GRASP - Generic seaRch Algorithm for the Satisfiability Problem

Gutiérrez García, Roberto
López Peña, José Ignacio

¿ Qué es GRASP ?

- Es un marco integrado de algoritmos para SAT
- Integra varias técnicas de búsqueda podada.
- Basado en la inevitabilidad de conflictos durante la búsqueda.
- Su característica principal es el incremento de la búsqueda de backtraking básica con un poderosos procedimiento de análisis de conflictos.

¿Porqué utilizar GRASP?

- Los problemas de satisfacibilidad (SAT) aparece en muchos contextos:

  Diseño de circuitos integrados con ayuda de computadoras,

  Análisis de sincronización,

  Pruebas de retraso,

  Verificación de lógica,

  etc.

- A pesar de estar bien estudiados y ampliamente investigados, siguen estando en el foco de interés debido a que con nuevas técnicas eficientes para su solución se puede llegar a obtener un gran impacto.
- Los problemas SAT pertenecen a la clase de problemas NP-completos cuyas soluciones suelen tener complejidad exponencial en el peor de los casos.

Conceptos Básicos

- Forma Normal Conjuntiva – CNF
  - (Cláusula) Λ (Cláusula)Λ (Cláusula)
  - Cláusula = A ∨ B ∨ C
- Una formula es completa si consiste del conjunto completo de implicantes primos y tendrá un número exponencial de cláusulas
- Implicante primo es un implicante que no puede ser cubierto por un implicante mas general (con menos literales)

Backtracking search

- Una búsqueda en retroceso para SAT, es un proceso de búsqueda que atraviesa el espacio de 2^n asignaciones binarias posibles a las variables problema.
- Una variable cuyo valor ya ha sido determinado se dice asignada.
- Si no se ha determinado la variable entonces esta sin asignar y puede tomar valores de $x \equiv \{0,1\}$

- Asignación de verdad – Es el conjunto de variables asignadas con su correspondiente valor binario.

$$A = \{x_1 = 1,\ \ x_3 = 0,\ \ x_4 = 1\}$$

- Se dice que *A* esta completa si contiene a todas las variables de la fórmula. De otra forma se dice que es parcial.
- Al evaluar *A* en la fórmula se tienen tres salidas posibles: 0, 1, X

- Una asignación divide las cláusulas en:
  - Satisfechas
  - Insatisfechas
  - Sin resolver

- Las literales que no se han asignado se les llama literales libres.
- Si el número de literales libres en una cláusula es 1 se le llama cláusula unidad.

¿Cómo es el proceso de búsqueda?

- Se comienza por una asignación de verdad.
- Se organiza la búsqueda para una asignación de satisfacibilidad al mantener un árbol de decisión.
- Cada nodo del árbol contiene una asignación a una variable sin asignar y se le llama asignación por decisión.
- Se asocia un nivel de decisión a cada asignación por decisión para denotar su profundidad en el árbol.

Estructura del proceso de búsqueda

- El proceso itera de la siguiente forma:
- Se realiza una asignación por decisión sobre una variable no asignada.
- La búsqueda termina exitosamente si todas las cláusulas son satisfechas.
- Termina sin éxito si algunas cláusulas no son satisfechas y todas las posibles asignaciones han sido probadas.

- Se extiende la asignación actual siguiendo las condiciones lógicas.
- Estas nuevas asignaciones se llaman asignaciones de implicación o implicaciones.
- Si hay cláusulas que no se satisfacen, no fue una asignación de satisfacción.
- A esto se le llama conflicto y las asignaciones asociadas se les llama asignaciones de conflicto.
- Se deshace la asignación actual para intentar con una nueva asignación

- Aumentar implicados en una formula ayuda a incrementar el poder deductivo durante la búsqueda.
- El aprendizaje puede ser estático o dinámico
- GRASP utiliza el enfoque dinámico basado en el diagnostico de las causas de los conflictos.
- Introduce implicados adicionales solo cuando es requerido.

- BCP es el mecanismo básico para derivar implicaciones.
- Una asignación antecedente es el conjunto de variables que son responsables de implicar una asignación dada una cláusula.
- La secuencia de implicaciones generada por BCP es capturada por una gráfica de implicación dirigida que se define de la siguiente forma.

Gráfica de implicación

1. Cada vértice en la gráfica corresponde a una asignación de variable x=v(x).

2. Los predecesores del vértice x=v(x) en la gráfica son las asignaciones antecedentes A(x) que corresponden a la cláusula unitaria ω que lleva a la implicación de x. Las aristas dirigidas desde los vértices en A(x) al vértice x=v(x) son etiquetadas con ω. Los vértices que no tienen predecesores corresponden a las asignaciones por decisión.

3. Se agregan vértices especiales a la gráfica para indicar que hubo un conflicto.

Los predecesores de un vértice de conflicto κ corresponden a asignaciones variables que forzan a la cláusula ω a volverse insatisfacible y son vistas como la asignación antecedente A(κ). Todas las aristas dirigidas desde los vértices en A(κ) a son etiquetados con ω.

Los niveles de decisión son asignados de acuerdo a:

$$\partial(x)=\max\{\partial(y)|(y,v(y)) \text{ in } A(x)$$

Gráfica de implicación

Asignación actual:

$\{x_9=0@1,\ x_{10}=0@3,\ x_{11}=0@3,\ x_{12}=1@2,\ x_{13}=1@2\}$

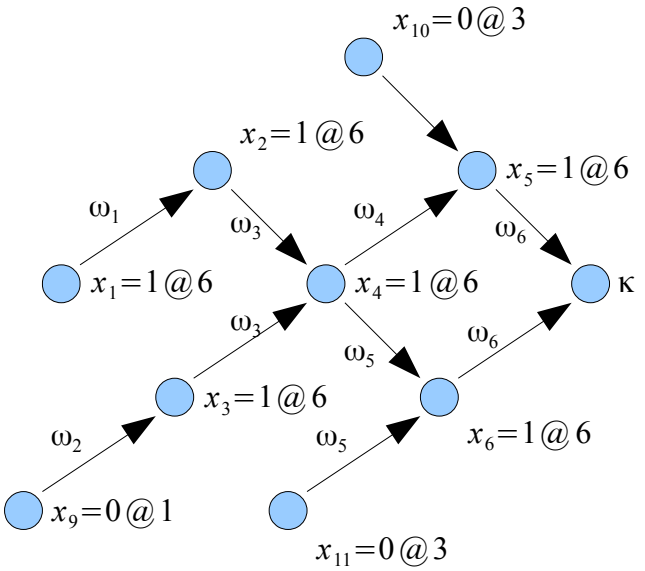
Asignación por Decisión:

$\{x_1=1@6\}$

Cláusulas:

$\omega_1=(\neg x_1+x_2)$
$\omega_2=(\neg x_1+x_3+x_9)$
$\omega_3=(\neg x_2+\neg x_3+x_4)$
$\omega_4=(\neg x_4+x_5+x_{10})$
$\omega_5=(\neg x_4+x_6+x_{11})$
$\omega_6=(\neg x_5+\neg x_6)$
$\omega_7=(x_1+x_7+\neg x_{12})$
$\omega_8=(x_1+x_8)$
$\omega_9=(\neg x_7+\neg x_8+\neg x_{13})$

Gráfica de implicación:



Estructura General de GRASP

```
GRASP()  {
    return  (Search (d, &ß ) != SUCCESS) ?
            FAILURE : SUCCESS;
}
Search (d, & ß) {
    if (Decide (d) == SUCCESS)
       return SUCCESS;
    while (TRUE) {
       if  (Deduce (d) != CONFLICT) {
           if (Search (d + 1,  ) == SUCCESS)
               return SUCCESS;
           else if ( ß != d) {
               Erase(); return CONFLICT;
           }
       }
       if (Diagnose (d, ß ) == CONFLICT) {
           Ease(); return CONFLICT;
       }
       Erase();
    }
}
```

```
Diagnose (d, &ß) {
      ωc(κ) = Conflict_Induced_Clause();
      Update_Clause_Database (ωc(κ));
      ß = Compute_Max_Level();
      if (ß!= d) {
          add new conflict vertex κ  to I;
          record    A(κ)  ;
          return CONFLICT;
       }
       return SUCCESS;
}
```

- La búsqueda recursiva consiste de cuatro operaciones
  - 1. Decide(). Escoge una decisión por asignación en cada uno de las etapas del proceso de búsqueda.
  - Por ejemplo: En cada nodo del árbol de decisión evalúa el número de cláusulas que se satisfacen directamente en cada asignación para cada variable. Escoge la variable y la asignación que satisface directamente al mayor número de cláusulas
  - 2. Deduce(). Implementa BCP y mantiene la gráfica de implicación resultante.
  - 3. Diagnose(). Identifica las causas de los conflictos y puede aumentar la base de cláusulas con implicados adicionales.
  - 4. Erase(), la cual borra las asignaciones en el nivel de decisión actual.

- A Decide(), Deduce() y Diagnose() se les conoce como los motores de decisión, deducción y diagnóstico.
- Distintas implementación de estos motores llevan a diferentes algoritmos SAT

- En GRASP se da una cláusula inicial, una asignación inicial en el nivel de decisión 0.
- La asignación inicial puede ser vacía.
- La asignación inicial es una restricción que hace se limite la búsqueda.
- Las base de cláusulas y la asignación inicial se van modificando conforme proceda la búsqueda.

Análisis de conflictos

- Se determina las variables que causan el conflicto, analizando las aristas que convergen en el vértice κ
- La conjunción de estas es un implicante que es condición suficiente para que aparezca el conflicto.
- Negando este implicante se puede añadir a la base original.
- Da las bases para implementar el backtracking no cronológico

- El diagnostico de conflictos produce:
  - Implicados que no existían en la base original (equivalencias basadas en conflictos)
  - Indicación de que el conflicto
    - Fue por una asignación por decisión reciente (se intenta con la asignación opuesta)
    - Fue por una asignación por decisión anterior (se hace un regreso al nivel de decisión donde se realizó la decisión)
- Esto es un backtracking no cronológico y se llama backtracking dirigido por conflicto.

Gráfica de implicación

Asignación actual:

$\{x_9=0@1,\ x_{10}=0@3,\ x_{11}=0@3,\ x_{12}=1@2,\ x_{13}=1@2\}$

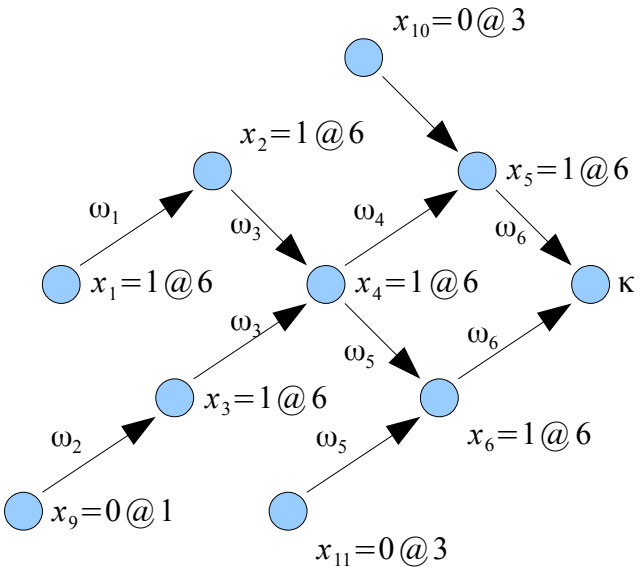
Asignación por Decisión:

$\{x_1=1@6\}$

Cláusulas:

$\omega_1=(\neg x_1+x_2)$
$\omega_2=(\neg x_1+x_3+x_9)$
$\omega_3=(\neg x_2+\neg x_3+x_4)$
$\omega_4=(\neg x_4+x_5+x_{10})$
$\omega_5=(\neg x_4+x_6+x_{11})$
$\omega_6=(\neg x_5+\neg x_6)$
$\omega_7=(x_1+x_7+\neg x_{12})$
$\omega_8=(x_1+x_8)$
$\omega_9=(\neg x_7+\neg x_8+\neg x_{13})$

Gráfica de implicación:



$A_C(\kappa)=\{x_1=1,\ x_9=0,\ x_{10}=0,\ x_{11}=0\}$
$\omega_C(\kappa)=(\neg x_1+x_9+x_{10}+x_{11})$

- $\omega_C(\kappa)$ es la cláusula inducida por el conflicto
- $A_C(\kappa)$ es la asignación conflictiva en el vértice de κ
- Para facilitar el computo de $A_C(\kappa)$ se separa de la siguiente forma:

$$\Lambda(x) = \{(y,v(y)) \in A(x) | \delta(y) < \delta(x)\}$$

$$\Sigma(x) = \{(y,v(y)) \in A(x) | \delta(y) = \delta(x)\}$$

- De forma recursiva de define como:

$$A_C(x) = \begin{cases} (x, v(x)) & \text{if } A(x) = \emptyset \\ \Lambda(x) \cup \left[\bigcup_{(y, v(y)) \in \Sigma(x)} A_C(y) \right] & \text{otherwise} \end{cases}$$

- La cláusula inducida por el conflicto

$$\omega_C(\kappa) = \sum_{(x, v(x)) \in A_C(\kappa)} x^{(v(x))}$$

Motor estándar de diagnóstico de conflicto

- Una cláusula inducida por conflicto habilita la derivación de mas implicaciones y ayuda a podar la búsqueda
- Se puede asegurar la variable de decisión y determina el nivel de retroceso para la búsqueda
- No es requisito que se agregue a la base
- Si se agrega se asegura que no se genere una asigna que cause el mismo conflicto.

Afirmaciones por fallas

- Se dan cuando en la cláusula inducida por conflicto se encuentra la variable de decisión actual.
- Al borrar la secuencia de implicación en el nivel actual de decisión hace que se convierta en una cláusula unitaria.
- Esto hace que la variable de decisión tome el valor opuesto.

Regreso dirigido por conflicto

- Si todas las literales en la cláusula inducida por el conflicto corresponden a variables asignadas en niveles de decisión mas bajos que el actual, entonces el proceso de búsqueda necesita regresar.
- Esto solo sucede si un conflicto ya ha sido diagnosticado previamente.
- El regreso se hace al nivel más alto donde se fueron realizadas las asignaciones conflictivas.

$$ß=\max\{\partial(x)|(x,v(x)) \text{ in } A_C(\kappa')\}$$

- Si ß=*d* – 1 ; *d* = nivel de decisión actual

  Se realiza un backtrack cronológico.

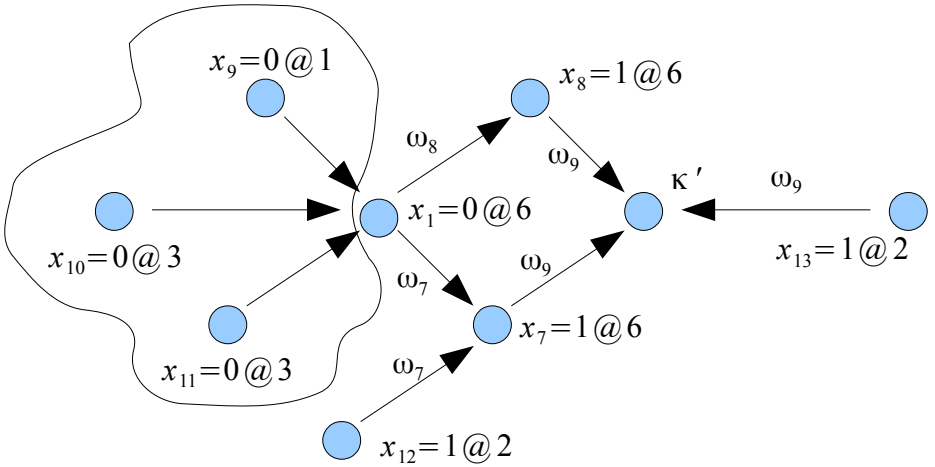
  La búsqueda se regresa al nivel inmediatamente anterior

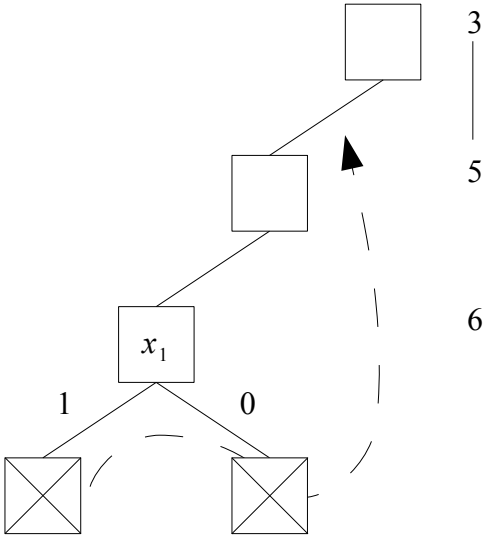
- Si ß < *d* -1

  El backtrack es no cronológico.

  La búsqueda puede regresar saltando varios niveles de decisión

Backtracking no cronológico





Desventajas

- GRASP tiene dos desventajas visibles
- El análisis de conflictos introduce un overhead que puede desembocar en tiempos altos de ejecución
- El tamaño de la base de cláusulas crece cada vez que se realiza un backtrack

En resumen ...

- Determinar las causas de los conflictos y lleva un registro de ellas.
- Realiza un backtrack no cronológico a niveles anteriores del árbol de búsqueda.
- Reduce el espacio de búsqueda al podar grandes porciones del árbol de decisión.
- Se anticipa a la ocurrencia de conflictos similares que se puedan presentar en búsquedas posteriores.
- Lleva un registro simple de las cadenas de causalidad que conducen a los conflictos, ayuda en la identificación de asignaciones necesarias para encontrar una solución.