

Tema 4. Búsqueda para problemas de satisfacción de restricciones

T4. Búsqueda para problemas de satisfacción de restricciones

Satisfacción de restricciones

- Formulación de CSPs como redes de restricciones
- Ejemplos
- Métodos de resolución:
 - Esquema backtracking
 - Esquema Forward Checking
 - Esquema de propagación de restricciones

Problemas de satisfacción de restricciones (CSP)

Conjunto de **variables** definidas sobre **dominios** finitos y conjunto de **restricciones** definidas sobre subconjuntos de dichas variables.

$$(V, D, \rho)$$

→ un conjunto de **variables**

$$V = \{V_1, V_2, \dots, V_n\} \quad V = \{V_i\}_{i=1..n}$$

→ definidas sobre **dominios** discretos D_i (conjunto finito de posibles valores)

$$D = \{D_1, D_2, \dots, D_n\} \quad D = \{D_j\}_{j=1..n}$$

→ un conjunto de **restricciones** definidas sobre subconjuntos de dichas variables

$$\rho = \{\rho_1, \rho_2, \dots, \rho_n\} \quad \rho = \{\rho_k\}_{k=1..n}$$

Problemas de satisfacción de restricciones (CSP)

Ejemplo:

$$X::\{1,2\}, Y::\{1,2\}, Z::\{1,2\}$$

$$X = Y, X \neq Z, Y > Z$$

Solución: encontrar asignaciones de valor a las variables que satisfagan todas las restricciones.

$$(X = 2, Y = 2, Z = 1)$$

Solución al problema: la relación n-aria que satisface todas las restricciones del problema

Dependiendo de los requerimientos del problema hay que encontrar todas las soluciones o sólo una

Redes de restricciones

- Un CSP se puede representar como un grafo.
- Sobre el grafo se puede definir una red de restricciones:

Quíntupla $\langle V, E, c, l, a \rangle$

- V : conjunto de nodos.
- E : conjunto de aristas.
- $c: E \rightarrow V^k$, $k \leq n$; función de asignación de aristas a tuplas de nodos.
- $l: E \rightarrow \rho$; función de asignación de aristas a restricciones.
- a : permutación que define el orden de selección para resolver el problema.

CSP binario

- Variables
- $$V = \{V_1, V_2, \dots, V_n\}$$
- Dominios discretos y finitos

$$D = \{D_1, D_2, \dots, D_n\}$$

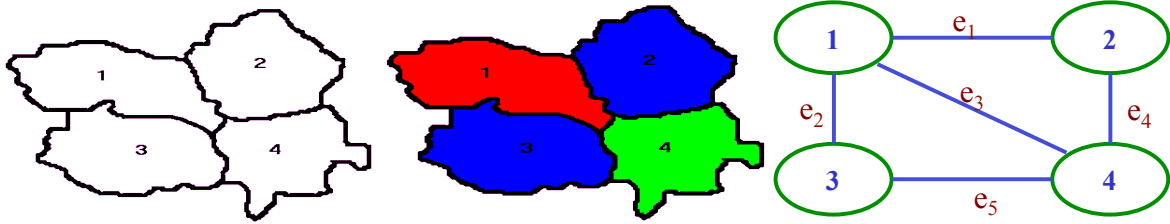
- Restricciones binarias

$$\{R_{ij}\}$$

Todo problema **n-ario** se puede formular como un problema **binario**

- Ejemplos de CSP binarios:
 - Coloreado de mapas
 - Asignación de tareas para un robot
 - N-reinas
 - Generación de crucigramas

Ejemplo: Coloreado de mapas



$$V = \{V_1, V_2, V_3, V_4\}$$

$$D_i = \{\text{rojo}, \text{azul}, \text{verde}\}, \forall i, 1 \leq i \leq 4$$

$$E = \{e_1, e_2, e_3, e_4, e_5\}$$

$$\rho_k(V_i, V_j) = \{ \langle v_i, v_j \rangle \mid v_i \in D_i, v_j \in D_j, v_i \neq v_j \}, \forall k, 1 \leq k \leq 5$$

$$c(e_1) = \langle V_1, V_2 \rangle, c(e_2) = \langle V_1, V_3 \rangle, c(e_3) = \langle V_1, V_4 \rangle, \dots$$

$$l(e_j) = \{ \langle \text{azul}, \text{verde} \rangle, \langle \text{azul}, \text{rojo} \rangle, \langle \text{verde}, \text{azul} \rangle, \langle \text{verde}, \text{rojo} \rangle, \langle \text{rojo}, \text{azul} \rangle, \langle \text{rojo}, \text{verde} \rangle \} \forall j, 1 \leq j \leq 5$$

$$a = \{V_1, V_4, V_2, V_3\}$$

Generación de crucigramas

- Dada una rejilla y un diccionario, construir un crucigrama legal

Slot horizontal de tres letras

C O L

O

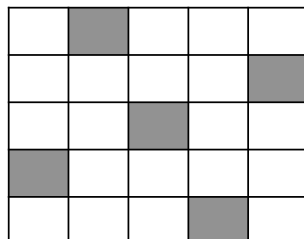
Z

4 palabras de 1 letra
4 palabras de 3 letras
2 palabras de 5 letras

- Formulación:
 - variables** : grupo de casillas para una palabra (*slots*)
 - dominios** : palabras del diccionario con la longitud adecuada
 - restricciones** : misma letra en la intersección de dos palabras
- Características :
 - CSP binario, discreto (dominios grandes)

N-reinas

- Posicionar n reinas en un tablero de ajedrez $n \times n$, de forma que no se ataquen



$n = 5$

- Formulación: (1 reina por fila)
 - variables** : reinas, X_i reina en la fila i -ésima
 - dominios** : columnas posibles $\{1, 2, \dots, n\}$
 - restricciones** : no colocar dos reinas en
 - la misma columna
 - la misma diagonal
- Características :
 - Dominios discretos y restricciones binarias

T4. Búsqueda para problemas de satisfacción de restricciones

9

Criptoaritmética

- Sustituir cada letra por un dígito distinto (distinta cifra, distinta letra) de manera que la suma sea correcta
- Formulación:

- variables** : G, O, T, A, U, C_1 , C_2 , C_3
(C_1 , C_2 , C_3 variables de acarreo)
 - dominios** : O, T, U $\in \{0, \dots, 9\}$
G, A $\in \{1, \dots, 9\}$
 $C_1, C_2, C_3 \in \{0, \dots, 4\}$
 - restricciones** :
 - letras distintas G \neq O, G \neq T, ..., A \neq U
 - suma correcta

(*unidades*)

(*decenas*)

(*centenas*)

(*unidades millar*)

$5 \cdot A = 10 \cdot C_1 + A$

$5 \cdot T + C_1 = 10 \cdot C_2 + U$

$5 \cdot O + C_2 = 10 \cdot C_3 + G$

$5 \cdot G + C_3 = A$

GOTA

GOTA

+

GOTA

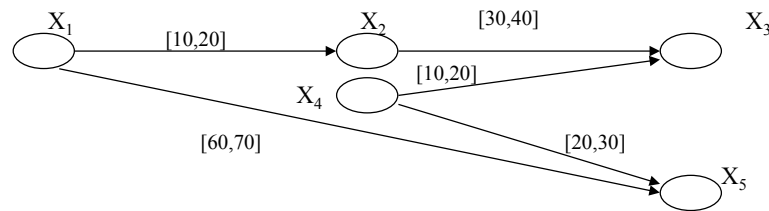
GOTA

GOTA

AGUA
- Características :
 - Dominios discretos y restricciones múltiples

Restricciones temporales

- Dado un conjunto de sucesos que ocurren en intervalos temporales con ciertas relaciones, encontrar un asignación temporal consistente



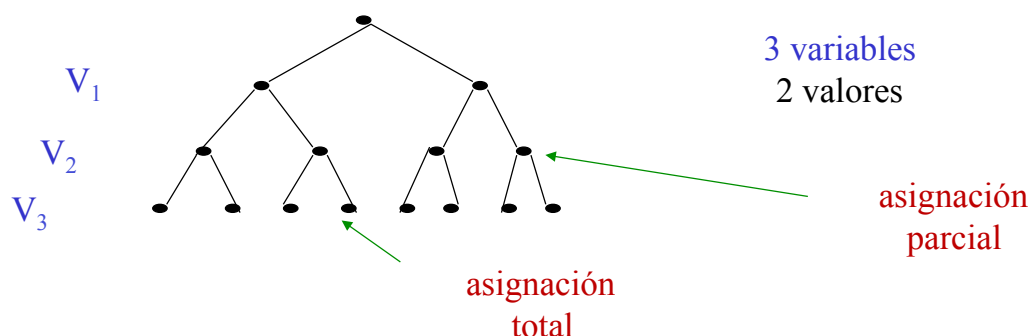
- Formulación:
 - variables** : sucesos
 - dominios** : intervalo temporal para cada suceso
 - restricciones** : distancia temporal permitida entre sucesos; relaciones temporales antes, después, solapado, etc.
- Características :
 - Dominios continuos y restricciones binarias

T4. Búsqueda para problemas de satisfacción de restricciones

11

Arbol de interpretaciones

- Partimos de un nodo raíz que supervisa el proceso.
- Cada nivel corresponde a una asignación de valor para una característica de datos. El orden de descenso viene especificado por a .
- Cada nodo identifica una posibilidad de asignación (Variable, valor).
- La solución se construye de forma incremental de tal forma que cada hoja es una interpretación.



T4. Búsqueda para problemas de satisfacción de restricciones

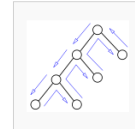
12

Métodos de resolución

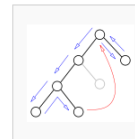
Búsqueda

Generación y test : generar de forma sistemática y exhaustiva cada una de las posibles asignaciones a las variables y comprobar si satisfacen todas las restricciones. Hay que explorar el espacio definido por el producto cartesiano de los dominios de las variables.

Backtracking : se trata de construir la solución de forma gradual, instanciando variables en el orden definido por la permutación dada



Backjumping: parecido al BT pero el retroceso no se hace a la variable instanciada anteriormente sino a la variable más profunda que está en conflicto con la variable actual.



Explorar el espacio de estados hasta encontrar una solución, demostrar que no existe o agotar los recursos

Métodos de resolución

Inferencia

- **Consistencia de arco**
- Consistencia de caminos
- K-consistencia

Deducir un problema equivalente que sea más fácil de resolver

Algoritmos híbridos

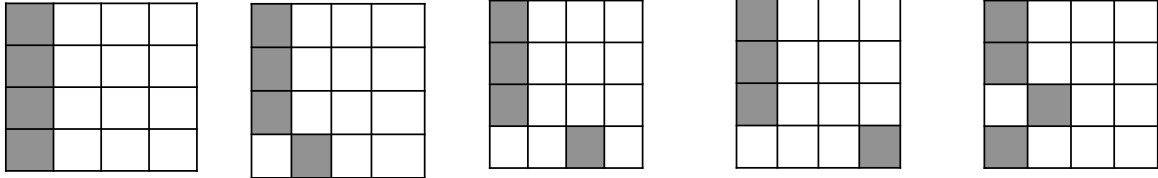
- **Forward Checking**
- Maintaining Arc Consistency
- Heurísticas

Combinación de las aproximaciones anteriores. Sobre un esquema de búsqueda se incorporan métodos de inferencia

Generación y test

Estrategia:

- generación y test de todas las asignaciones totales posibles
 1. Generar una asignación de todas las variables
 2. Comprobar si es solución. Si es, stop, sino ir a 1



Eficiencia:

- es muy poco eficiente
- genera muchas asignaciones que violan la misma restricción

Backtracking

Estrategia:

- Construir una solución parcial : asignación parcial que satisface las restricciones de las variables involucradas
- Extender la solución parcial, incluyendo una variable cada vez hasta llegar una solución total
- Si no se puede extender: backtracking
 - cronológico: se elimina la última decisión
 - no cronológico : se elimina una decisión anterior

Backtracking recursivo

funcion bt (x variable): booleano

Para todo $a \in D(x)$ hacer

$x \leftarrow a$

si test (x, pasadas)

pasadas \leftarrow pasadas + {x}

si bt(siguiente(x)) retorna CIERTO

sino pasadas \leftarrow pasadas - {x}

retorna FALSO

pasadas: variables \in
solución parcial

funcion test (x variable, P conjunto): booleano

para todo $y \in P$ hacer

si $(val(x), val(y)) \notin rel(R_{xy})$ retorna FALSO

retorna CIERTO

la función test comprueba que el valor de la variable actual (x) sea consistente con la solución parcial

Limitaciones del backtracking

- **Trashing e inconsistencia de nodo**

Relacionado con las **restricciones unarias**. Sucede cuando un dominio contiene un valor que no satisface una restricción unaria.

- **Inconsistencia de arista**

Relacionado con las **restricciones binarias**. Sucede cuando existe una restricción binaria entre dos variables de tal forma que para un determinado valor de la primera variable no existe ninguna asignación posible para la segunda.

- **Dependencia de la ordenación**

El orden de selección de las variables es un factor crítico. Se han desarrollado diversas heurísticas de selección de variable y de valor.

Variable: Orden estático y Orden dinámico

Valor: p.e. los que conducen a un CPS más simple

Forward checking

- En cada etapa de la búsqueda, FC comprueba hacia delante la asignación actual con todos los valores de las futuras variables que están restringidas con la variable actual.
- Los valores de las variables futuras que son inconsistentes con la asignación actual son temporalmente eliminados de sus dominios.
- Si el dominio de una variable futura se queda vacío, la instanciación de la variable actual se deshace y se prueba con un nuevo valor. Si ningún valor es consistente, entonces se lleva a cabo el backtracking cronológico.

Forward checking

```

funcion FC(i variable): booleano
  para cada a ∈ factibles[i] hacer
    Xi ← a
    si i=N solución retorna CIERTO
    sino
      si forward (i,a)
        si FC(i+1) retorna CIERTO
      Restaurar (i)
  retorna FALSO

funcion forward(i variable, a valor): booleano
  para toda j=i+1 hasta N hacer
    Vacio ← CIERTO
    para cada b ∈ factibles[j] hacer
      si (a,b) ∈ Rij vacio ← FALSO
      sino eliminar b de factible[j]
      Añadir b a podado[j]
    si vacio retorna FALSO
  retorna CIERTO

procedimiento restaura(i variable)
  para toda j=i+1 hasta N hacer
    para todo b ∈ podado[j] hacer
      si Xi responsable filtrado b
        Eliminar b de podado[j]
        Añadir b a factible[j]
  
```

Forward Checking

Ejemplo Forward Checking:

Variables x, y

$D_x = D_y = \{1, 2, 3, 4, 5\}$

Restricción $x < y - 1$

Inicialmente $CD_x = CD_y = \{1, 2, 3, 4, 5\}$

Si asignamos $x = 2$, entonces:

Los únicos valores que puede tomar y son 4, 5
por tanto $CD_y = \{4, 5\}$

..
..

Si asignamos $x = 4$, entonces:

No hay asignación posible compatible con la restricción.
por tanto $CD_y = \{\}$
Deshacer $x = 4$ y backtracking

Propagación de Restricciones

- Transformar el problema en otro más sencillo sin inconsistencias de arco.
- Propiedad de consistencia de arista
Una arista dirigida $c(e_p) = \langle V_i, V_j \rangle$ es consistente si y sólo si para todo valor asignable a V_i existe al menos un valor en V_j que satisface la restricción asociada a la arista.
- Un CSP puede transformarse en una red consistente mediante un algoritmo sencillo (AC3) que examina las aristas, eliminando los valores que causan inconsistencia del dominio de cada variable.
- Después del proceso:
 - No hay solución
 - Hay más de una solución
 - Hay una única solución

Algoritmo AC3

$Q = \{c(e_p) = \langle V_i, V_j \rangle | e_p \in E, i \neq j\}$

Mientras $Q \neq \emptyset$ hacer

$\langle V_k, V_m \rangle = \text{seleccionar_y_borrar}(Q)$

cambio = falso

Para todo $v_k \in D_k$ hacer

Si no_consistente(v_k, D_m) entonces

borrar(v_k, D_k)

cambio = cierto

FinSi

FinPara

Si $D_k = \emptyset$ entonces salir_sin_solución FinSi

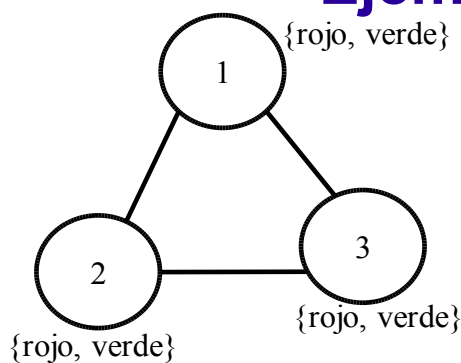
Si cambio = cierto entonces

$Q = Q \setminus \{c(e_r) = \langle V_i, V_k \rangle | e_r \in E, i \neq k, i \neq m\}$

FinSi

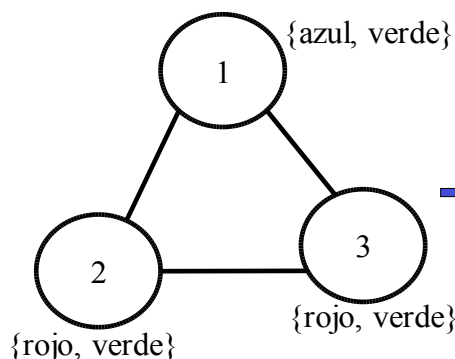
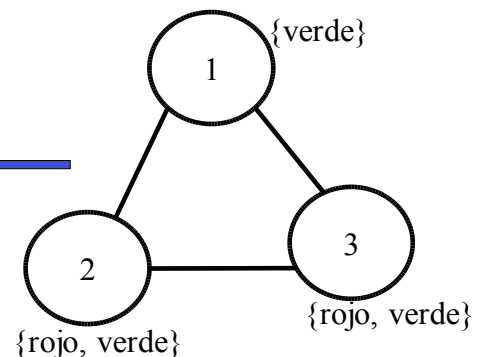
FinMientras

Ejemplos de grafos



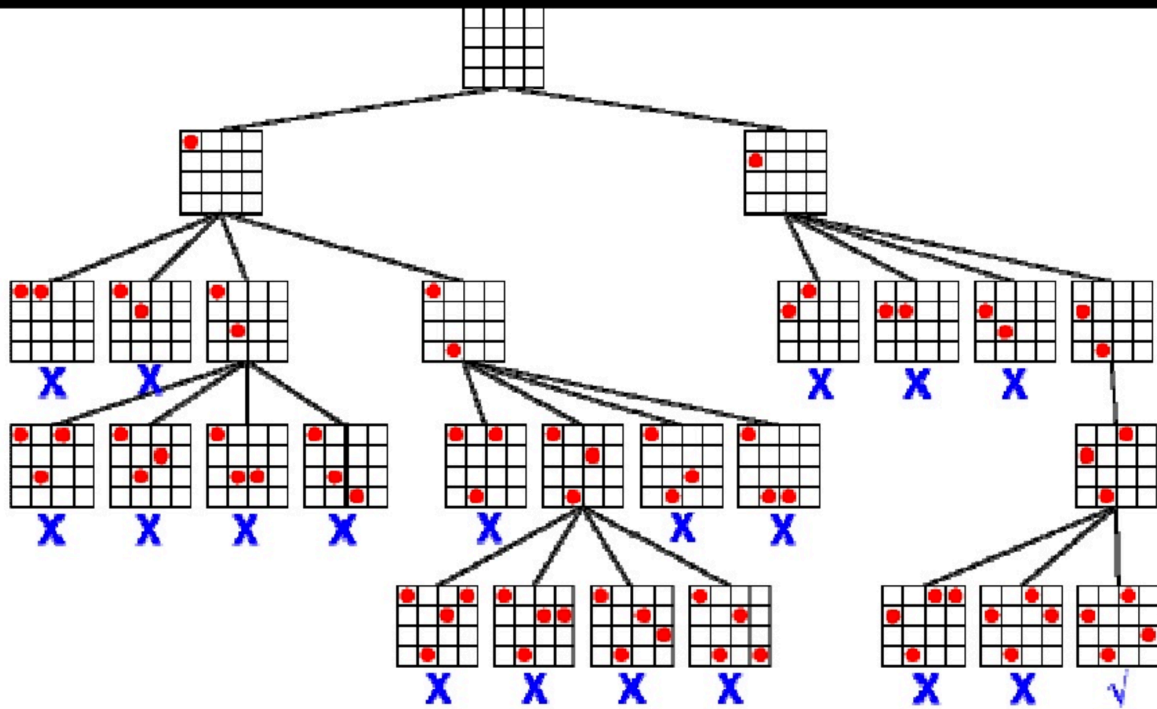
Consistente, sin solución

Inconsistente, sin solución

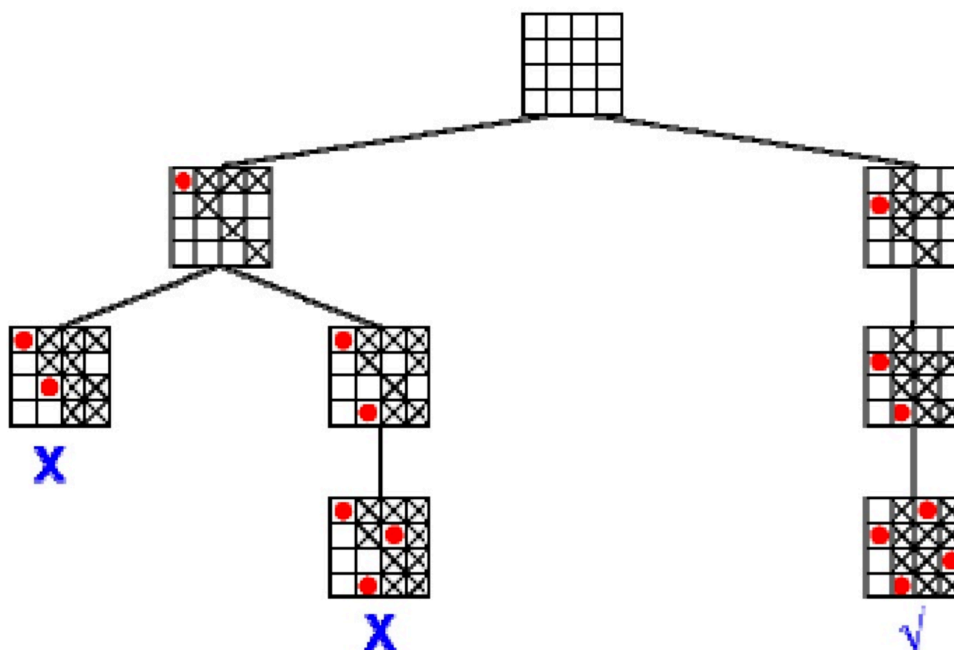


Consistente, con dos soluciones

Algoritmo AC-3 para backtracking ejemplo



Algoritmo AC-3 para Forward checking: ejemplo



Tema 2. Estrategias de búsqueda.

Bibliografía

- Stuart Russell, Peter Norving. “Inteligencia Artificial. Un enfoque Moderno” Ed. Pearson. Prentice Hall. 2004.