Tema 2 - Problemas de Satisfacción de Restricciones

Profesor Antonio González Muñoz, Curso 20-21

- Es un tipo particular de problemas que van a tener asociado
  - Métodos de búsqueda particulares
  - Heurísticas de propósito general que sirven en general para estos problemas.
- Un problema de este tipo se denomina CSP y está definido por X, D y C.
  - X es el conjunto de las variables $\{X_1, X_2, \dots, X_n\}$
  - D es el conjunto de dominios $\{D_1, D_2, \dots, D_n\}$, uno por cada variable.
  - C es el conjunto de restricciones $\{C_1, C_2, \dots, C_m\}$ que especifican las combinaciones permitidas de valores.
- Cada dominio D_i representa un conjunto $\{V_1, \dots, V_k\}$ de valores permitidos para X_i, y cada restricción $C_i =$ <lista de variables y relación entre ellas>

Satisfacción de Restricciones

- El estado de un problema es una asignación de valores a algunas variables o a todas.
- Para tener un lenguaje que indique que clase de solución se está buscando, se tiene…
  - Asignaciones **consistentes**: Una asignación que no viola ninguna restricción, es decir, que los valores que se les asigna a las variables no están rompiendo ninguna restricción que se tienen en el problema.
  - Asignaciones **completas**: Una asignación en la que aparecen todas las variables.
  - Asignación **parciales**: Una asignación en la que aparecen algunas de las variables.
- Lo que se busca para un problema de esta clase es una asignación completa y consistente.
- Algunos CSPs requieren una solución que maximice o minimice una función objetivo
  - **COP**: Constraint Optimization Problems, problemas de optimización de restricciones. Un subgrupo de los CSPs más complejos por la restricción de optimizar.
- Las restricciones entre las variables se pueden representar como un grafo de restricciones, en donde los nodos son las variables y los arcos entre variables son las restricciones del problema que tengan que ver los dos nodos.

Tipos de CSPs

- Según el tipo de variables
  - Variables Discretas (Enteros, Símbolos, Cadenas)
    - Dominios Finitos
      - Tienen n variables, el tamaño del dominio es $d \rightarrow O(d^n)$ asignaciones completas. Es la más simple y aún así, tiene un crecimiento exponencial.
      - Ejemplos: Diseño de circuitos, demostración de teoremas, verificación y validación de software.
    - Dominios Infinitos
      - Utilizan enteros y cadenas con un dominio infinito, es necesario tener un lenguaje de restricciones.
      - Ejemplos: Asignación de tareas y máquinas.
  - Variables Continuas (Números reales)
    - Tienen restricciones lineales
    - Por ejemplo, tiempo de inicio y de fin de las observaciones del Telescopio Espacial Hubble.
- Según su tipo de restricciones
  - Restricciones Unarias, involucran una sola variable.
    - $SA \neq Green$
  - Restricciones Binarias que involucran parejas de variables.
    - $SA \neq WA$
  - Restricciones de Orden Superior que involucran 3 o más variables, restricciones globales.
    - Restricciones Criptoarimtéticas de Columnas o Puzles Criptoariméticos
  - Preferencias o Restricciones suaves
    - Rojo es mejor que verde.

¿Cómo resolver estos problemas?

- **Formulación Incremental:** Se realiza una asignación paso a paso, se asigna una variable a un valor, luego otra, hasta llegar al final.
- **Formulación Completa de Estados**

Formulación Incremental

- Se comienza con un **estado inicial** de una asignación vacía, $\{\}$. Ninguna variable tiene valores.
- Se tiene una **función sucesor** en la que un valor se puede asignar a cualquier variable no asignada con la condición de que no suponga ningún conflicto con las variables que han sido anteriormente asignadas. Es un proceso más tradicional de búsqueda.
- El **test del objetivo** es que la asignación actual es completa, dado que mientras se esté ejecutando se tendrá siempre asignaciones parciales.
- El **costo del camino** es constante para cada paso en principio.
- Para n variables, la solución se encuentra a una profundidad de n. Puede tener un tamaño gigantesco pero tiene una frontera.

Formulación Completa de Estados

- Dado que el camino no interesa para la solución, se pueden manejar estados con asignaciones completas y sobre ella se resuelve con métodos de escalada para problemas que incumplan restricciones.
- Hace uso de métodos de Búsqueda Local, ya que estos métodos funcionan mejor con soluciones completas. A diferencia de la incremental, estas no tienen garantías pero son más eficientes de ejecutar.

Complejidad

- Estos problemas son extremadamente complejos, obtener soluciones es en general NP-Completo y la obtención de soluciones óptimas es NP-Duro.
- Se requiere por lo tanto gran eficiencia de los métodos de búsqueda.

Búsqueda en la Formulación Incremental

- El uso de Búsqueda en Anchura hace que se genere un árbol con $n! \times d^n$ hojas (mucho).
- Existe una propiedad crucial que es común en los CSPs: La conmutatividad, es decir, el orden de aplicación de cualquier conjunto de acciones no tiene ningún efecto sobre el resultado. El orden en el que se asignan las variables no importa, porque lo que importa es el resultado final.
- La técnica más eficiente para esta clases de problemas es la Búsqueda en Profundidad Retroactiva (Backtracking), en general las búsquedas retroactivas funcionan bien ya que tienen profundidad limitada pero no expanden tantos nodos.

Mejoras del Modelo

- Al backtracking se le puede añadir una heurística para mejorar la funcionalidad del algoritmo
  - ¿Qué variable debe asignarse después, y en qué orden deberían intentarse los 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 los siguientes nodos?
- Los problemas CSP disponen de heurísticas genéricas independientes del problema.

Variables y Ordenamiento de Valores

- **Mínimos Valores Restantes (MVR) o Variable Más Restringida**
  - En principio no hay ningún criterio en la selección de la variable, por lo que el orden en que se tienen las variables en el problema afecta mucho su rendimiento.
  - Esta heurística toma de primero las variables que poseen más restricciones frente a las que poseen menos, tiene la suerte de ser una heurística sencilla, fácil de calcular e implementar. Funciona mejor porque hay menos alternativas en las que se va a ir a un camino peor y tener que regresar.
  - En los problemas en los que se tienen la misma cantidad de restricciones, se añade más información para mitigar esto de manera que los dominios de las variables se va a ir reduciendo conforme se vayan asignando más variables.
- **Grado Heurístico**
  - Selecciona la variable, entre las que no estén asignadas, que esté implicada en el mayor número de restricciones. La MVR es una mejor heurística, pero esta funciona junto a la MVR para resolver

empates de restricciones.

- **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 menos restringe los valores del resto de variables no asignadas.
  - Se toman decisiones que eviten en lo posible que restringa las otras variables porque de este modo se mejoran las posibilidades de llegar a la solución sin realizar una vuelta atrás.

Propagación de la Información a través de las Restricciones

Para mejorar el backtracking, luego de la heurística, se considera la propagación de información por las restricciones.

- **Comprobación hacia delante (Foward-Checking)**
  - Es un algoritmo adicional que se incluye en el proceso de búsqueda.
  - Modifica los dominios para mantener solo los valores legales para variables no asignadas.
    - Si selecciono y asigno X, compruebo variables relacionadas con X en el grafo de restricciones y elimino los valores no legales.
    - Cuando se tiene un dominio vacío se realiza la vuelta atrás.
  - Aun así, este método no logra descubrir todas las inconsistencias que se tienen.
- **Propagación de Restricciones (Constraint Propagation)**
  - Es más costoso computacionalmente que la Comprobación hacia Delante pero lo que se ahorra en expandir menos nodos mejora en general el tiempo de ejecución.
  - La idea no es comparar una variable asignada con el resto, sino hacer comprobaciones de cada par de variables en el grafo de restricciones después de haber hecho una asignación, para ello se define:
    - Arco Consistencia: X es arco consistente con Y si y solo si
      - Para todo valor de x_i hay alguno y_i permitido
      - La arco consistencia es equivalente a la consistencia sobre una condición binaria que representa un arco.
      - Que X sea arco consistente con Y no implica que Y sea arco consistente con X.
  - **Grafo de restricciones completo arco consistente**: Si todas sus variables son arco consistentes, es decir, si los dominios de las variables satisfacen todas las restricciones.
  - El algoritmo para mantener la arco consistencia, se llama Mantenimiento de Arco Consistencia (**MAC**).
    - Cada vez que se asigna una variable, se realiza sobre todas las restantes variables no asignadas un chequeo de arco consistencia y se fuerza a que sean arco consistentes.
      - Se fuerza eliminando valores de los dominios
      - Si algún dominio queda vacío, se debe hacer vuelta atrás.
    - Se mantiene una cola de arcos dirigidos para todas las variables no asignadas.
    - Se recorre la cola y comprueba/fuerza arco consistencia en cada arco.
  - La propagación de restricciones basada en arco-consistencia detecta inconsistencias antes que Foward Checking, puede ejecutarse como preprocesamiento antes de empezar la búsqueda.
- **Vuelta atrás inteligente**
  - El Backtracking normal hace uso de la vuelta atrás cronológica, se vuelve al punto de decisión más reciente. Pero puede suceder que al entrar en un callejón sin salida, la decisión que llevó a esto no está un salto hacia atrás, sino mucho más atrás, por lo que hacer varias vueltas atrás es un gasto.
  - El método de vuelta atrás inteligente se llama Salto Atrás.
    - Para una variable X, su conjunto conflicto es el conjunto de variables previamente asignadas Y, tales que el valor asignado a Y evita uno de los valores de X por una restricción entre ambas.
    - El método de Salto Atrás retrocede a la variable más reciente del conjunto conflicto aunque esta esté mucho más atrás en el camino actual.

<u>Búsqueda Local para Problemas de Satisfacción de Restricciones:</u>

- Se pueden utilizar Algoritmos de Búsqueda Local utilizando la formulación completa de estados, ya que para esta clase de problemas funcionan muy bien.
- Se utiliza la heurística de mínimos conflictos, se selecciona el valor que cause el menor número de conflictos con otras variables. Es una heurística genérica y no depende del problema.
- La técnica de mínimos conflictos es sorpresivamente eficaz para muchos CSPs, en particular cuando se parte de un estado inicial razonable.

- Otra ventaja es que se puede utilizar aunque las condiciones del problema cambien mientras se está en ejecución, se puede adaptar.