusulas.
  - Variables: $\{p_1, ..., p_n\}$.
  - Dominios: $\{T, F\}$ (*booleanos*)
  - Restricciones: $\{C_1, ..., C_m\}$

Ejemplo de problema de satisfacción de restricciones

- Problema criptoaritmético 1:

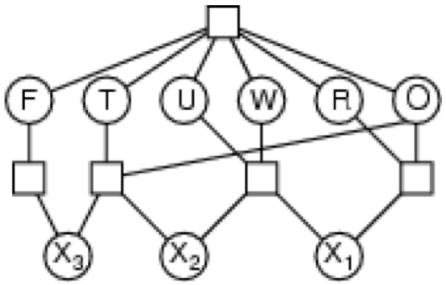
$$\begin{array}{r} \text{(C1 C2 C3)} \\ \text{I D E A} \\ \text{I D E A} \\ \hline \text{M E N T E} \end{array} \quad +$$

- Variables: $I, D, E, A, M, N, T, C_1, C_2, C_3$
- Dominios: $\{1, ..., 8\}$ para I, D, E, A, M, N, T y $\{0, 1\}$ para C_1, C_2, C_3.
- Restricciones:
  - Primera suma: $2 \cdot A = (10 \cdot C_3) + E$
  - Segunda suma: $(2 \cdot E) + C_3 = (10 \cdot C_2) + T$
  - Tercera suma: $(2 \cdot D) + C_2 = (10 \cdot C_1) + N$
  - Cuarta suma: $(2 \cdot I) + C_1 = (10 \cdot M) + E$

Ejemplo de problema de satisfacción de restricciones

- Problema criptoaritmético 2:

$$\begin{array}{r} \,T\;W\;O \\ +\;T\;W\;O \\ \hline F\;O\;U\;R \end{array}$$



- Variables: $F, T, U, W, R, O, X_1, X_2, X_3$.
- Dominios $D_i = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
- Restricciones:
  - $F \neq T \neq W \neq U \neq O \neq R$
  - $O + O = R + 10 \cdot X_1$
  - $X_1 + W + W = U + 10 \cdot X_2$
  - $X_2 + T + T = O + 10 \cdot X_3$
  - $X_3 = F$; $T \neq O$; $F \neq O$.

Ejemplo de problema de satisfacción de restricciones

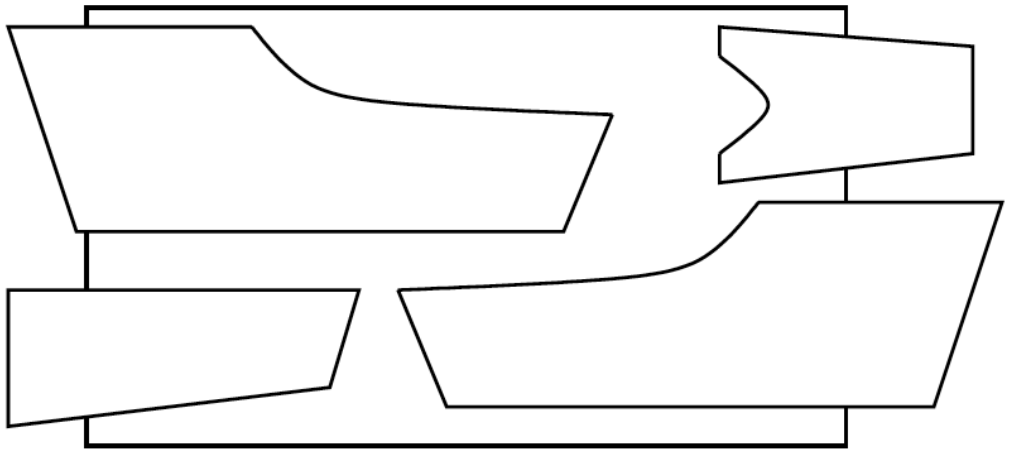
- Asignación de tareas:
  - Asignar tareas a empleados de acuerdo con su capacidad para desarrollarlas.
  - Tablas de capacidades (C_i):

	T_1	T_2	T_3
E_1	1	3	2
E_2	3	2	1
E_3	2	3	1

  - Variables: E_i
  - Dominios: $Di = \{T_1, ..., T_n\}$.
  - Restricciones:
    - Obtener la mejor $\sum_{i=1}^{n} C_i$

Ejemplo de problema de satisfacción de restricciones

- Planificación de corte:
  - Encontrar la manera de situar patrones de corte en una pieza de cartón.
  - Variables: P_1, P_2, P_3, P_4 (piezas)
  - Dominios: Coordenadas en el plano.
  - Restricciones:
    - Las piezas no deben superponerse.



Índice

Técnicas de resolución de los PSR

- Técnicas de consistencia, basadas en la eliminación de valores inconsistentes.
  - Ejemplo: Problema del carcelero y los tres sombreros.
- Algoritmos de búsqueda, basados en la exploración sistemática del espacio de posibles soluciones hasta encontrar una (o probar que no hay ninguna).
  - Cada nodo tendrá una variable asignada más que su antecesor.
- Optimización: permitiendo estados con restricciones no satisfechas.

Algoritmo simple de vuelta atrás para problemas de satisfacción de restricciones

- El término *búsqueda con vuelta atrás* se utiliza para la búsqueda en profundidad que:
  - elige valores para una variable a la vez y
  - vuelve atrás cuando una variable no tiene ningún valor legal para asignarle.
- El algoritmo:
  - genera los sucesores incrementalmente, uno a la vez.
  - extiende la asignación actual para generar un sucesor, más que volver a copiarlo.

Algoritmo simple de vuelta atrás para problemas de satisfacción de restricciones

función Búsqueda-Con-Vuelta-Atrás (*psr*) **devuelve** solución o fallo

devolver Vuelta-Atrás-Recursiva ({}, *psr*)

función Vuelta-Atrás-Recursiva (*asignación*, *psr*) **devuelve** una solución o fallo

si *asignación* es completa **entonces devolver** *asignación*

var $\leftarrow$ Selec.-Variable-NoAsignada(Variables[*psr*], *asignación*, *psr*)

para cada *valor* en Orden-Valores(*var*, *asignación*, *psr*) **hacer**

si *valor* es consistente con *asignación* de acuerdo a las Restricciones[*psr*] **hacer**

añadir $\{var = valor\}$ a *asignación*

resultado $\leftarrow$ Vuelta-Atrás-Recursiva(*asignación*, *psr*)

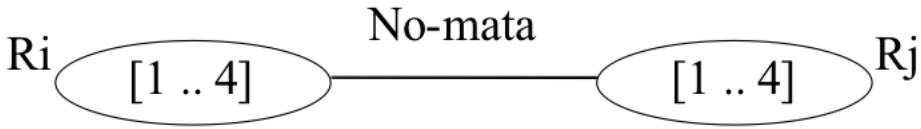
si *resultado* $<>$ *fallo* **entonces devolver** *resultado*

borrar $\{var = valor\}$ de *asignación*

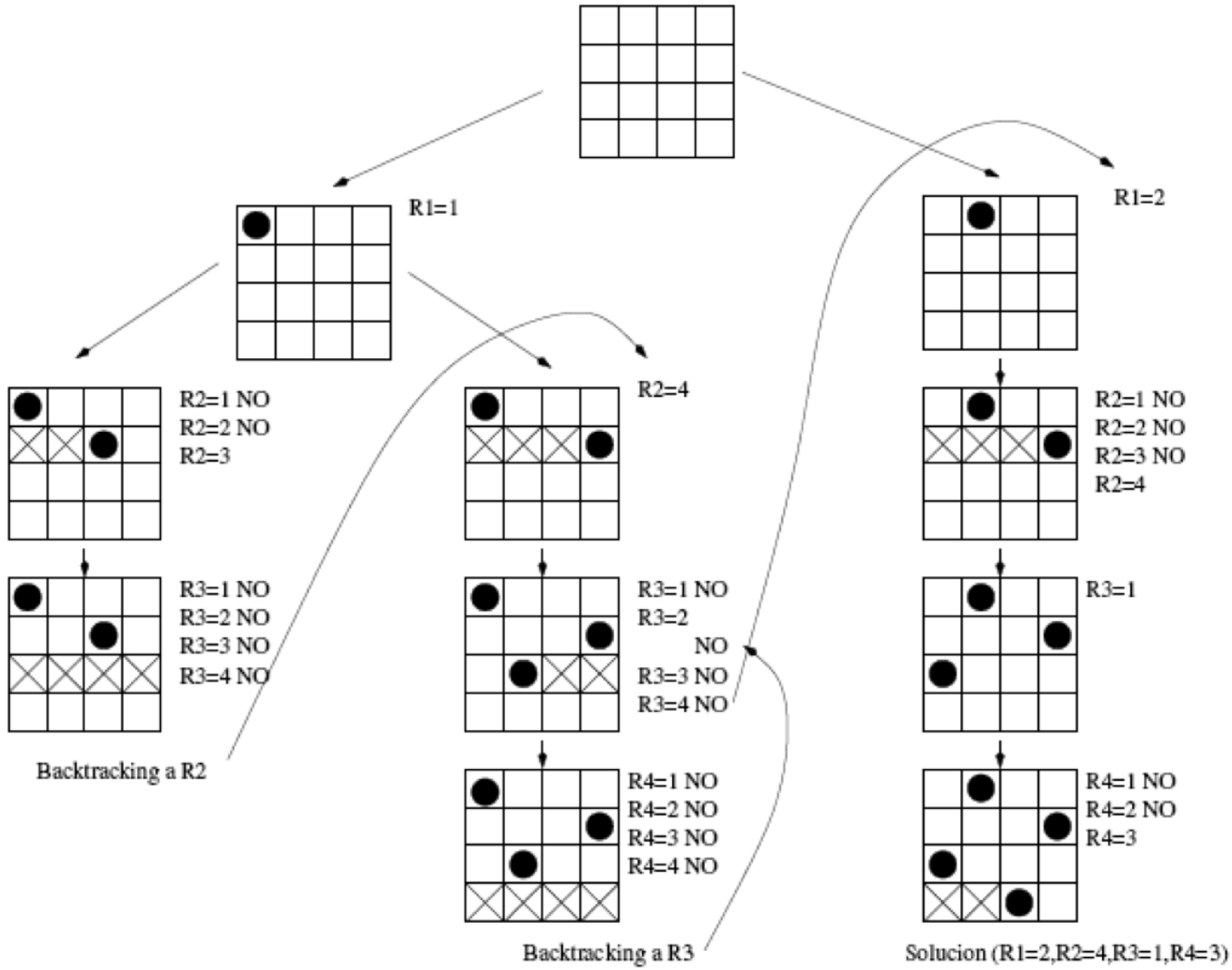
devolver *fallo*

4-reinas con usando un algoritmo de búsqueda con vuelta atrás cronológica

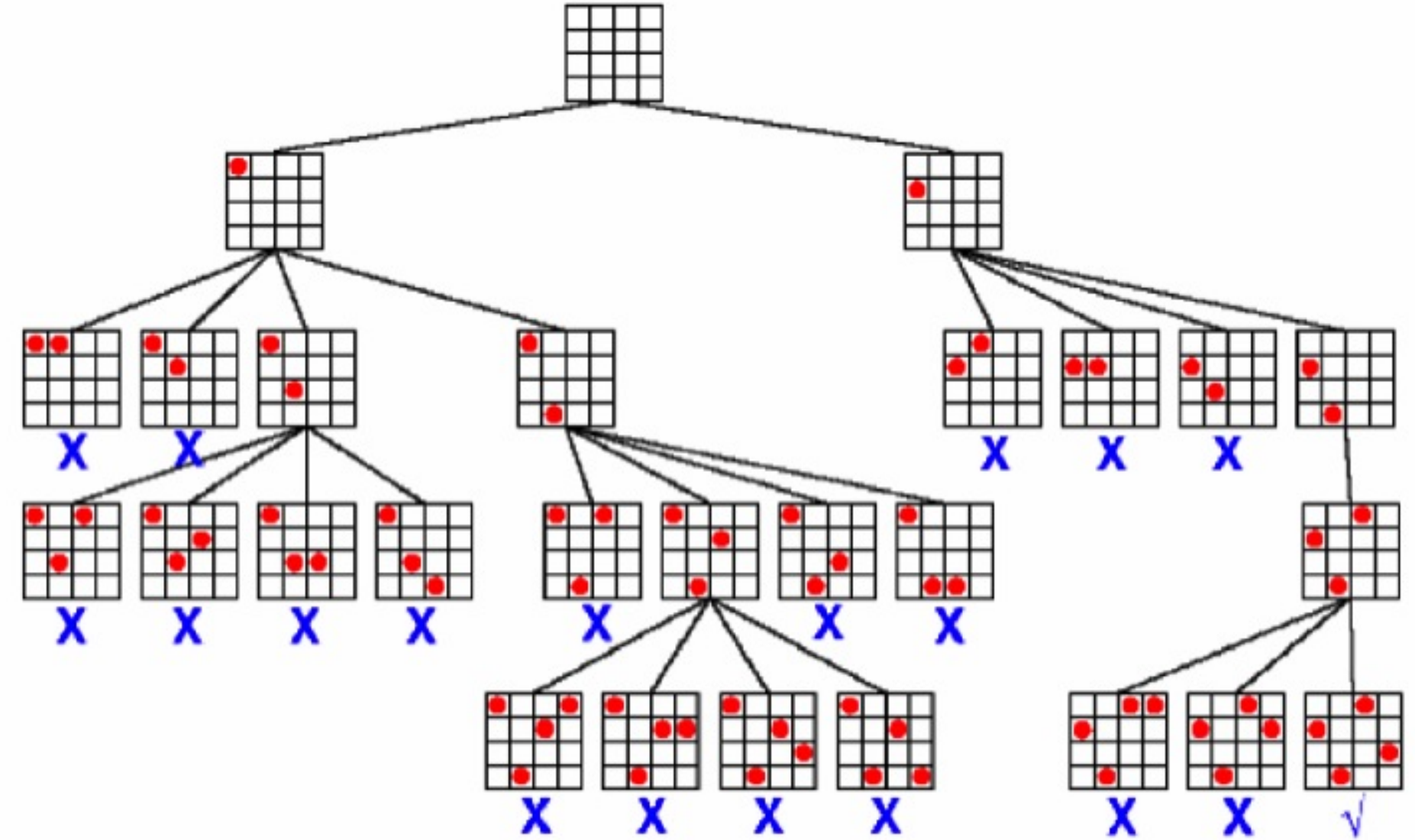
- Colocar 4 reinas, una en cada fila de un tablero $4x4$, sin que se maten.
  - Variables: $R_1, ..., R_4$ (reinas)
  - Dominios: $\{1, ..., 4\}$ para cada R_i (columna)
  - Restricciones: R_i no-mata R_j
- Grafo:



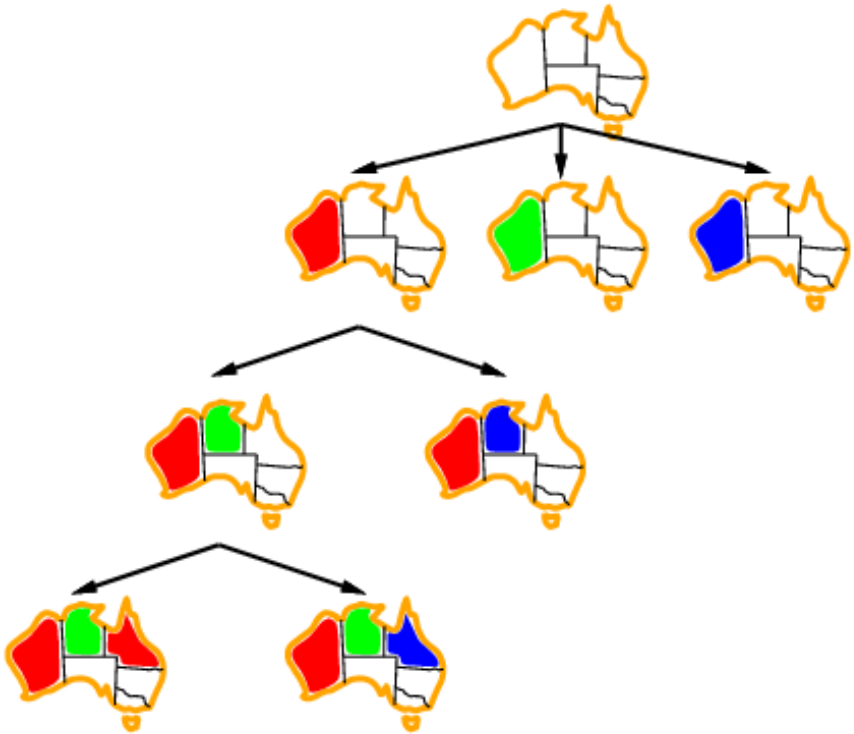
4-reinas con usando un algoritmo de búsqueda con vuelta atrás cronológica



4-reinas con usando un algoritmo de búsqueda con vuelta atrás cronológica



Relleno del mapa usando un algoritmo de búsqueda con vuelta atrás cronológica



Mejoras en la eficiencia

- En principio, la vuelta atrás realiza una búsqueda ciega
  - Búsqueda no informada, ineficiente en la práctica.
- Es posible dotar al algoritmo de cierta heurística que mejora considerablemente su rendimiento.
  - Estas heurísticas son de propósito general.
  - Independientes del problema.
  - Sacan partido de la estructura especial de los PSR.

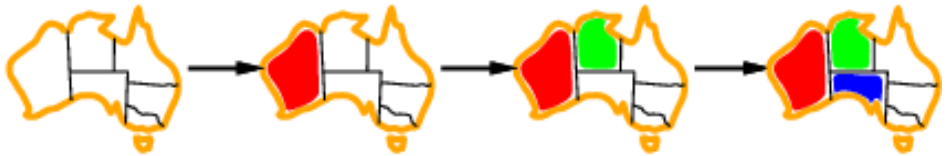
Mejoras en la eficiencia

- Posibles mejoras heurísticas:
  - ¿Qué variable se debería asignar en siguiente lugar?
  - ¿En qué orden se deberían probar sus valores?
  - ¿Cuáles son las implicaciones de las variables actuales para las otras variables no asignadas?
  - Cuando un camino falla, ¿puede la búsqueda evitar repetir este fracaso en caminos siguientes?

Variables y ordenamiento de valor

Heurística de Mínimos Valores Restantes (MVR):

- Escoger la variable con menos valores legales.

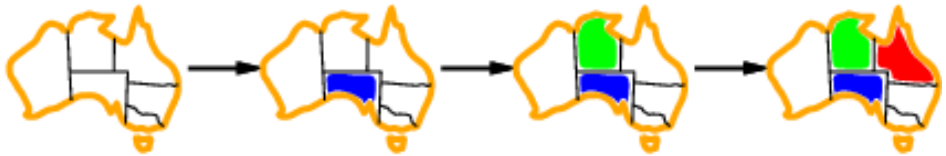


- También llamada heurística de mínimos valores restantes (MVR) o de *primero en fallar*, porque escoge una variable que con mayor probabilidad causará pronto un fracaso, lo que poda el árbol de búsqueda.

Variables y ordenamiento de valor

Grado heurístico

- Escoger la variable con más restricciones en las variables restantes.



- La heurística MVR es en general una guía más poderosa, pero el grado heurístico puede ser útil como desempate.

Variables y ordenamiento de valor

El valor menos restrictivo

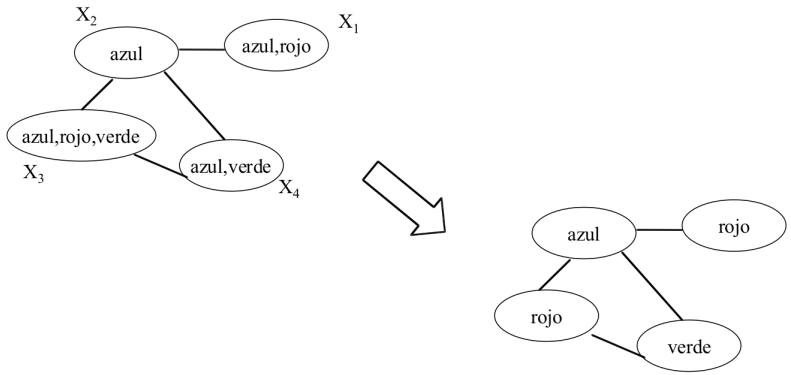
- Una vez seleccionada una variable, el algoritmo debe decidir el orden para examinar sus valores.
- Dada una variable, escoger el valor menos restrictivo (el que elimina menos valores posibles en los nodos vecinos del grafo de restricciones):

  - Ejemplo: Si AO=rojo y TN=Verde, Q=Azul es una mala opción, porque deja *bloqueado* a AS.

- Si debemos encontrar todas las soluciones a un problema, o no hay soluciones para el mismo, el orden no importa.

Propagación de la información a través de las restricciones

- Un conjunto de restricciones puede inducir otras (que estaban implícitas).
- La propagación de restricciones (PR) es el proceso de hacerlas explícitas.



- El papel de la Propagación de Restricciones es disminuir el espacio de búsqueda. Pudiéndose realizar:
  - Como pre-proceso: eliminar zonas del espacio donde no hay soluciones (arc consistency).
  - Durante el proceso: podar el espacio a medida que la búsqueda progresa (forward checking).

Propagación de la información a través de las restricciones

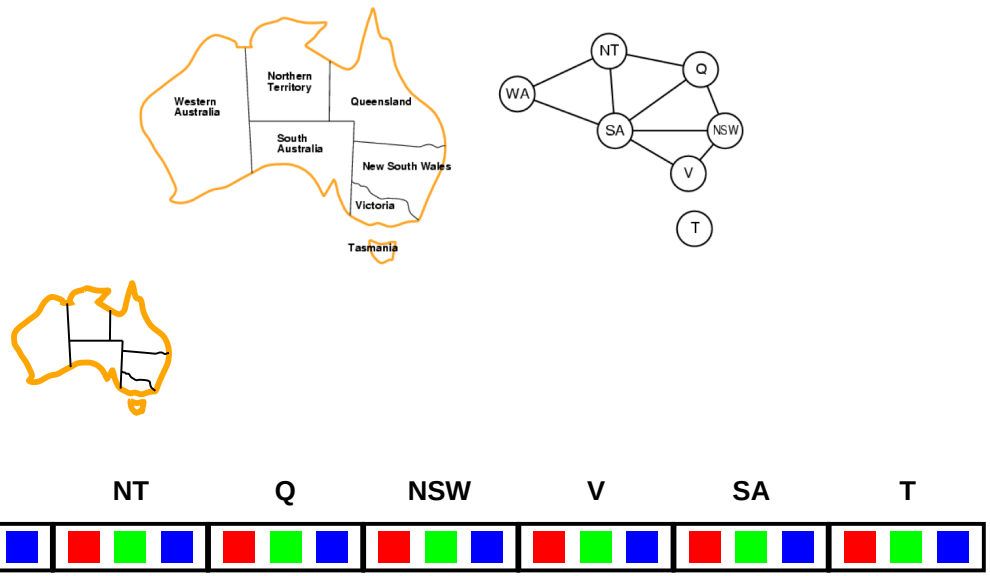
Comprobación hacia adelante

- Modificación del algoritmo de búsqueda en profundidad con vuelta atrás cronológica.
- Detecta cuanto antes caminos sin solución y lo poda.
  - Asignar un valor y consultar las restricciones sobre las variables futuras con arco desde la actual.
  - Se eliminan valores no compatibles de los dominios correspondientes a dichas variables futuras.
- Equivale a hacer arco-consistente la variable actual con las futuras en cada paso.

Propagación de la información a través de las restricciones

Comprobación hacia adelante

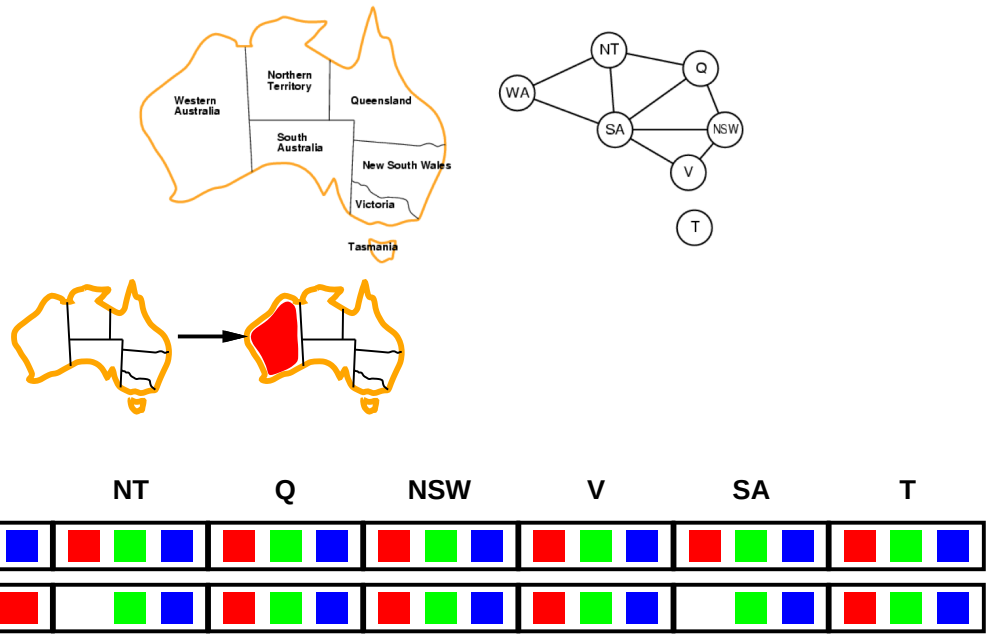
- Seguir la pista de los valores legales restantes para variables no asignadas.
- Finalizar la búsqueda si ninguna variable tiene ningún valor legal.



Propagación de la información a través de las restricciones

Comprobación hacia adelante

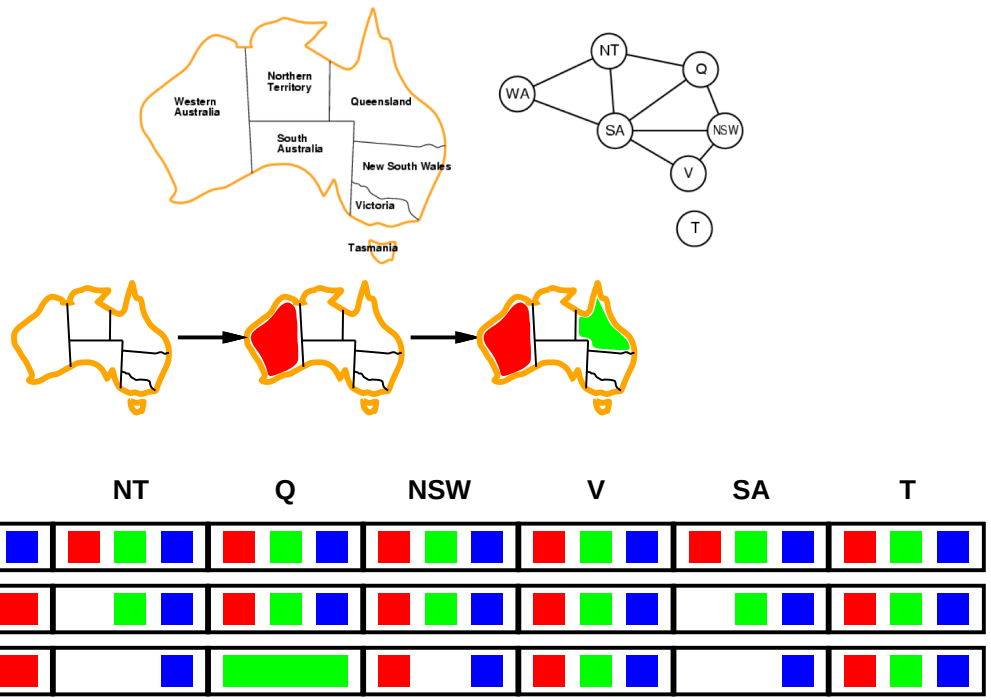
- Seguir la pista de los valores legales restantes para variables no asignadas.
- Finalizar la búsqueda si ninguna variable tiene ningún valor legal.



Propagación de la información a través de las restricciones

Comprobación hacia adelante

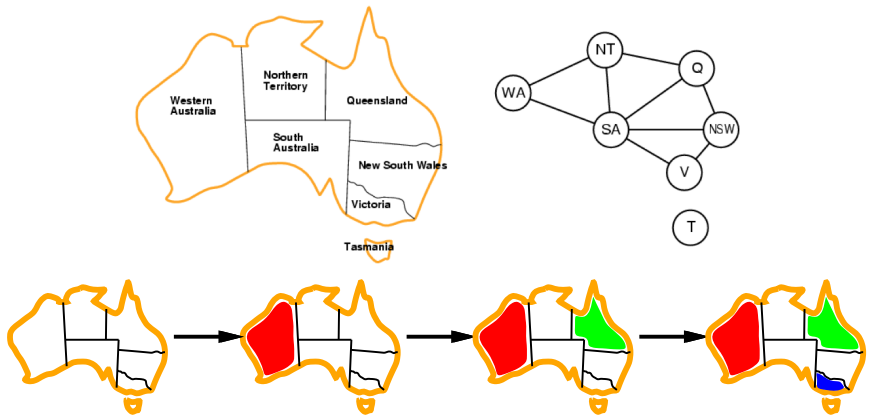
- Seguir la pista de los valores legales restantes para variables no asignadas.
- Finalizar la búsqueda si ninguna variable tiene ningún valor legal.

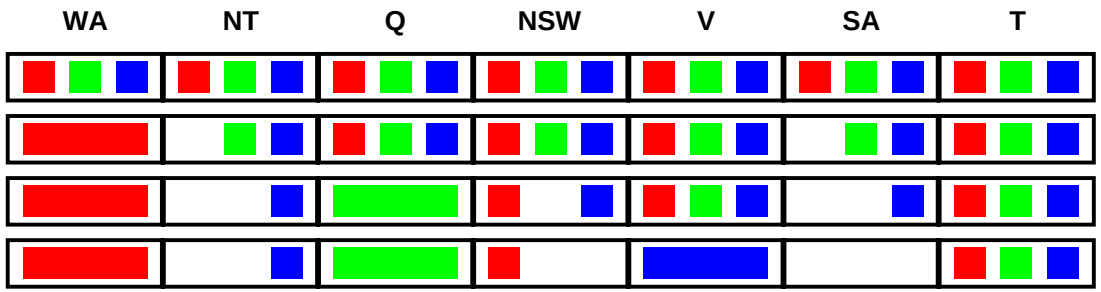


Propagación de la información a través de las restricciones

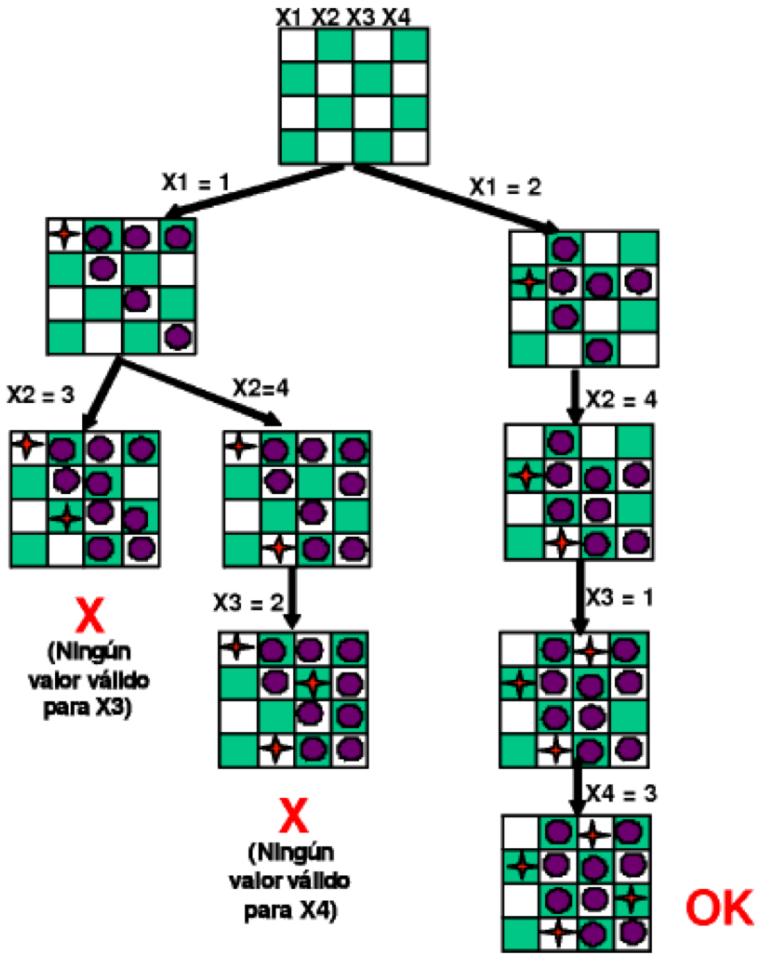
Comprobación hacia adelante

- Seguir la pista de los valores legales restantes para variables no asignadas.
- Finalizar la búsqueda si ninguna variable tiene ningún valor legal.





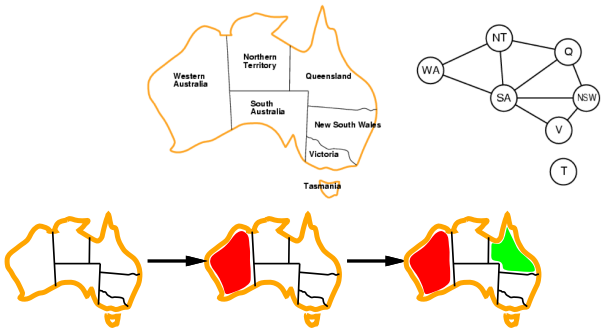
Ejemplo: Comprobación hacia delante aplicado al problema de las 4 reinas



Propagación de la información a través de las restricciones

Propagación de restricciones

- La comprobación hacia adelante propaga la información de variables asignadas a no asignadas pero no proporciona una detección temprana de fallos.



- Pero `TN` y `AS` no pueden tener ambas color azul.
- La propagación de restricciones refuerza de forma continua las restricciones localmente.

Propagación de la información a través de las restricciones

- Se pueden definir propiedades sobre los grafos de restricciones que permiten reducir el espacio de búsqueda.

Propagación de arco consistente

- **k-consistency:** poda de valores que no sean posibles para un grupo de k variables.
- **Arc consistency (2-consistency):** Eliminamos valores imposibles para parejas de variables.
- **Path consistency (3-consistency):** Eliminamos valores imposibles para ternas de variables

Propagación de la información a través de las restricciones

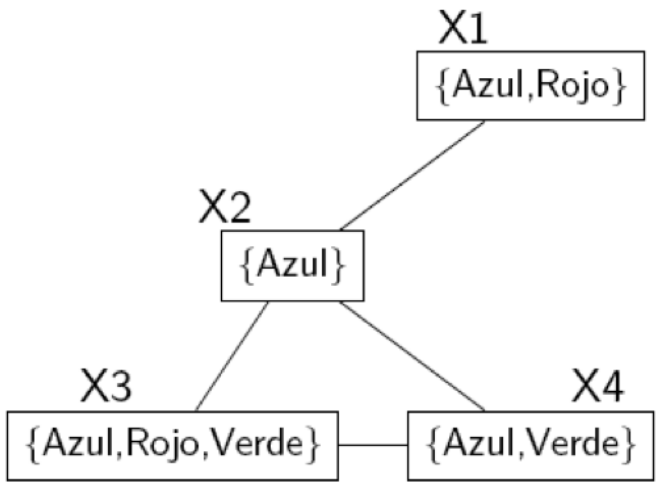
Propagación de arco consistente

- La forma más simple de propagación hace que cada arco sea consistente.
- $X \rightarrow Y$ es consistente si para cada valor de X existe algún valor permitido de Y.
- Si $\mathtt{X}$ pierde un valor es necesario volver a comprobar a sus vecinos.
- Detecta fallos antes que la comprobación hacia delante.
- Se puede ejecutar como un preprocesamiento de cualquier asignación.

Propagación de la información a través de las restricciones

Propagación de arco consistente

- Ejemplo de Arco Consistente.

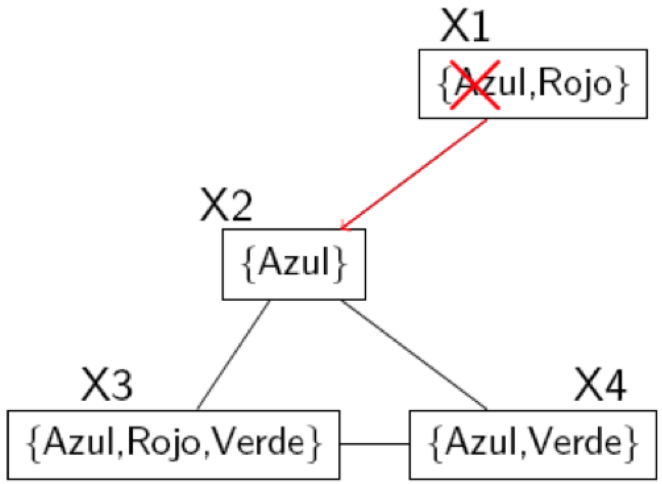


Lista de arcos inicial:
(X1,X2), (X2,X1), (X2,X3), (X3,X2),
(X2,X4), (X4,X2), (X3,X4), (X4,X3)

Propagación de la información a través de las restricciones

Propagación de arco consistente

- Ejemplo de Arco Consistente.



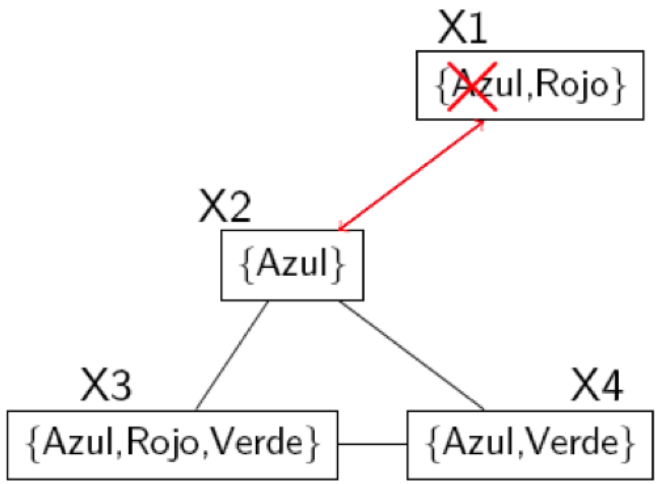
1. $X_1 - X_2 \rightarrow$ Quitar Azul de X_1

Lista de arcos inicial:
(X1,X2), (X2,X1), (X2,X3), (X3,X2),
(X2,X4), (X4,X2), (X3,X4), (X4,X3)

Propagación de la información a través de las restricciones

Propagación de arco consistente

- Ejemplo de Arco Consistente.



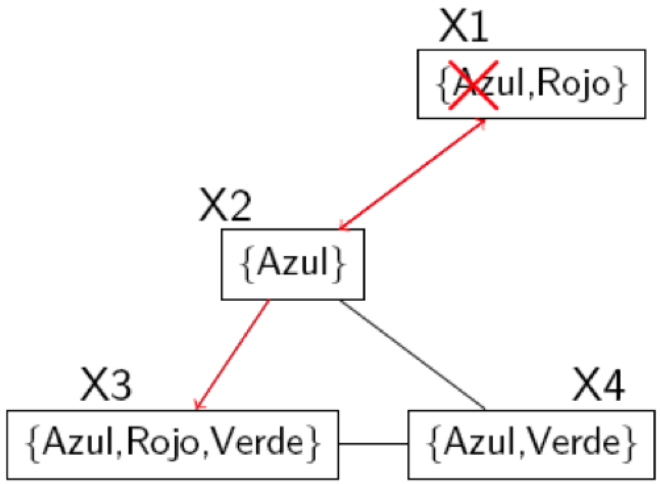
1. $X_1 - X_2 \rightarrow$ Quitar Azul de X_1
2. $X_2 - X_1 \rightarrow$ Todo consistente

Lista de arcos inicial:
(X1,X2), (X2,X1), (X2,X3), (X3,X2),
(X2,X4), (X4,X2), (X3,X4), (X4,X3)

Propagación de la información a través de las restricciones

Propagación de arco consistente

- Ejemplo de Arco Consistente.



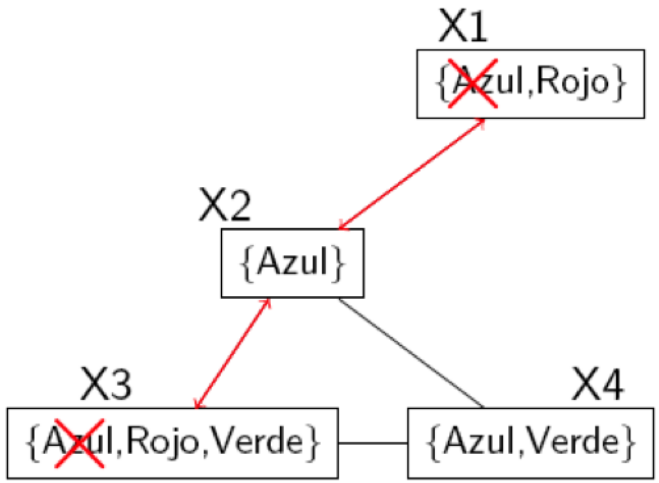
1. $X_1 - X_2 \rightarrow$ Quitar Azul de X_1
2. $X_2 - X_1 \rightarrow$ Todo consistente
3. $X_2 - X_3 \rightarrow$ Todo consistente

Lista de arcos inicial:
(X1,X2), (X2,X1), (X2,X3), (X3,X2),
(X2,X4), (X4,X2), (X3,X4), (X4,X3)

Propagación de la información a través de las restricciones

Propagación de arco consistente

- Ejemplo de Arco Consistente.



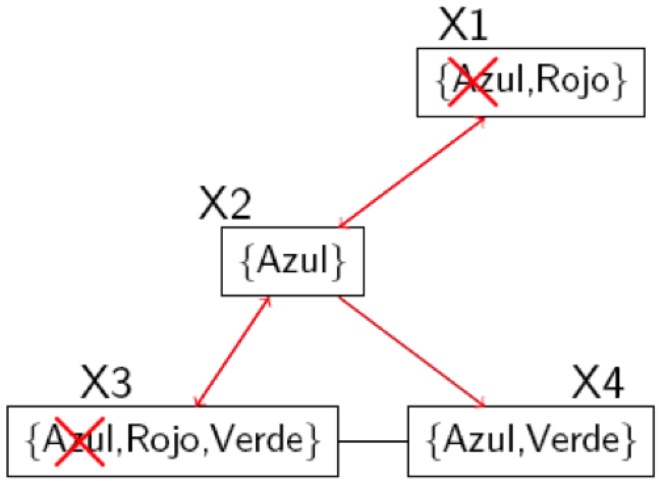
1. $X_1 - X_2 \rightarrow$ Quitar Azul de X_1
2. $X_2 - X_1 \rightarrow$ Todo consistente
3. $X_2 - X_3 \rightarrow$ Todo consistente
4. $X_3 - X_2 \rightarrow$ Quitar Azul de X_3, Tendríamos que añadir $X_4 - X_3$ pero ya está

Lista de arcos inicial:
(X1,X2), (X2,X1), (X2,X3), (X3,X2),
(X2,X4), (X4,X2), (X3,X4), (X4,X3)

Propagación de la información a través de las restricciones

Propagación de arco consistente

- Ejemplo de Arco Consistente.



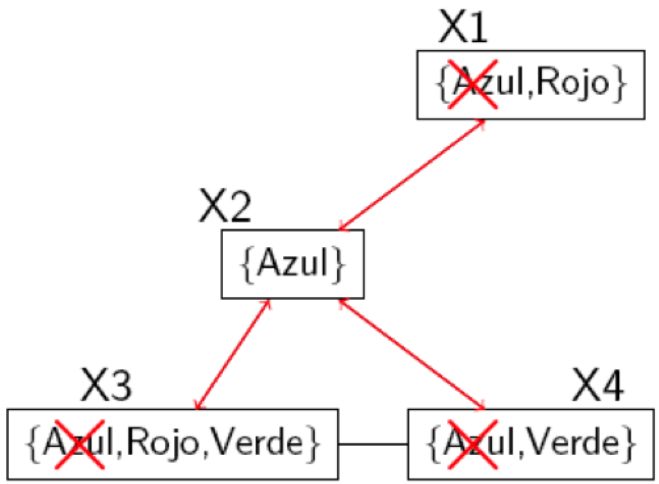
1. $X_1 - X_2 \rightarrow$ Quitar Azul de X_1
2. $X_2 - X_1 \rightarrow$ Todo consistente
3. $X_2 - X_3 \rightarrow$ Todo consistente
4. $X_3 - X_2 \rightarrow$ Quitar Azul de X_3, Tendríamos que añadir $X_4 - X_3$ pero ya está
5. $X_2 - X_4 \rightarrow$ Todo consistente

Lista de arcos inicial:
(X1,X2), (X2,X1), (X2,X3), (X3,X2),
(X2,X4), (X4,X2), (X3,X4), (X4,X3)

Propagación de la información a través de las restricciones

Propagación de arco consistente

- Ejemplo de Arco Consistente.



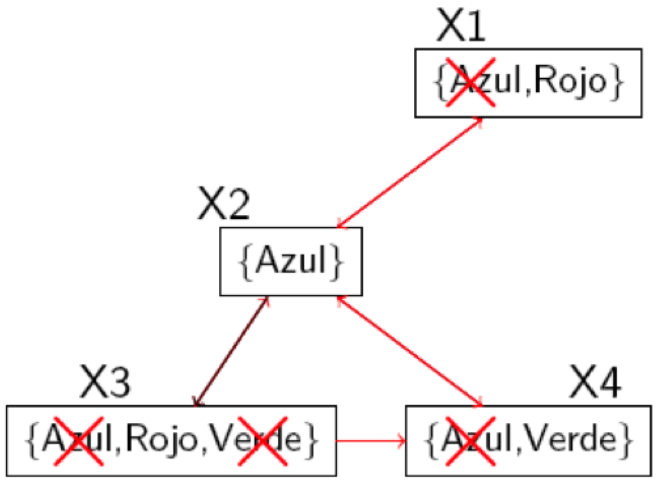
Lista de arcos inicial:
(X1,X2), (X2,X1), (X2,X3), (X3,X2),
(X2,X4), (X4,X2), (X3,X4), (X4,X3)

1. $X_1 - X_2 \rightarrow$ Quitar Azul de X_1
2. $X_2 - X_1 \rightarrow$ Todo consistente
3. $X_2 - X_3 \rightarrow$ Todo consistente
4. $X_3 - X_2 \rightarrow$ Quitar Azul de X_3, Tendríamos que añadir $X_4 - X_3$ pero ya está
5. $X_2 - X_4 \rightarrow$ Todo consistente
6. $X_4 - X_2 \rightarrow$ Quitar Azul de X_4, Tendríamos que añadir $X_3 - X_4$ pero ya está

Propagación de la información a través de las restricciones

Propagación de arco consistente

- Ejemplo de Arco Consistente.



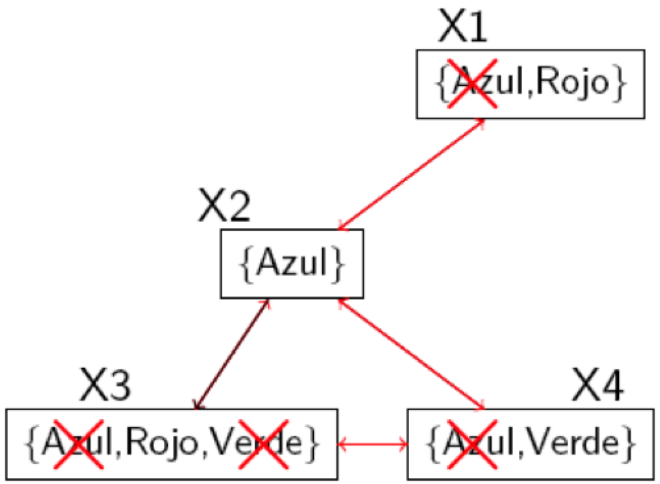
Lista de arcos inicial:
(X1,X2), (X2,X1), (X2,X3), (X3,X2),
(X2,X4), (X4,X2), (X3,X4), (X4,X3)

1. $X_1 - X_2 \rightarrow$ Quitar Azul de X_1
2. $X_2 - X_1 \rightarrow$ Todo consistente
3. $X_2 - X_3 \rightarrow$ Todo consistente
4. $X_3 - X_2 \rightarrow$ Quitar Azul de X_3, Tendríamos que añadir $X_4 - X_3$ pero ya está
5. $X_2 - X_4 \rightarrow$ Todo consistente
6. $X_4 - X_2 \rightarrow$ Quitar Azul de X_4, Tendríamos que añadir $X_3 - X_4$ pero ya está
7. $X_3 - X_4 \rightarrow$ Quitar Verde de X_3, Añadimos $X_2 - X_3$

Propagación de la información a través de las restricciones

Propagación de arco consistente

- Ejemplo de Arco Consistente.



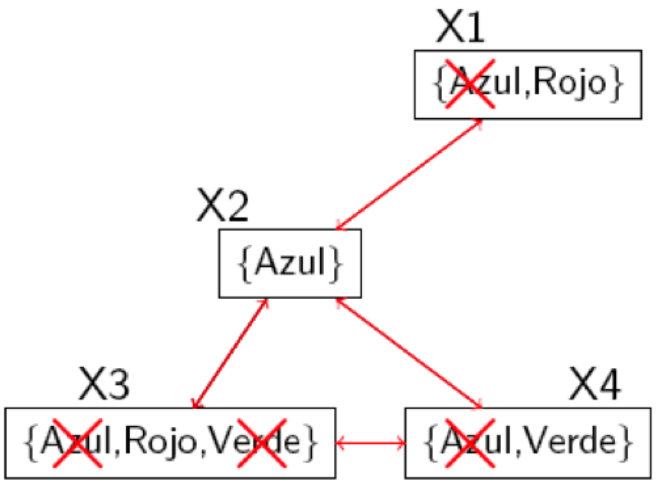
Lista de arcos inicial:
(X1,X2), (X2,X1), (X2,X3), (X3,X2),
(X2,X4), (X4,X2), (X3,X4), (X4,X3)

1. $X_1 - X_2 \rightarrow$ Quitar Azul de X_1
2. $X_2 - X_1 \rightarrow$ Todo consistente
3. $X_2 - X_3 \rightarrow$ Todo consistente
4. $X_3 - X_2 \rightarrow$ Quitar Azul de X_3, Tendríamos que añadir $X_4 - X_3$ pero ya está
5. $X_2 - X_4 \rightarrow$ Todo consistente
6. $X_4 - X_2 \rightarrow$ Quitar Azul de X_4, Tendríamos que añadir $X_3 - X_4$ pero ya está
7. $X_3 - X_4 \rightarrow$ Quitar Verde de X_3, Añadimos $X_2 - X_3$
8. $X_4 - X_3 \rightarrow$ Todo consistente

Propagación de la información a través de las restricciones

Propagación de arco consistente

- Ejemplo de Arco Consistente.



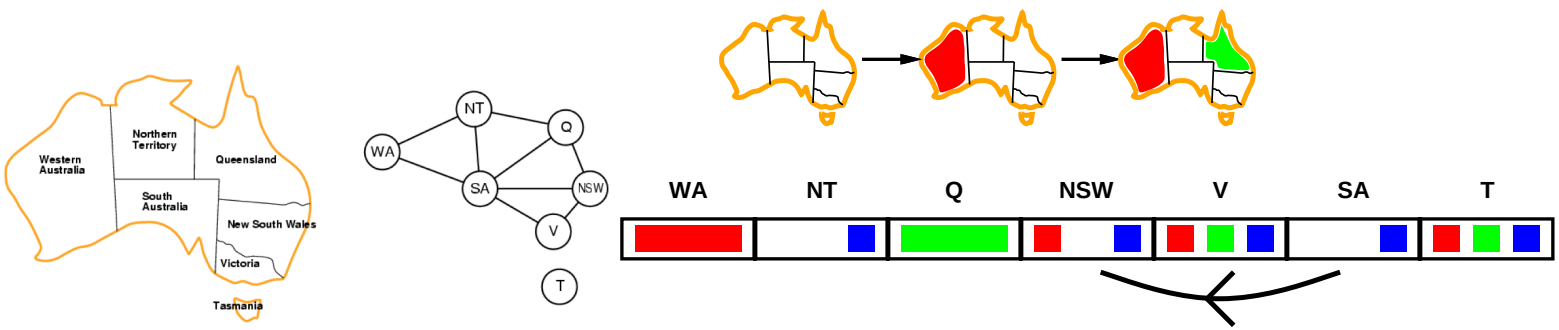
Lista de arcos inicial:
(X1,X2), (X2,X1), (X2,X3), (X3,X2), (X2,X4), (X4,X2), (X3,X4), (X4,X3)

1. $X_1 - X_2 \rightarrow$ Quitar Azul de X_1
2. $X_2 - X_1 \rightarrow$ Todo consistente
3. $X_2 - X_3 \rightarrow$ Todo consistente
4. $X_3 - X_2 \rightarrow$ Quitar Azul de X_3, Tendríamos que añadir $X_4 - X_3$ pero ya está
5. $X_2 - X_4 \rightarrow$ Todo consistente
6. $X_4 - X_2 \rightarrow$ Quitar Azul de X_4, Tendríamos que añadir $X_3 - X_4$ pero ya está
7. $X_3 - X_4 \rightarrow$ Quitar Verde de X_3, Añadimos $X_2 - X_3$
8. $X_4 - X_3 \rightarrow$ Todo consistente
9. $X_2 - X_3 \rightarrow$ Todo consistente

Propagación de la información a través de las restricciones

Propagación de arco consistente

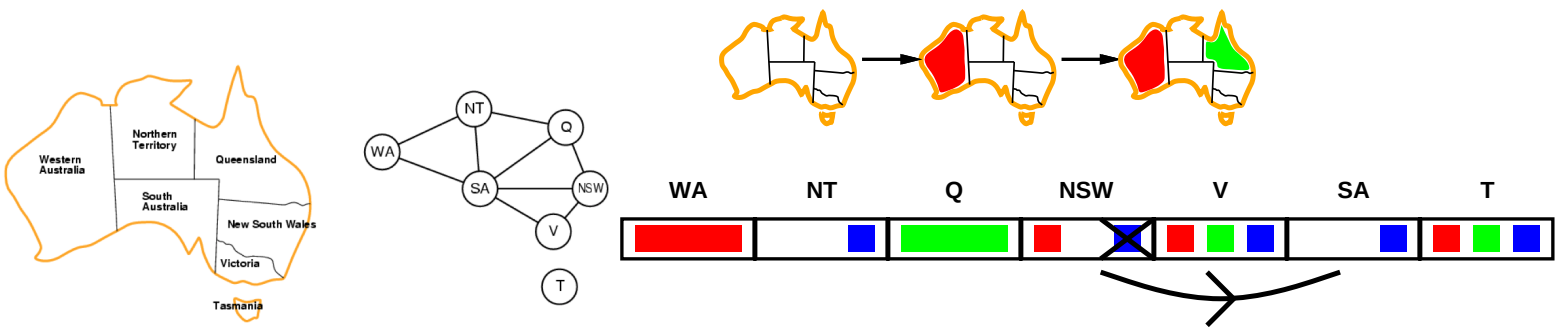
- Ejemplo de Arco Consistente.



Propagación de la información a través de las restricciones

Propagación de arco consistente

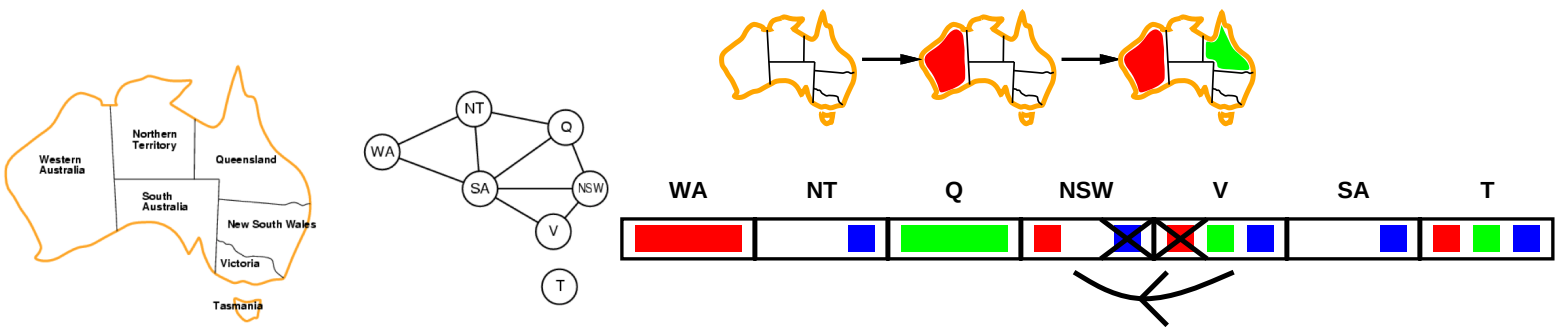
- Ejemplo de Arco Consistente.



Propagación de la información a través de las restricciones

Propagación de arco consistente

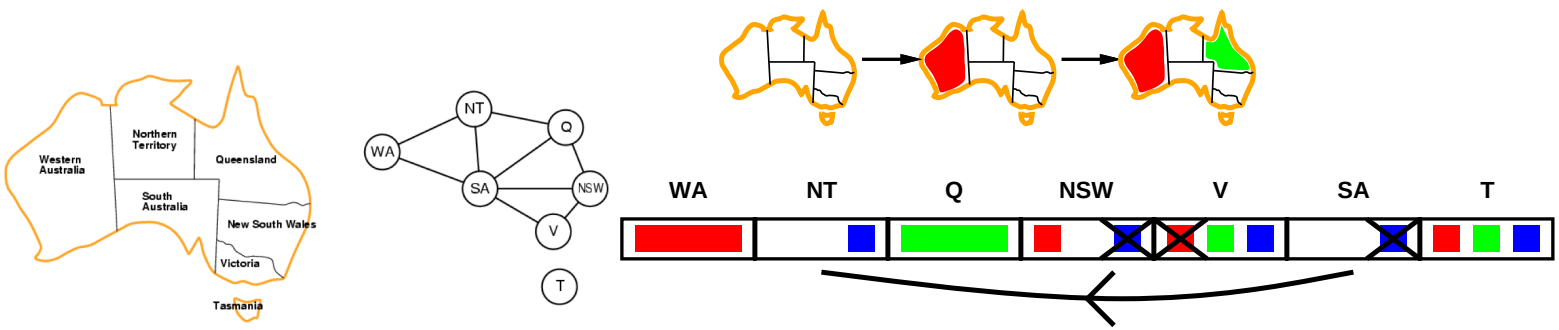
- Ejemplo de Arco Consistente.



Propagación de la información a través de las restricciones

Propagación de arco consistente

- Ejemplo de Arco Consistente.



Propagación de la información a través de las restricciones

Algorithm 3.1: función AC-3 (*psr*) **devuelve** el PSR

Data: *psr*, un PSR binario con variables $X_1, X_2, \ldots, X_n$
Result: El PSR, posiblemente con dominio reducido

```
1  cola ← todos los arcos del psr
2  while cola ≠ ∅ do
3      (X_i, X_j) ← BorrarPrimero(cola)
4      if BorrarValoresInconsistentes(X_i, X_j) then
5          for cada X_k en Vecinos[X_i] do
6              añadir (X_k, X_i) a la cola
```

Después de aplicar AC-3, cada arco es arco-consistente, o alguna variable tiene un dominio vacío, indicando que el PSR no puede hacerse arco-consistente (y no puede resolverse).

Propagación de la información a través de las restricciones

Algorithm 3.2: función BorrarValoresInconsistentes (X_i, X_j) **devuelve** verdadero si y sólo si hemos borrado un valor

Result: *verdadero* si se ha borrado un valor, *falso* e.c.c

```
1  borrado ← falso
2  for cada x en Dominio[X_i] do
3      if no hay un y en Dominio[X_j] que permita a (x, y) satisfacer la
       restricción entre X_i y X_j then
4          borrar x de Dominio[X_i]
5          borrado ← verdadero
6  return borrado
```

Búsqueda local para PSR

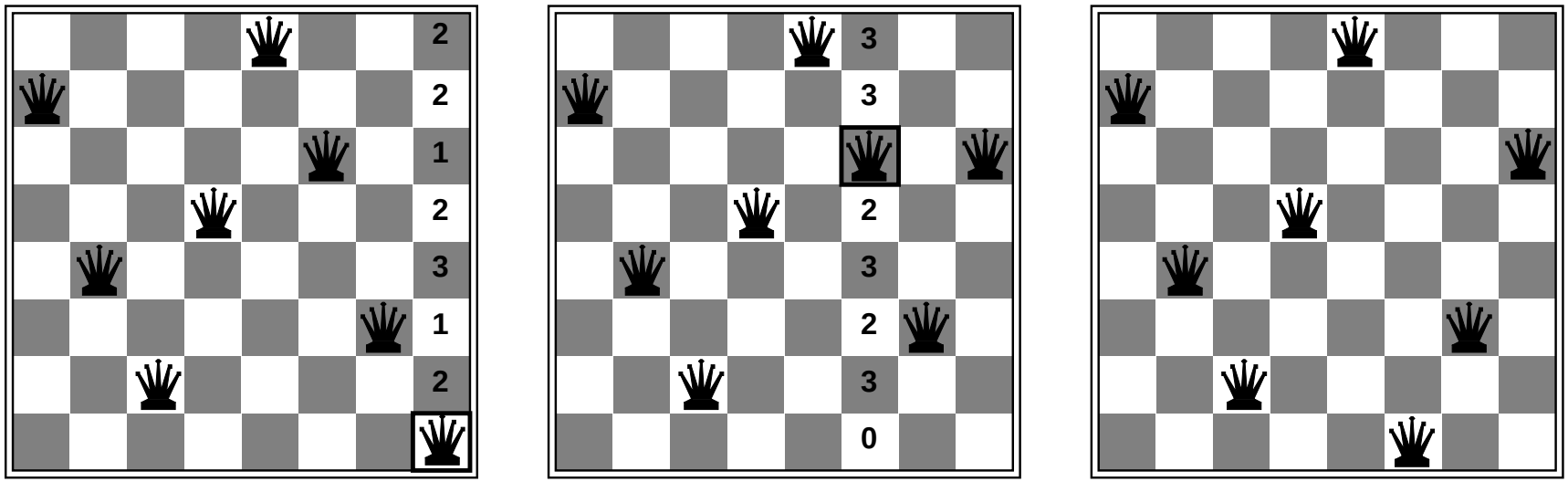
- Los algoritmos de ascensión de colinas o temple simulado trabajan habitualmente con estados completos (con todas las variables asignadas).
- Para aplicarlos a PRS:
  - Se permitirán estados que no satisfagan todas las restricciones.
  - Los operadores reasignarán los valores a las variables.
- Selección de variables: Aleatoria entre las variables conflictivas.
- Selección de valor: Mediante la heurística de mínimos conflictos: elegir el valor que incumple el número menor de restricciones.

Búsqueda local para PSR

Algorithm 3.3: función MínimosConflictos (*psr*, *maxpasos*) **devuelve** una solución o fallo

```
1  actual ← una asignación completa inicial para psr
2  for i = 1 to maxpasos do
3      if actual es una solución para psr then
4          return actual
5      var ← el valor v para var que minimiza Conflictos(var, v, actual, psr)
       conjunto var =valor en actual
```

Búsqueda local para PSR



Una solución de dos pasos para un problema de 8-reinas utilizando mínimos conflictos.

Índice

Resumen

- Los PSR son un tipo especial de problema en los que:
  - Los estados se definen a través de los valores que se asignan a un conjunto de variables.
  - El test objetivo se define mediante las restricciones en los valores de las variables.
- Algoritmo de búsqueda estándar más utilizado en PSR

  Backtracking = Búsqueda de primero en profundidad con una variable asignada por nodo.
- Las heurísticas de ordenación de variables y selección de valores ayudan significativamente.
- La comprobación hacia delante previene asignaciones que garantizan un fallo posterior.
- La propagación de restricciones (p.e. la consistencia de arcos) hace un trabajo adicional en restricción de valores y detección de inconsistencias.
- La búsqueda local basada en mínimos conflictos es efectiva en la práctica.

Índice

Bibliografía

- S. Russell, P. Norvig. Inteligencia Artificial. Un enfoque moderno. Segunda edición. Prentice Hall, 2004.
- S. Russel, P. Norvig. Artificial Intelligence. A modern approach. Third edition. Prentice Hall, 2010.
- Guía sobre problemas de satisfacción de restricciones: *On-Line Guide To Constraint Programming* de R.Barták accesible en http://ktiml.mff.cuni.cz/~bartak/constraints/index.html