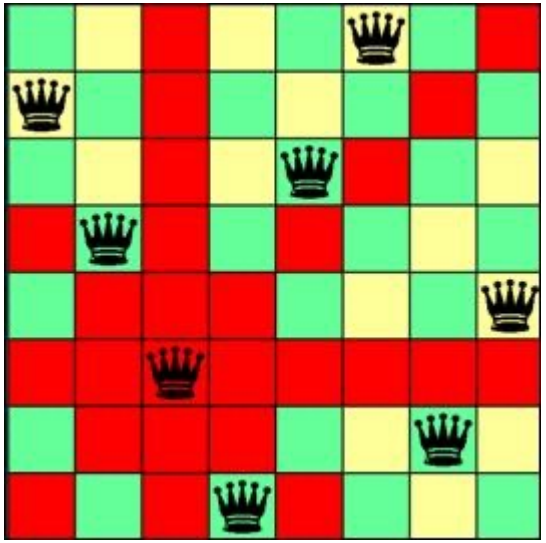
Tema 2: Problemas de satisfacción de restricciones

- Satisfacción de restricciones
- Métodos de búsqueda
- Búsqueda local para problemas de satisfacción de restricciones



Objetivos

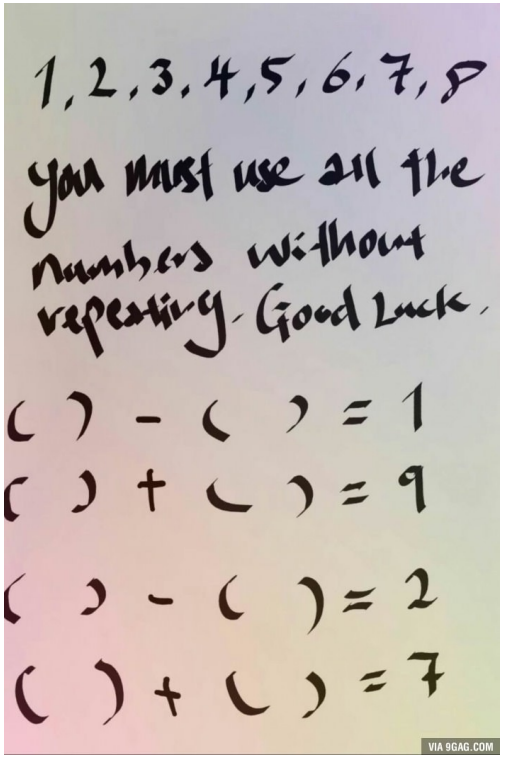
- Definir los modelos de resolución de los problemas de satisfacción de restricciones.
- Analizar el uso de la búsqueda heurística en la resolución de estos problemas y la aplicación de diversas heurísticas de carácter general.

Estudia el tema en …

- S. Russell, P. Norvig, Artificial Intelligence: A modern Approach, Tercera Edición, Ed. Pearson, 2010.

Motivación

- Problema



- Un posible modelo
- **Variables**
- **Dominios**
  - Dom()=[1,2,3,4,5,6,7,8]
- **Restricciones**
  - …
  - Sin repetir valores
    - …

Motivación

- Un modelo
- **Variables**
- **Dominios**
  - Dom()=[1,2,3,4,5,6,7,8]
- **Restricciones**
  - …
  - Sin repetir valores
    - …

- ¿Cómo resuelvo el problema así modelado?
- **Búsqueda,** por ejemplo:
  - Asignar valores a las variables
  - Comprobar restricciones
  - Si se cumplen las restricciones
  - Solución = Asignación actual
  - Si no volver a asignar…

Motivación

- Las claves:
  - Aprender a modelar un problema dado como un Problema de Satisfacción de Restricciones (CSP: Constraints Satisfaction Problem).
    - Variables, Dominios, Restricciones
    - Entender que un CSP, una vez modelado, se resuelve mediante búsqueda.
  - Conocer técnicas de búsqueda para resolverlo.
  - Conocer heurísticas para mejorar los procesos de búsqueda en CSPs.

Aplicaciones: Asignación de mostradores/puertas a vuelos

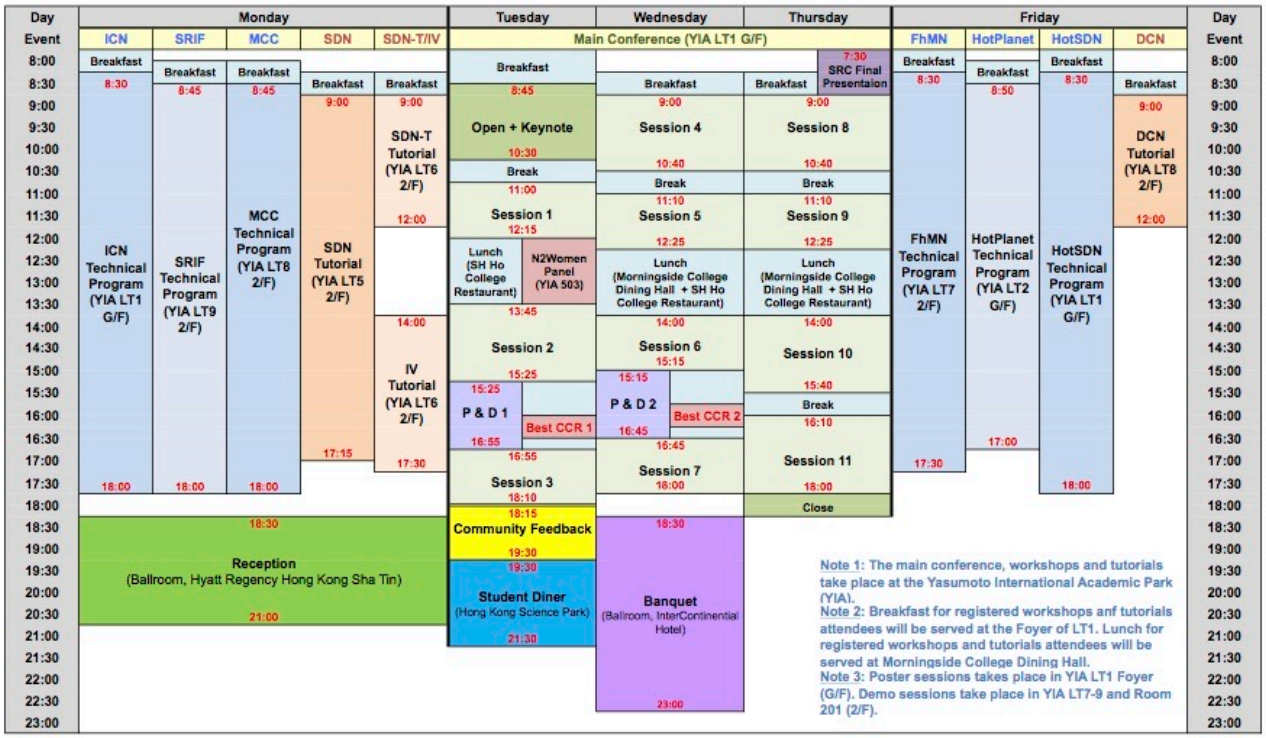
Problema de asignación de mostradores en aeropuertos: asignar suficientes mostradores y personal (el número depende del tipo de aeronave) a cada vuelo. Los mostradores están agrupados en islas y para cada vuelo todos los mostradores asignados tienen que estar en la misma isla. El personal tiene regulaciones de trabajo que debe cumplir (descansos, bajas, etc.).

Aplicaciones: Planificación de Horarios, Asignación de aulas (conferencias, facultades, ….)

Una conferencia consta de N sesiones de igual duración. El programa se organizará como una secuencia de slots, donde cada uno contiene hasta 5 sesiones paralelas. Hay restricciones concretas de sesión:

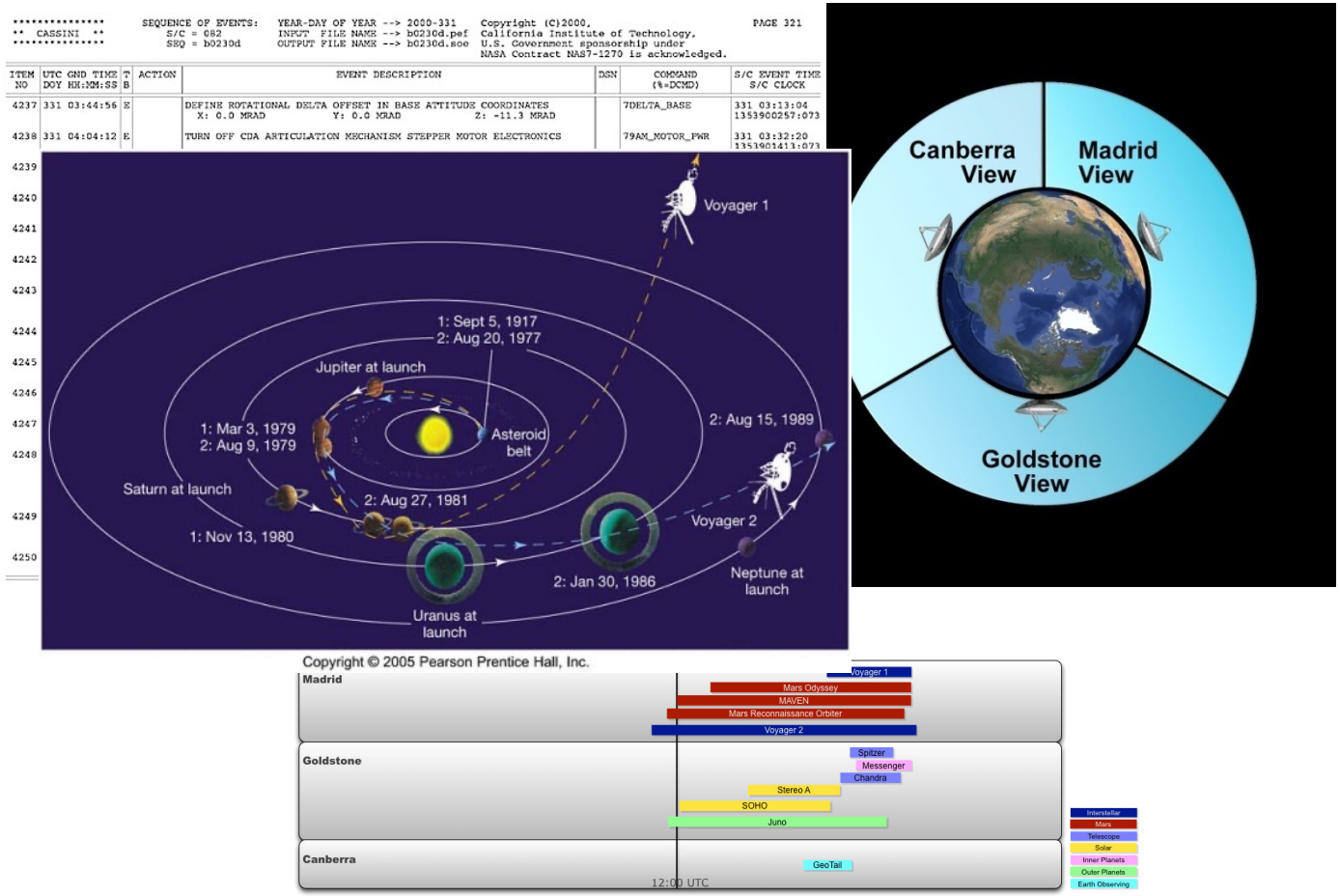
1. Sesión 4 debe tener lugar antes de la sesión 11.
2. Sesión 5 debe tener lugar antes de la sesión 10.
3. Sesión 6 debe tener lugar antes de la sesión 11.
...
8. Sesión 6 no debe estar en paralelo con 7 y 10.
9. Sesión 7 no debe estar en paralelo con 8 y 9.
10. Sesión 8 no debe estar en paralelo con 10.

Minimizar el número de slots de tiempo.



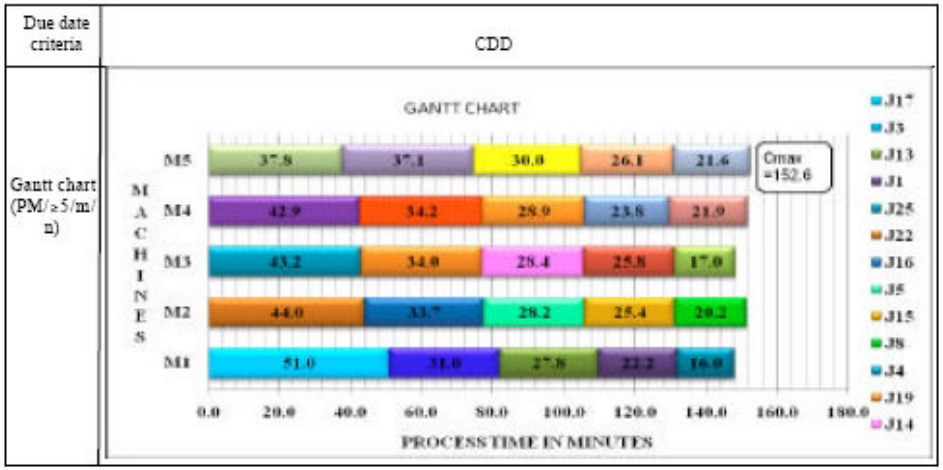
Aplicaciones: Planificación de operaciones en espacio profundo

Cada sonda enviada tiene slots de tiempo en los que se comunica con la Tierra, en distintos centros de recepción de señales.



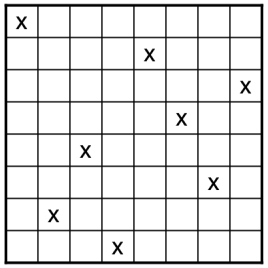
Aplicaciones: Job Shop Scheduling

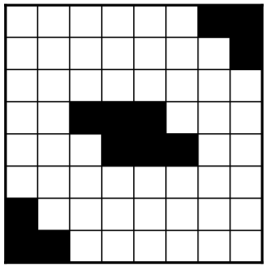
Hay N puestos de trabajo y M máquinas . Cada trabajo requiere la ejecución de una secuencia de operaciones dentro de un intervalo de tiempo, y cada operación Oi requiere el uso exclusivo de una máquina designada Mi para una cantidad especificada pi de tiempo de procesamiento. Determinar un horario para la producción que satisfaga las restricciones de tiempo y de capacidad de recursos.



Problemas de satisfacción de restricciones

- Un tipo particular de problemas que van a tener asociados:
  - Métodos de búsqueda particulares
  - Heurísticas de propósito general
  - Se hará uso de la estructura interna del estado





Problema de satisfacción de restricciones

- Un problema de satisfacción de restricciones (CSP) está definido por X,D,C:
  - **X** es un conjunto de variables $\{X_1, X_2,\ldots,X_n\}$.
  - **D** es un conjunto de dominios $\{D_1, D_2,\ldots,D_n\}$, uno por cada variable
  - **C** es un conjunto de restricciones $\{C_1,C_2,\ldots,C_m\}$, que especifican combinaciones permitidas de valores
- Cada dominio D_i representa un conjunto $\{v_1,\ldots,v_k\}$ de valores permitidos para X_i
- Cada restricción Ci =
  - <lista de variables, relación entre variables>

Ejemplo

- X = {X1,X2}
- D1=D2 ={a,b}
- C1 = *puede representarse como...*
  - <(X1,X2), X1≠X2> definición intensional
  - <(X1,X2), [(a,b),(b,a)]> definición extensional
  - X1≠X2 abreviada

- **Estado para cualquier problema**
  - es una ***asignación de valores*** a algunas variables (o a todas) $\{X_i=v_i,X_j=v_j,...\}$
    - {X1 = a, X2 = a}
    - {X1 = a, X2 = b}
    - {X1 = b, X2 = a}
    - {X1 = b, X2 = b}

Satisfacción de restricciones

- Asignación **consistente**:
  - Una asignación que ***no viola ninguna restricción***
- Asignación **completa**
  - Una asignación en la que aparecen ***todas las variables***.
- Asignación **parcial**:
  - Una asignación en la que aparecen ***algunas variables***
- **Solución**
  - Una asignación **completa** y **consistente**.
- Algunos CSPs requieren una solución que maximiza/minimiza una función objetivo
  - COP: Constraint Optimization Problems, problemas de optimización de restricciones.
  - Ejemplo: encontrar una asignación de valores a variables que minimice su suma total.

Coloreado de mapas

- 7 regiones en Australia
- 3 colores
- regiones vecinas tienen distinto color

Variables : *cada variable representa una región*

X = {WA,NT,Q,NSW,V,SA,T}

Dominios : *cada variable puede tener 3 valores posibles*

D_i={red,green,blue}

Restricciones

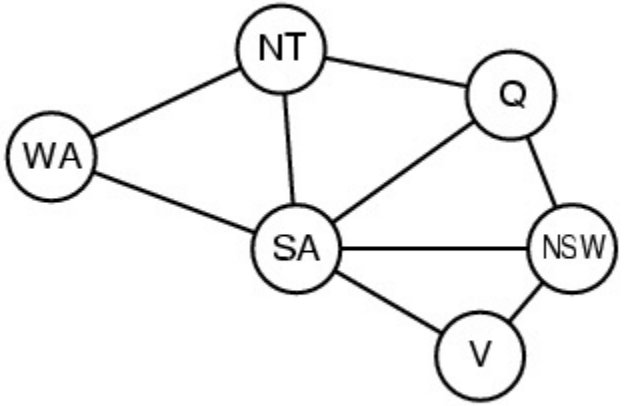
C1 = WA ≠ NT :*el valor de WA distinto del de NT*

= (WA, NT) in `{(red,green),(red,blue),(green,red),(green,blue),(blue,red),(blue,green)}`

.....

C = {SA ≠ WA, SA ≠ NT, SA ≠ Q,SA ≠ NSW, SA ≠ V, WA ≠ NT, NT ≠ Q, Q ≠ NSW, NSW ≠ V}

- Grafo de Restricciones



NODOS = Variables
ARCOS=Restricciones

Coloreado de mapas

- 7 regiones en Australia
- 3 colores
- regiones vecinas con distinto color

Variables

X = {WA,NT,Q,NSW,V,SA,T}

Dominios

D_i={red,green,blue}

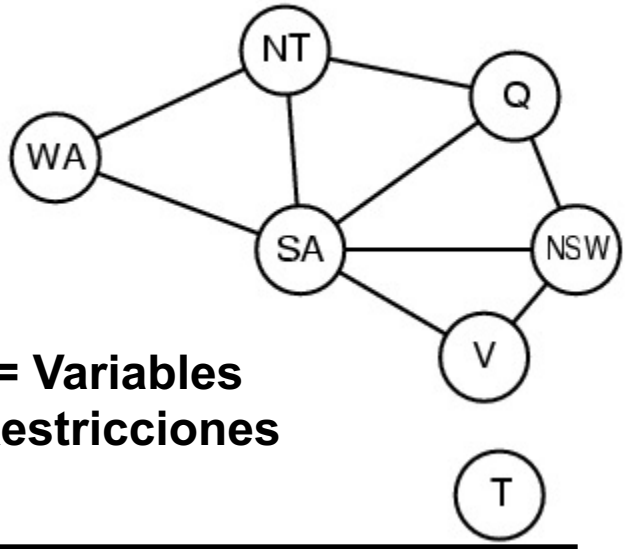
Restricciones

C = {SA ≠ WA, SA ≠ NT, SA ≠ Q,SA ≠ NSW, SA ≠ V, WA ≠ NT, NT ≠ Q, Q ≠ NSW, NSW ≠ V}

Solución: asignación completa y consistente

{WA = red, NT = green,Q = red,NSW = green,V = red,SA = blue,T = green}

- Grafo de Restricciones



NODOS = Variables
ARCOS=Restricciones

Tipos de CSPs

Según tipo de variables

- Variables Discretas (enteros, símbolos, strings)
  - **dominios finitos:**
    - *n* variables, tamaño del dominio *d* → $O(d^n)$ asignaciones completas
    - Ejemplos: diseño de circuitos, demostración de teoremas, verificación y validación de software.
  - dominios infinitos:
    - enteros, cadenas, ...
    - Por ejemplo: job shop scheduling (asignación de tareas y máquinas)
      - variables son fechas inicio/fin de tareas
      - es necesario un **lenguaje de restricciones**
        - » p.e: $StartJob_1 + 5 \leq StartJob_3$

- Variables continuas (reales)
  - Por ejemplo, tiempo de inicio y final para observaciones del Hubble Space Telescope
  - Restricciones lineales
    - 3X + 5Y – Z > 2X-Z …

Tipos de CSPs

Según tipo de restricciones

- *Restricciones Unarias* involucran una sola variable,
  - e.g., SA ≠ green
- *Restricciones Binarias* involucran parejas de variables,
  - e.g., SA ≠ WA
- *Restricciones de orden superior* involucran 3 o más variables (*GLOBAL CONSTRAINTS*)
  - e.g., cryptarithmetic column constraints
- *Preferencias* (soft constraints)
  - e.g. *rojo* es mejor que *verde*

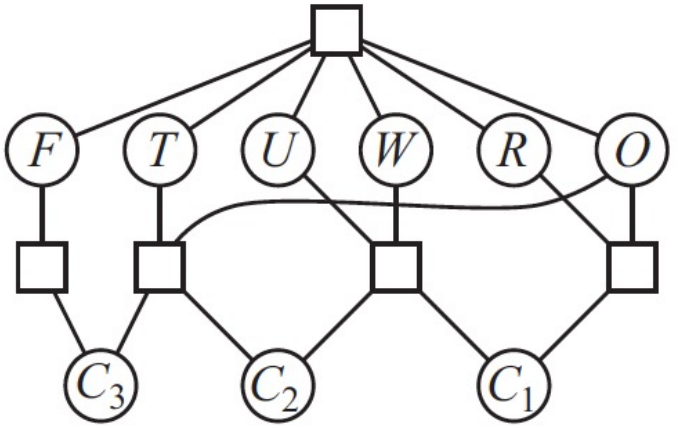
Puzles cripto-aritméticos

Ejemplo dominio discreto, infinito y restricciones globales

TodasDif(F,T,U,W,R,O)

$O+O=R+10\times C_1$
$C_1+W+W=U+10\times C_2$
$C_2+T+T=O+10\times C_3$
$C_3=F$

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



Hiper-grafo de restricciones

¿Cómo resolverlo?

- Formulación incremental
- Formulación completa de estados

Formulación incremental

- Estado inicial: la asignación vacía {}
- Función sucesor: un valor se puede asignar a cualquier variable no asignada, a condición de que no suponga ningún conflicto con variables antes asignadas
- Test objetivo: la asignación actual es completa
- Costo del camino: un costo constante para cada paso
- Para n variables, la solución se encuentra a una profundidad n

Formulación completa de estados

- Dado que el camino que alcanza una solución es irrelevante, se pueden manejar estados con asignaciones completas.
- Uso de métodos de búsqueda local.

Complejidad

- La obtención de soluciones en CSP es, en general, NP-completo
- La obtención de soluciones óptimas es NP-duro

- Se requiere una gran eficiencia en los procesos de búsqueda

Búsqueda en la formulación incremental

- Uso de búsqueda en anchura: se genera un árbol con $n! \times d^n$ hojas.
- Propiedad crucial común a todos los CSPs: la conmutatividad (el orden de aplicación de cualquier conjunto de acciones no tiene ningún efecto sobre el resultado).
- Técnica utilizada: Búsqueda en profundidad retroactiva.

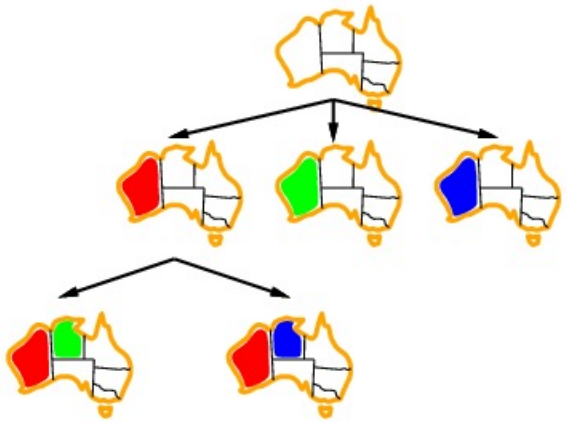
Búsqueda en profundidad retroactiva

```
function BACKTRACKING-SEARCH(csp) % returns a solution or failure
    return RECURSIVE-BACKTRACKING({} , csp)

function RECURSIVE-BACKTRACKING(assignment, csp) % returns a solution or failure
    if assignment is complete then return assignment
    var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp],assignment,csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
          if value is consistent with assignment according to CONSTRAINTS[csp]

                then
                  add {var=value} to assignment
                  result ← RECURSIVE-BACKTRACKING(assignment, csp)
                  if result ≠ failure  then return result
                  remove {var=value} from assignment
    return failure
```

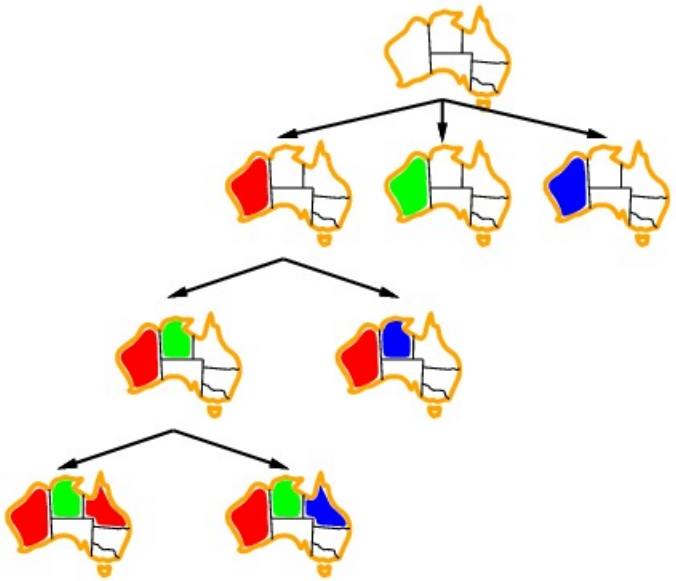
Parte de un árbol de búsqueda generado

Parte de un árbol de búsqueda generado

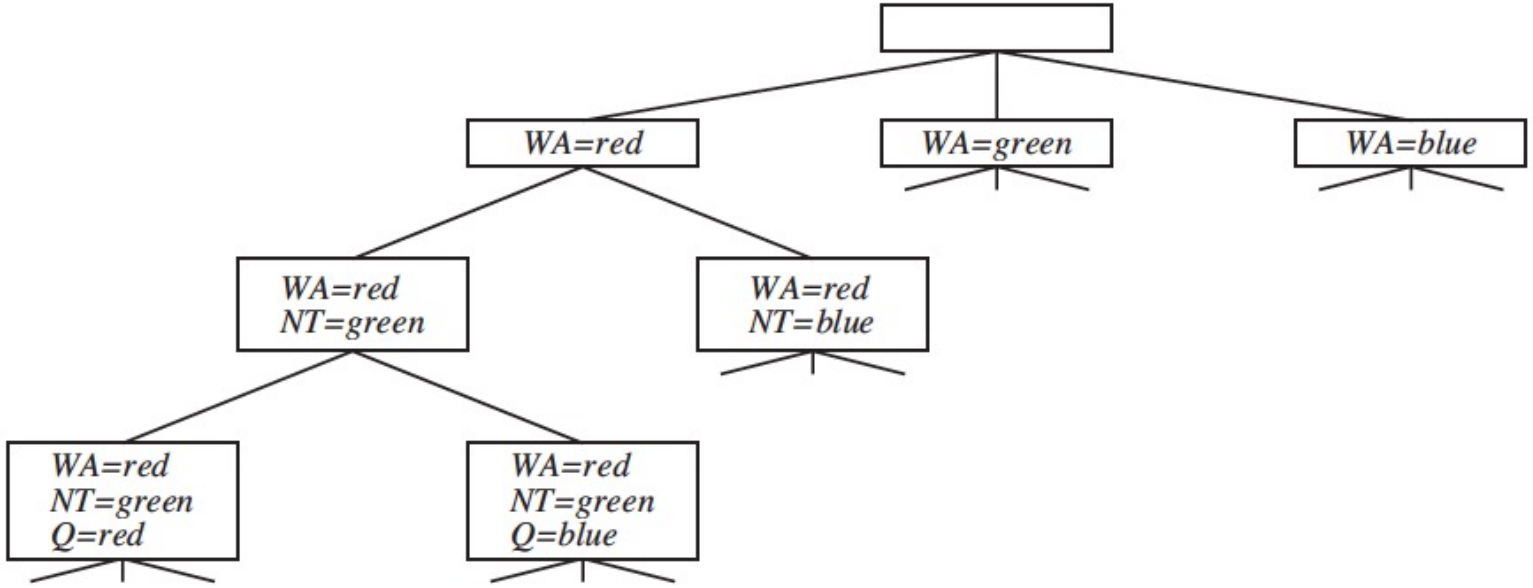
Parte de un árbol de búsqueda generado



Parte de un árbol de búsqueda generado



Parte de un árbol de búsqueda generado



Mejoras del modelo

- ¿Qué variable debe asignarse después, y en qué orden deberían intentarse sus valores?
- ¿Cuáles son las implicaciones de las variables actuales para el resto de variables no asignadas?
- Cuando un camino falla, ¿puede la búsqueda evitar repetir este fracaso en caminos siguientes?

Variable y ordenamiento de valor

- Mínimos valores restantes
- Grado heurístico
- Valor menos restringido

Mínimos valores restantes

- En principio no hay ningún criterio en la selección de la variable.
- Uso de la heurística de mínimos valores restantes (MVR) o de la variable más restringida.

Grado heurístico

- Selecciona la variable, entre las no asignadas, que esté implicada en el mayor número de restricciones

La heurística MVR es más poderosa, pero el grado heurístico es útil en casos de empate

Valor menos restringido

- Una vez seleccionada una variable debe decidirse un orden para examinar sus valores.
- Heurística del valor menos restringido: se prefiere el valor que excluye el menor número de restricciones en las variables vecinas del grafo.

WA=roja, NT=verde, Q=¿?
Q puede ser azul o roja

Si azul entonces AS no tiene alternativa

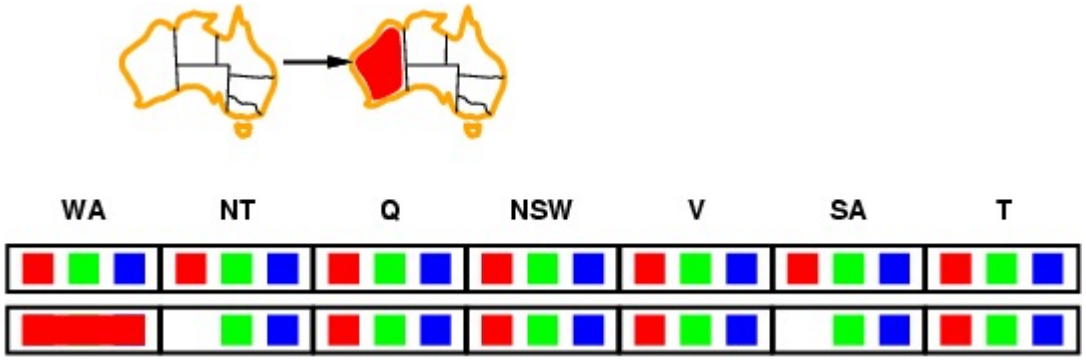
La heurística opta por Q=roja

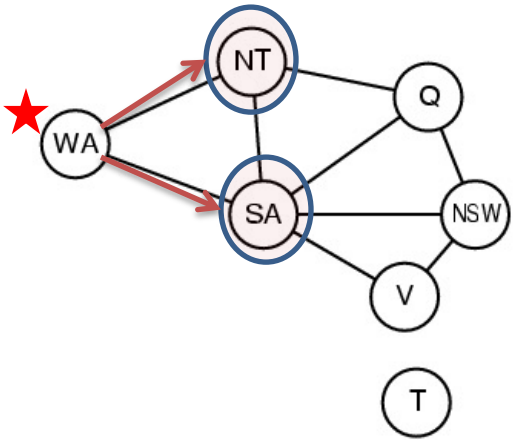
Propagación de la información a través de las restricciones

- Comprobación hacia delante
- Propagación de restricciones
- Vuelta atrás inteligente

Comprobación hacia delante (Forward checking)

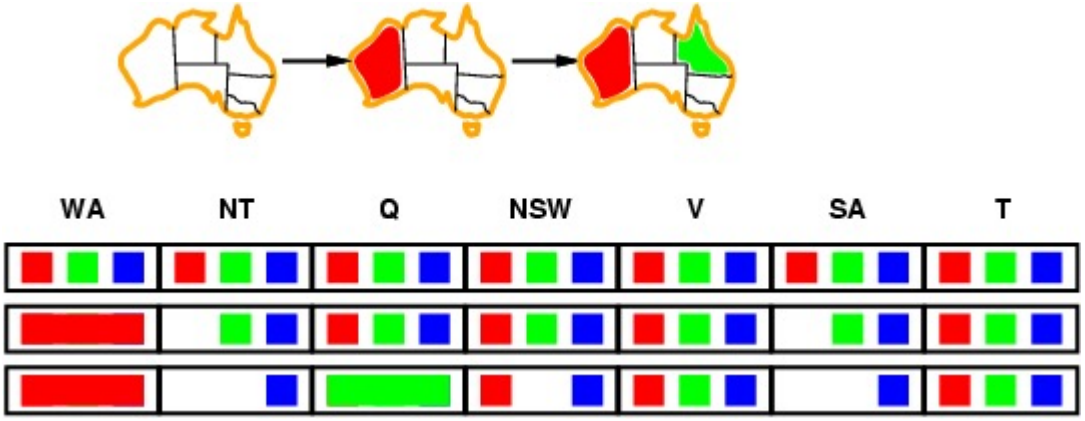
- Modificar los dominios para mantener sólo los valores legales para variables no asignadas.
- Si selecciono y asigno X, compruebo variables relacionadas con X en el grafo de restricciones, y elimino valores no legales.
- Backtracking cuando alguna variable tiene un dominio vacío.

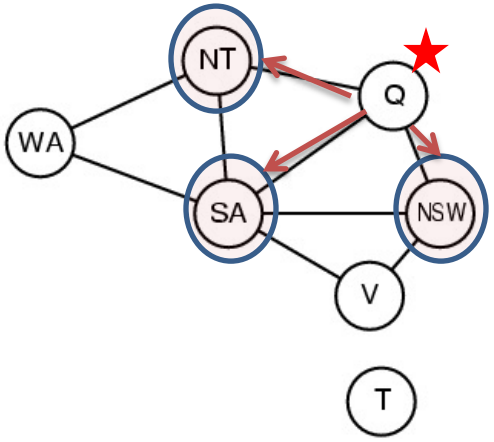




Comprobación hacia delante (Forward checking)

- Modificar los dominios para mantener sólo los valores legales para variables no asignadas.
- Si selecciono y asigno X, compruebo variables relacionadas con X en el grafo de restricciones, y elimino valores no legales.
- Backtracking cuando alguna variable tiene un dominio vacío.

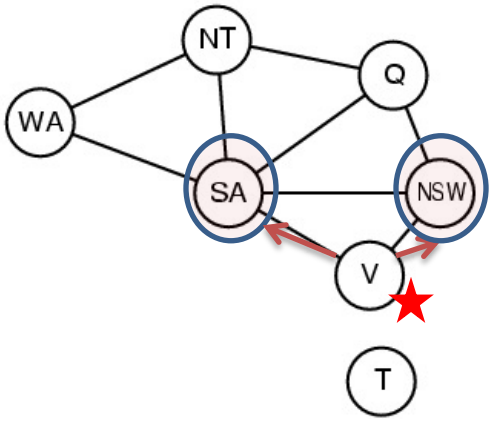




de restricciones

Comprobación hacia delante (Forward checking)

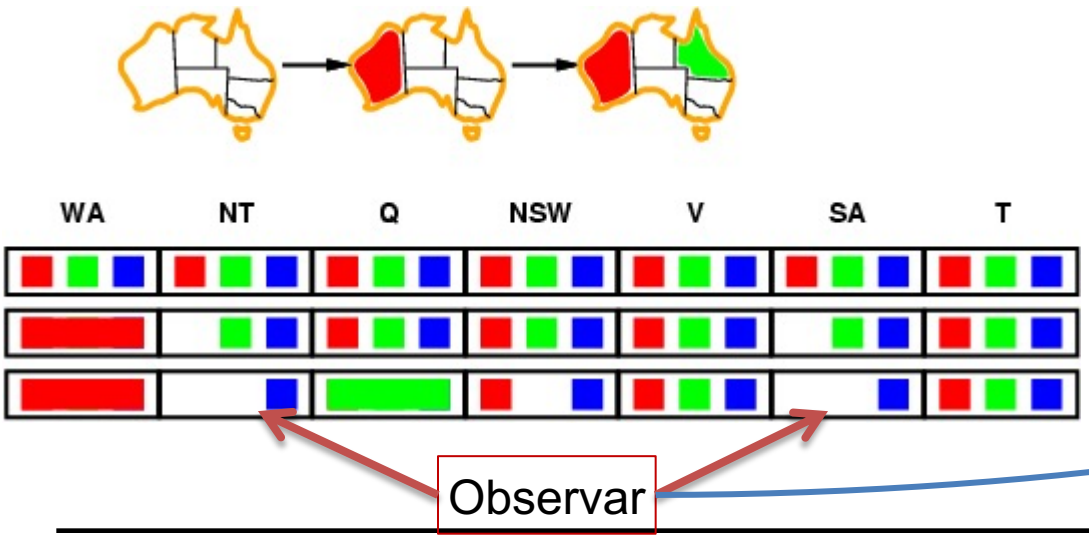
- Modificar los dominios para mantener sólo los valores legales para variables no asignadas.
- Si selecciono y asigno X, compruebo variables relacionadas con X en el grafo de restricciones, y elimino valores no legales.
- Backtracking cuando alguna variable tiene un dominio vacío.

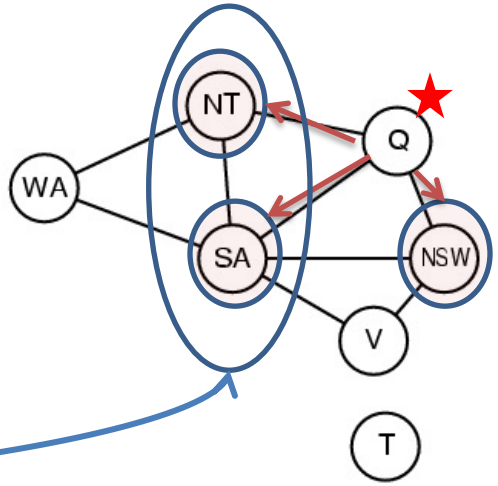


Dominio Vacío!!!

Propagación de restricciones (Constraint propagation)

- La comprobación hacia delante no descubre todas las inconsistencias
- Por ejemplo, en un paso anterior:
  - después de asignar WA=red y Q=green,
  - tanto NT como SA están obligadas a tomar el valor blue
  - pero como son adyacentes violan las restricciones
- ¿Cómo podría detectarse?





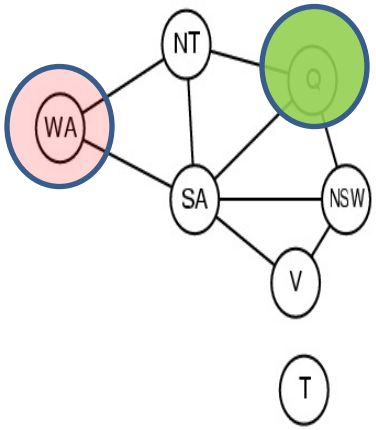
Propagación de restricciones

- Arco consistencia.
  - X es arco consistente con Y sii
    - **para todo** valor x_i hay **alguno** y_i permitido,
    - Arco consistencia ≡ Consistencia sobre la restricción binaria que representa un arco.
    - Ojo: X arco consistente Y **no implica** Y arco consistente X
  - Grafo de restricciones arco consistente
    - Si todas su variables son arco consistentes, es decir, si los dominios de las variables satisfacen todas las restricciones binarias.

Propagación de restricciones

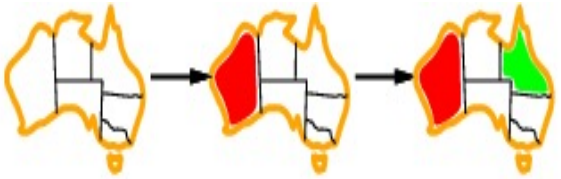
- Idea : Mantenimiento de Arco Consistencia (MAC)
  - Cada vez que se asigna una variable
  - sobre **todas las variables no asignadas**,
    - se **comprueba** arco-consistencia y se **fuerza** a que sean arco-consistentes.
      - Estrechando y actualizando los dominios
      - Si algún dominio es vacío -> Backtracking.

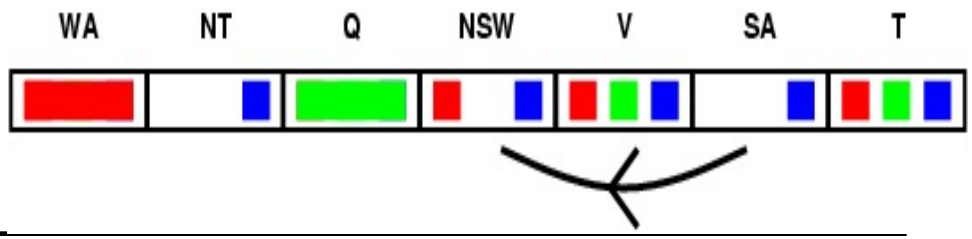
Propagación de restricciones



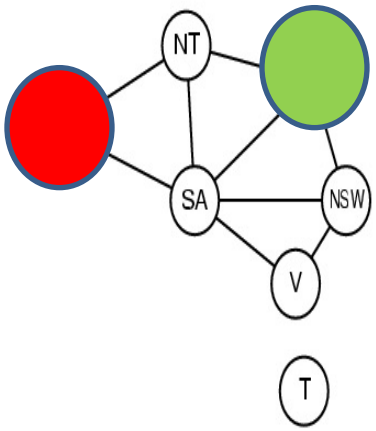
X → Y	
SA	NSW
NSW	SA
NSW	V
V	NSW
SA	NT
NT	SA

- Mantiene una cola de arcos dirigidos, para todas las variables no asignadas.
- Recorre la cola y comprueba/fuerza arco consistencia para cada arco
  - ¿Hay algún valor en NSW consistente para cada valor de SA?



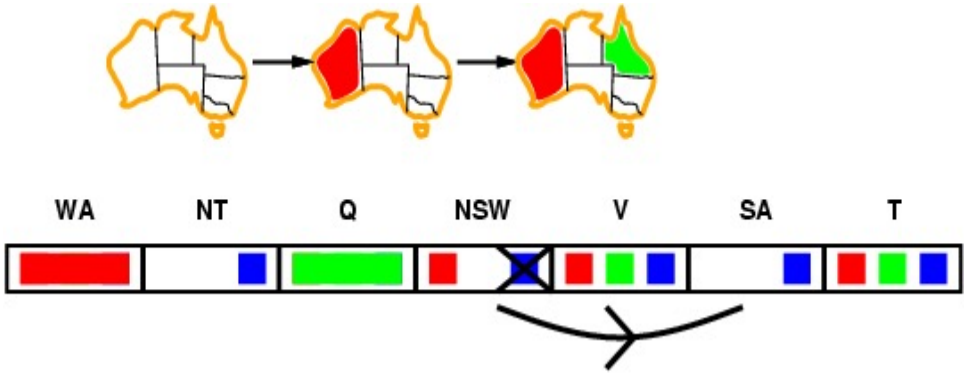


Propagación de restricciones

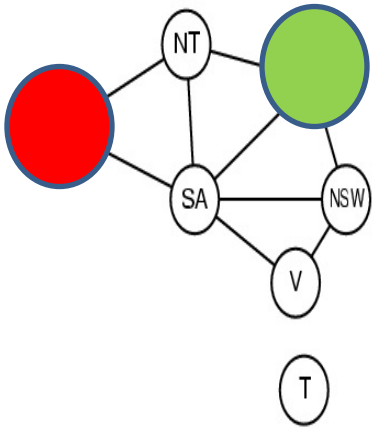


X →	Y
NSW	SA
NSW	V
V	NSW
SA	NT
NT	SA

- Mantiene una cola de arcos dirigidos, para todas las variables no asignadas.
- Recorre la cola y comprueba/fuerza arco consistencia para cada arco
  - ¿Hay algún valor en SA consistente para cada valor de NSW?
    - Sí, para red.
    - No, para blue

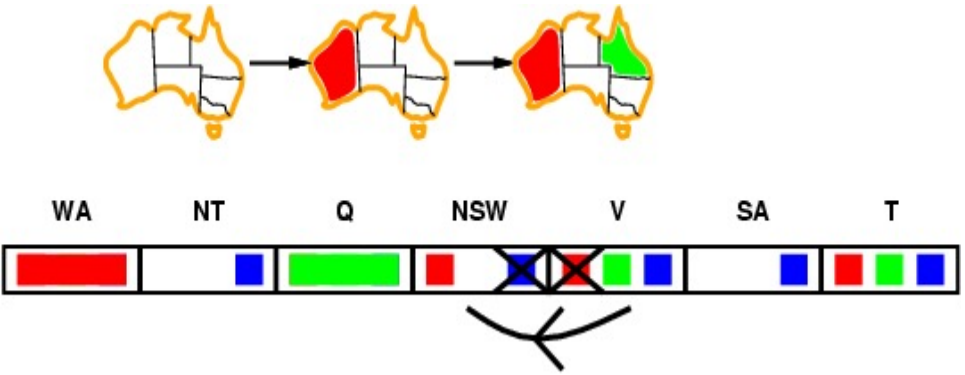


Propagación de restricciones

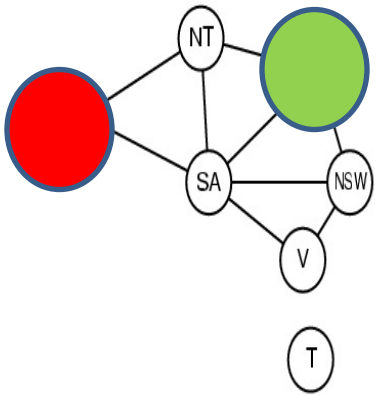


X → Y	
NSW	V
V	NSW
SA	NT
NT	SA

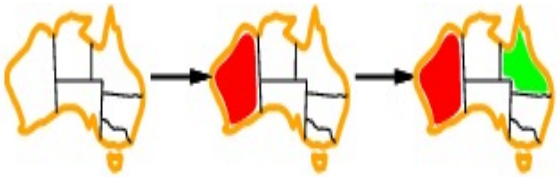
- Mantiene una cola de arcos dirigidos, para todas las variables no asignadas.
- Recorre la cola y comprueba/fuerza arco consistencia para cada arco
  - ¿Hay algún valor en V consistente para cada valor de NSW?
    - Sí, para todos
  - ¿Hay un valor en NSW para cada valor de V?
    - No para *red, se elimina*.

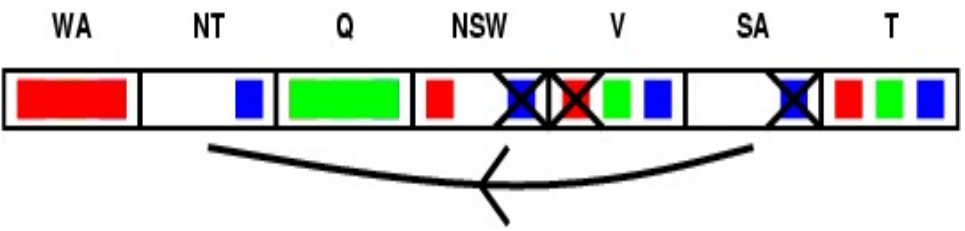


Propagación de restricciones



X →	Y
SA	NT
NT	SA

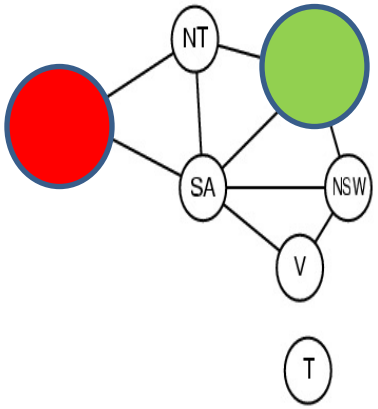




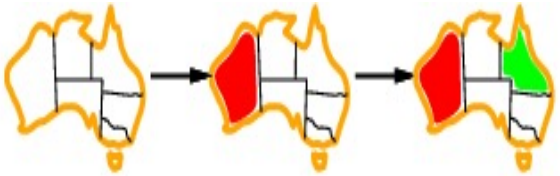
- Mantiene una cola de arcos dirigidos, para todas las variables no asignadas.
- Recorre la cola y comprueba/fuerza arco consistencia para cada arco
  - ¿Hay algún valor en NT para cada valor de SA?
    - No, para blue.
    - SA con dominio vacío.
- **Backtracking**
  - Hay que revisar la asignación

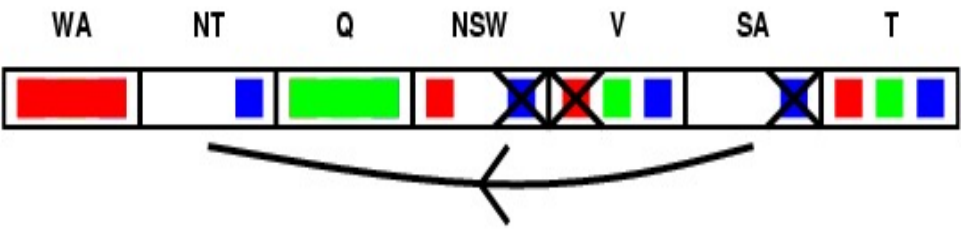
  {Q = green} y probar otra.

Propagación de restricciones



X →	Y
SA	NT
NT	SA





- Propagación de restricciones basada en arco-consistencia detecta inconsistencias antes que Forward Checking.
- Puede ejecutarse como preprocesamiento antes de empezar la búsqueda.
  - La cola inicial contiene todos los arcos del grafo de restricciones.

Vuelta atrás inteligente

- Vuelta atrás cronológica: se visita el punto de decisión más reciente

Orden de actuación: Q, NSW, V, T, SA, WA, TN

{Q=rojo, NSW=verde, V=azul, T=rojo)

SA ?

vuelta atrás a T inútil

Vuelta atrás inteligente: salto atrás

- Conjunto conflicto para la variable X es el conjunto de variables previamente asignadas Y, tales que el valor asignado a Y evita uno de los valores de X, debido a la restricción entre ambas.
- El método de salto-atrás retrocede a la variable más reciente en el conjunto conflicto.

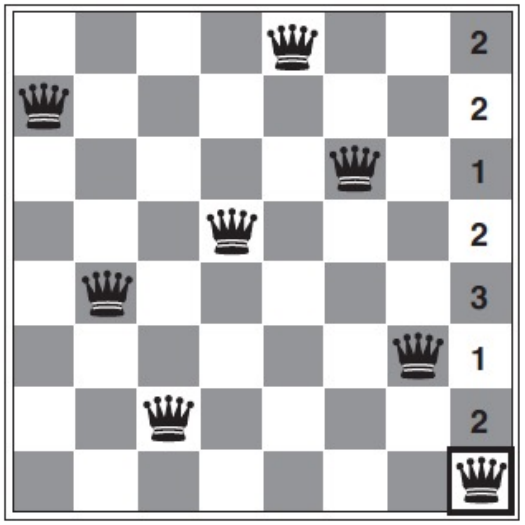
Conjunto conflicto para SA: {Q, NSW, V}

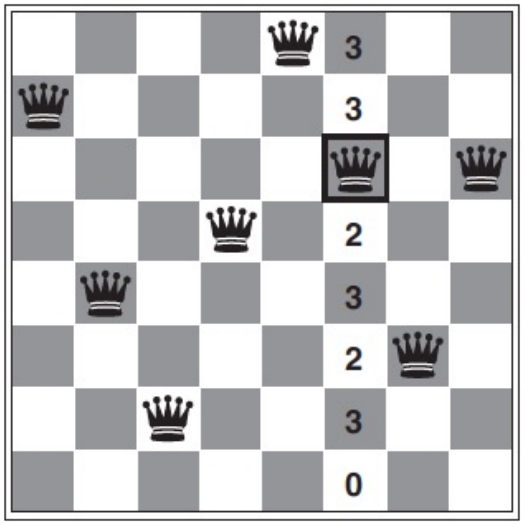
El método salto-atrás retrocede a V sin considerar T

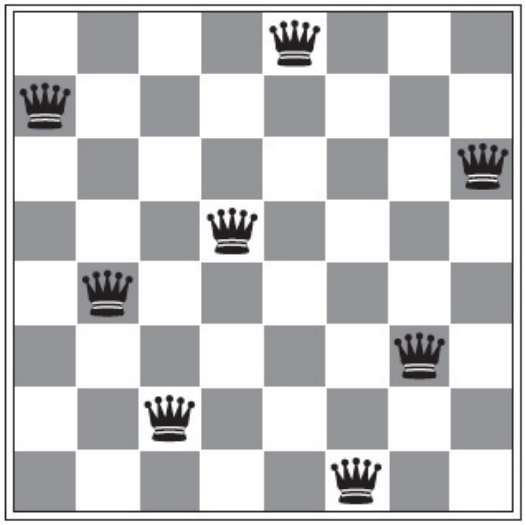
Búsqueda local para problemas de satisfacción de restricciones

- Volvemos a la formulación completa de estados para investigar el uso de algoritmos de búsqueda local sobre estados completos.
- Heurística de los mínimos conflictos: seleccionar el valor que cause el número mínimo de conflictos con otras variables.

Búsqueda local para problemas de satisfacción de restricciones







Búsqueda local para problemas de satisfacción de restricciones

- La técnica de los mínimos conflictos es sorprendentemente eficaz para muchos CSPs, en particular, cuando se parte de un estado inicial razonable.
- Por ejemplo, se ha utilizado para programar las observaciones del telescopio Hubble, reduciendo el tiempo utilizado para programar una semana de observaciones, de tres semanas a alrededor de 10 minutos.
- Otra ventaja es que se puede utilizar en ajuste online cuando las condiciones del problema cambian.