.UBAfiuba
FACULTAD DE INGENIERÍA

Branch & Bound: Introducción

Teoría de Algoritmos I (75.29 / 95.06)

Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Branch & Bound (Ramificación y poda)

- **Variante de Backtracking para problemas de optimización**
  - Se busca responder como resultado la maximización o minimización de cierto valor.
- **Utiliza un árbol de espacio de estados.**
  - Recorre todo el espacio de estados del problema (en el peor de los casos)
  - Es considerado un algoritmo de búsqueda exhaustiva.
- **Utiliza la propiedad de corte para la poda de los estados en el árbol**
- **Agrega una función costo**
  - que permite un criterio de poda adicional
  - Que determina el orden de inspección de los estados
- **Permite el recorrido del árbol según diferentes métodos**

Problemas de optimización y soluciones óptimas

En el árbol de estados pueden existir varios estados respuesta

- Estos estados cumplen las restricciones implicitas y explicitas del problema.
- Cada uno de ellos tendrá una valor característico (de acuerdo al problema).
- Corresponde a soluciones factibles

Una solución factible es una solución óptima

- Si Maximiza/minimiza (según lo buscado) el valor característico.

Al recorrer el árbol de estados

- Guardamos la mejor solución factible encontrada hasta el momento
- Al finalizar la exploración tendremos la solución optima (si existe)

Poda del árbol

Para expresar una posible solución

utilizaremos una tupla de como mucho t≤n elementos (x1,x2, …xt-1, xt)

Un estado del problema

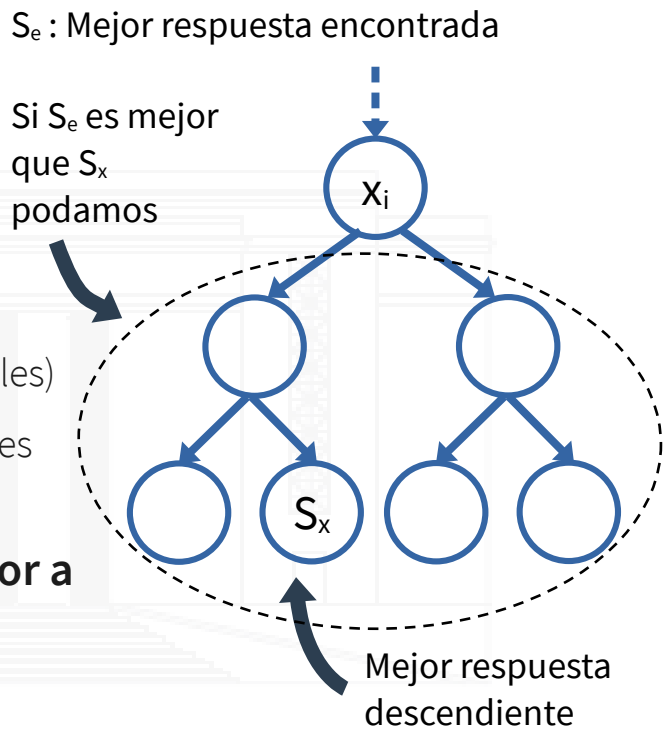
Contendrá una tupla parcial (o total) de la tupla solución

Corresponderá a un prefijo de un conjunto de estados respuesta (soluciones factibles)

Existirá un estado respuesta dentro de ese conjunto que sera el mayor (o menor si es un problema de minimzación) entre ellos

Si el "mejor" estado respuesta descendiente del estado actual es peor a la mejor solución previamente encontrada

Podemos evitar inspeccionar esas ramas del árbol



Función costo

Llamaremos función de costo

A la función que, dado un estado del problema determina el posible valor de un posible estado respuesta descendiente del mismo.

La función costo

Realiza una estimación el mejor valor descendiente posible (un valor aproximado)

Puede sobre estimar este valor, pero no debe infravalorarlo.

En caso de hacerlo se podría podar ramas incorrectamente y perder el resultado óptimo.

Podemos comparar dos estados del problema según sus funciones costo

Aquella con mejor valor es mejor candidato a tener entre sus descendientes la solución óptima al problema

Recorrido del árbol

En backtracking

- Utilizabamos Depth-First Searh para recorre el árbol
- No había un criterio establecido para determinar que rama profundizar del árbol

En Branch & Bound

- Se pueden aplicar diferentes estrategias de recorridos del árbol
- Éstas aprovechan la función costo
- Se privilegia la exploración de ramas con mejor valor en la función costo

Recorrido del árbol: Depth-first branch-and-bound

Partimos de un nodo

expandimos todos los nodos descendientes.

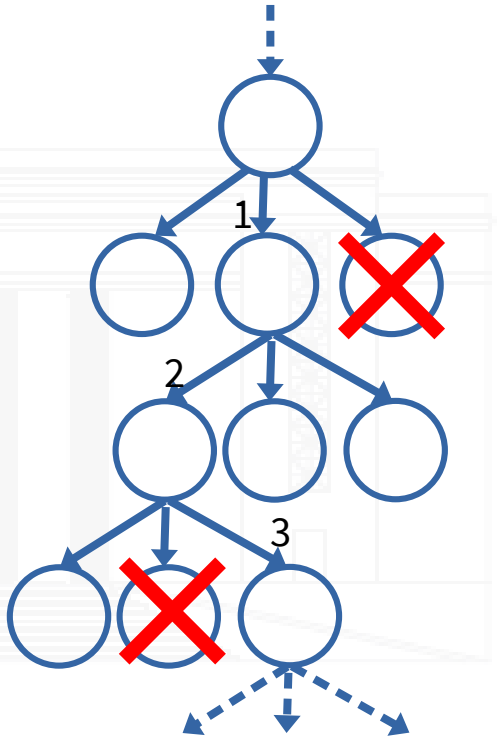
Se evalúan, podando utilizando la función límite

Entre los restantes se realiza la exploración seleccionando el más promisorio aún no explorado teniendo en cuenta la función costo

Al no quedar nodos por explorar se regresa al nodo padre.

El proceso finaliza

al no quedar nodos por explorar



Depth-first branch-and-bound – Pseudocódigo

```
Sea estadoInicial la raiz del arbol de estados
BranchAndBound(estadoInicial)

BranchAndBound (estadoActual):
    Sea descendientes los nodos descendientes de estadoActual
    Por cada posible estadoDescendiente de estadoActual
        Si estadoDescendiente supera la propiedad de corte
            Calcular fc funcion costo de estadoDescendiente
            Agregar estadoDescendiente a descendientes con fc

    Mientras existan estados descendientes no explorados
        Sea estadoProximo el estado en descendientes aún no analizado de mayor fc
        Si el fc de estadoProximo es mayor a la mejor solución obtenida
            Si estadoProximo es un estado respuesta y es superior a la mejor
                mejor = estadoProximo

            Backtrack(estadoProximo)
```

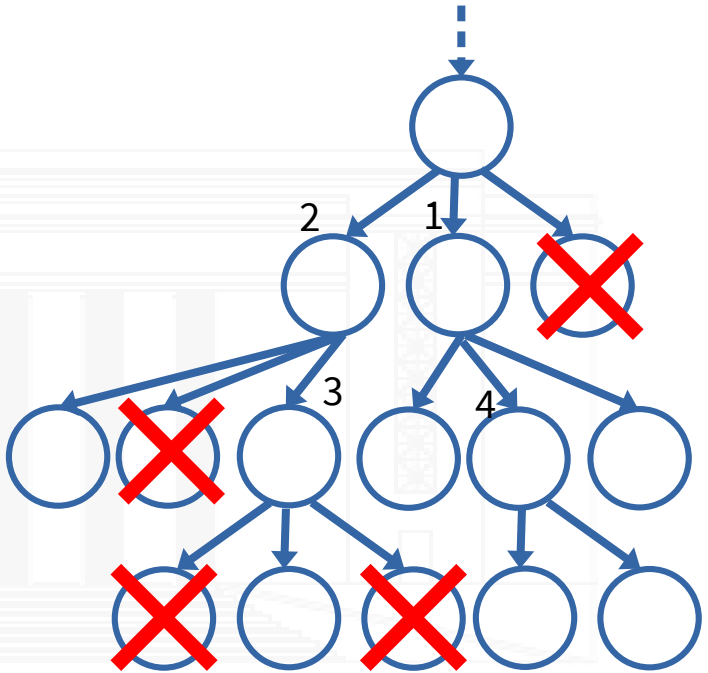
Recorrido del árbol – Best-first search

Se inicia con una cola de prioridad utilizando la funcion costo de forma descendente.

- En la cola de prioridad se incluye inicialmente la raiz del árbol

Se toma el nodo en la cola de mayor valor de costo que supere a la mejor solución encontrada

- Determinamos si corresponde a un estado respuesta y supera el mejor encontrado actual
- Expandimos todos los nodos descendientes.
- Podamos aquellos que no superan la función límite
- Calculamos su función costo y aquellos que superan al mejor encontrado actual lo insertamos en la cola



Best-first search – Pseudocódigo

```
Sea estadosDisponibles los estados en el árbol expandidos por analizar

Sea estadoInicial la raiz del arbol de estados
Calcular fc funcion costo de estadoInicial
Agregar estadoInicial con fc a estadosDisponibles

Mientras queden estados estos en estadosDisponibles
    Sea estadoActual el estado de mayor fc en estadosDisponibles

    Si estadoActual supera la propiedad de corte
        Si fc de estadoActual es mayor a la mejor solución obtenida
            Si estadoActual es un estado respuesta
                mejor = estadoProximo

            Por cada posible estadoSucesor de estadoActual
                Calcular fc funcion costo de estadoInicial
                Agregar estadoSucesor con fc a estadosDisponibles
```