Universidad de Huelva

Universidad de Huelva

Grado en Ingeniería Informática

Tema 6. Programación con Restricciones

Resumen

Autor: Alberto Fernández Merchán
Asignatura: Sistemas Inteligentes

1. Definición

Un problema de satisfacción de restricciones (CSP) está formado por:

- Conjunto de Variables
- Por cada variable, un conjunto de valores llamado Dominio.
- Conjunto de Restricciones.

El objetivo de este tipo de problemas es encontrar un valor para cada variable de manera que satisfagan todas las restricciones del problema. Se trata de encontrar tuplas de valores $(v_1, \ldots, v_n)$ de las variables $(X_1, \ldots, X_n)$ que satisfacen las restricciones.

Los problemas CSP pueden ser:

- **CSP discreto.** Todas las variables son discretas.
- **CSP continuo.** Todas las variables son continuas.
- **CSP mixto.** Consta de variables continuas y discretas.
- **CSP binario.** Todas las restricciones tienen dos variables.
- **CSP n-ario.** Las restricciones tienen cualquier número de variables.

2. Restricciones

Cada restricción limita el conjunto de asignaciones para las variables implicadas. Pueden darse explícita o implícitamente. Las restricciones pueden ser de los siguientes tipos:

- **Discretas**. Las variables tienen dominios discretos.
- **Continuas**. Las variables tienen dominios continuos.
- **Binarias**. Participan dos variables.
- **N-Arias**. Participan N variables. ($N > 2$)
- **Fuertes**. Satisfabilidad es imprescindible.
- **Débiles**. Satisfabilidad es prescindible.
- **Difusas**. Definidas sobre niveles de preferencias.
- **Disyuntivas**. Compuestas por un conjunto disjunto de restricciones.

3. Resolución de Problemas

La expresividad del problema viene definida por la etapa del modelado y la eficiencia del problema viene definida por la etapa de resolución. Ambas características son inversamente proporcionales entre sí.

3.1. Modelado

Es lo primero que hay que realizar. Para ello debemos encontrar los siguientes componentes:

- Variables. (Se puede elegir).
- Dominio. (Se puede elegir).
- Restricciones.(Nos las da el problema).

3.2. Procesamiento

3.2.1. Reducción de problemas: Consistencia

Para resolver el problema usaremos una etapa de consistencia. El objetivo de dicha etapa será evitar poner restricciones que sabemos de antemano que no van a ocurrir.

3.2.2. Algoritmos de Búsqueda

- **En el espacio de Estados**: Los **estados** se definen como asignaciones parciales de valores a variables y consistentes con las restricciones. El **estado inicial** es tener las variables sin ninguna asignación. El **estado final** será tener todas las asignaciones completas. El operador será: **asignar un valor a una variable no asignada**. Puesto que las soluciones se encuentran a una profundidad fija, es preferible hacer una búsqueda en profundidad.

  Simplificaciones El orden de aplicación de los operadores es irrelevante para la solución final. No es posible repetir un estado durante la búsqueda y, como consecuencia, no es necesario realizar comprobaciones para evitar repeticiones (No es necesaria una lista de estados cerrados).

```
FUNCION PSR-BUSQUEDA-EN-PROFUNDIDAD()
    ABIERTOS  = {AsigVacia}
    Mientras que ABIERTOS no esté vacía,
      ACTUAL = primero(ABIERTOS)
      Si ACTUAL es una asignación completa
        devolver ACTUAL y terminar.
      en caso contrario,
        NUEVOS-SUCESORES = SUCESORES(ACTUAL)
        ABIERTOS = NUEVOS-SUCESORES + ABIERTOS
3. Devolver FALLO.
```

Figura 1: Pseudocódigo Busqueda en Profundidad con restricciones

```
FUNCION SUCESORES():
     VARIABLE = SELECCIONA-VARIABLE(ESTADO)
     DOMINIO-ORD = ORDENA-VALORES(PSR-VAR-DOMINIO(VARIABLE))
     SUCESORES = []
     Para cada VALOR en DOMINIO-ORD {
        NUEVO-ESTADO = ESTADO +  {VARIABLE=VALOR}
        Si NUEVO-ESTADO es consistente con *RESTRICCIONES*,
           añadir NUEVO-ESTADO a SUCESORES
     Devolver SUCESORES
```

Figura 2: Cálculo de Sucesores

- **Genera y Comprueba**: Consiste en asignar un valor del dominio permitido a cada una de las variables de la tupla a evaluar. Después se comprueba si cumplen todas las restricciones. Si no se verifica, entonces se desecha la tupla y se genera la siguiente. Este proceso se repite hasta encontrar una solución válida o hasta que no haya más casos para generar.

- **Backtracking Simple**: El algoritmo de búsqueda anterior se suele llamar algoritmo de backtracking, aunque se suele representar de manera recursiva: Se asigna un valor del dominio permitido a la siguiente

```
FUNCION PSR-BACKTRACKING()
   Devolver PSR-BACKTRACKING-REC({})
FUNCION PSR-BACKTRACKING-REC(ESTADO)
   Si ESTADO es una asignación completa, devolver ESTADO y terminar
   VARIABLE = SELECCIONA-VARIABLE(ESTADO)
   DOMINIO-ORD = ORDENA-VALORES(CSP-VAR-DOMINIO(VARIABLE))
   Para cada VALOR en DOMINIO-ORD,
      NUEVO-ESTADO = ESTADO + {VARIABLE=VALOR}
      Si NUEVO-ESTADO es consistente con *RESTRICCIONES*:
         RESULTADO = PSR-BACKTRACKING-REC(NUEVO-ESTADO)
         Si RESULTADO no es FALLO, devolver RESULTADO y terminar
   Devolver FALLO
```

Figura 3: Búsqueda con Backtracking simple

variable a evaluar y se comprueba si forma parte de la solución parcial. Si no verifica las restricciones, entonces se toma el siguiente valor del dominio de la variable. Si ningún valor forma parte de la solución, entonces elimina ese valor de la variable anterior evaluada y se toma el siguiente. Esto se repite hasta encontrar una solución o que no haya más casos posibles.

Solo comprueba las restricciones entre la nueva variable y las ya evaluadas.

- **Backtracking con Chequeo Previo (LookForward)** Se asigna un valor de su dominio a la nueva variable a evaluar y se comprueba si forma parte de la solución parcial y, además, se comprueba si algún valor de una futura asignación tiene conflicto con este valor. Si es así se elimina temporalmente del dominio de esa futura variable.

  Si el dominio de una futura variable llega a ser vacío, significaría que la solución parcial es inconsistente y se debe comprobar otro valor. Con este algoritmo de búsqueda las ramas de los árboles que dan inconsistencias son podadas con anterioridad.

4. Heurísticas

Es posible dotar al algoritmo de Backtracking con cierta heurística que mejora su rendimiento. Las posibles mejoras son las siguientes:

- **Selección de nueva variable a asignar**. Esta idea consiste en clasificar las variables de las más restringidas a las que menos lo son. (Orden creciente a la talla de los dominios).
- **El orden de asignación de los valores a las variables**. La más común es el **first-fail**, es decir, se deben probar primero los valores con los que es más probable que falle.
- **Propagación de información a través de restricciones**.
- **Forward Checking**. Por cada estado mantiene información sobre posibles valores para las variables que quedan por asignar. Cada nodo del árbol debe contener el estado junto a la lista de valores posibles en las variables por asignar.

5. Técnicas de Consistencia

Son formas de mejorar la eficiencia de los algoritmos de búsqueda ayudando a podar el espacio de búsqueda eliminando valores inconsistentes de las variables. Se utilizan en etapas de preprocesamiento donde se detectan y eliminan las inconsistencias antes de empezar la búsqueda.

5.1. Arcos

Un arco es una restricción donde hay una variable distinguida. Un arco es consistente respecto a un conjunto de dominios asociados a un conjunto de variables si para cualquier estado del dominio asociado a la variable distinguida del arco, existe algún valor en los dominios de las variables restantes que satisfagan la restricción del arco.

5.2. Algoritmo Arc Consistency 3 (AC3)

El objetivo del algoritmo es devolver un conjunto de dominios actualizado tal que todos los arcos del problema sean consistentes a partir de los dominios iniciales de las variables.

Si un arco es inconsistente, podemos hacerlo consistente eliminando del dominio de la variable distinguida aquellos valores para los que no existen valores en los dominios de las restantes variables que satisfagan la restricción.

Si el dominio de una variable se actualiza, es necesario revisar la consistencia de los arcos en los que aparece la variable como variable no distinguida.

El algoritmo se detiene cuando todos los arcos son consistentes respecto a los dominios de las variables o cuando algún dominio queda vacío.