Ramificación y poda (Branch&Bound)

Ramificación y poda (Branch&Bound)
Objetivos

- Comprender la finalidad de branch&bound.
- Comprender el concepto de branch&bound.
- Comprender el esquema de branch&bound.

Ramificación y poda (Branch&Bound)
Comprender la finalidad de branch&bound

Branch&Bound es una técnica aplicada a problemas que cumplan estas condiciones:

- Que la solución se pueda expresar mediante una tupla (x1,x2,x3,...,xn).
- Que el problema se formule o reformule como la búsqueda de aquella tupla que maximiza un determinado criterio.

Branch&Bound realiza una búsqueda sistemática de la solución óptima la problema, sea cual sea el problema.

Ramificación y poda (Branch&Bound)
Comprender el concepto de branch&bound

Para ello, branch&bound aborda el problema de una forma parecida a backtracking:

- En cada momento, selecciona el mejor nodo disponible.
- Expande este nodo generando todas las soluciones que parten de él y los añade a los nodos restantes.

Luego se mejora el tiempo de exploración aplicando **podas**:

- Si solución encontrada no es factible.
- Si sabemos que la solución encontrada no deja alcanzar la óptima.
- Si localizamos ciclos (no está acotado el tamaño de la tupla).

Ramificación y poda (Branch&Bound)
Comprender el concepto de branch&bound

Se definen las restricciones implícitas como las restricciones que se nos dan en el enunciado del problema y que debemos comprobar para saber si una solución es factible.

Se definen las restricciones explícitas como aquellas que están limitadas por el funcionamiento del algoritmo (por ejemplo un bucle de 0 hasta n no va a valer más allá de n) y que no debemos comprobar para saber si una solución es factible.

Ramificación y poda (Branch&Bound)
Comprender el esquema de branch&bound

```
TDA solucion { coste, tupla[], pos }   //DEFINIR operator<

solucion branchandbound(solucion a) {
  priority_queue<solucion> candidatos;
  solucion actual,nuevo;
  candidatos.push(a);
  solucion mejor = a;
  while (!candidatos.empty()) {
    actual = candidatos.top();
    candidatos.pop();
    int nextpos = actual.getPosicionSiguiente();
    actual.nextpos++;
    for (i = 0;i < x;i++) { //Valores posibles.
      nuevo = actual;
      nuevo.Poner(nextpos,i);
      if (nuevo.calcularCoste() < mejor.calcularCoste())
        mejor = nuevo;
      candidatos.push(nuevo);
    }
  }
  return mejor;
}
```

Ramificación y poda (Branch&Bound)
¿Qué hemos aprendido?

- **Finalidad de branch&bound.**
- **Concepto de branch&bound.**
- **Esquema de branch&bound.**

Ramificación y poda (Branch&Bound)
Ejemplos

Ejemplo: Viajante de comercio

Objetivo: Escoger un ciclo que recorra una sola vez cada ciudad y cuya sumatoria del coste sea la menor posible.