HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

APPLIED ALGORITHMS

APPLIED ALGORITHMS

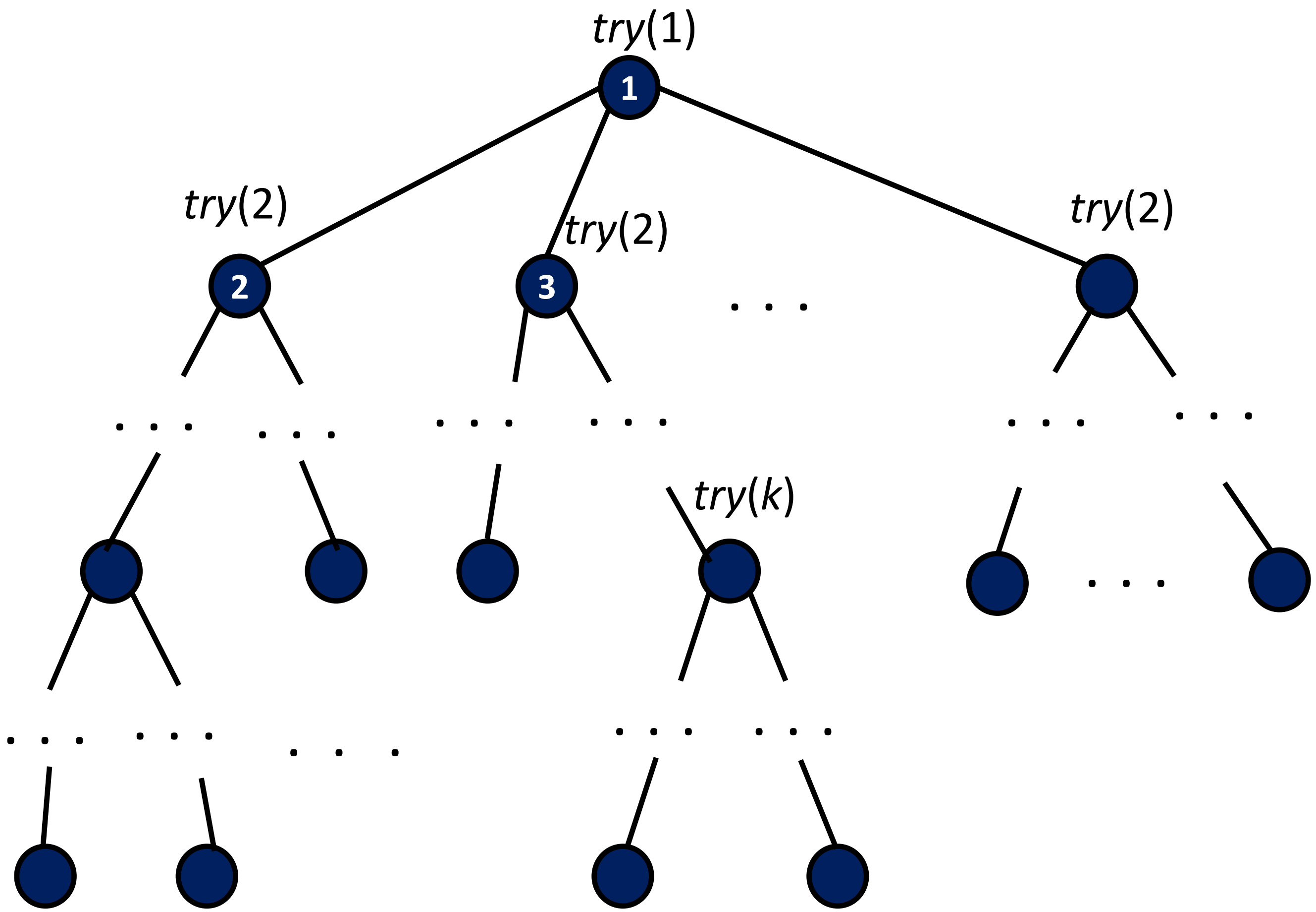
BACKTRACKING, BRANCH AND BOUND

CONTENTS

- General diagram of backtracking, branch and bound
- The problem of bus routes picking up and dropping off passengers
- Delivery truck route problem
- 2D material cutting problem

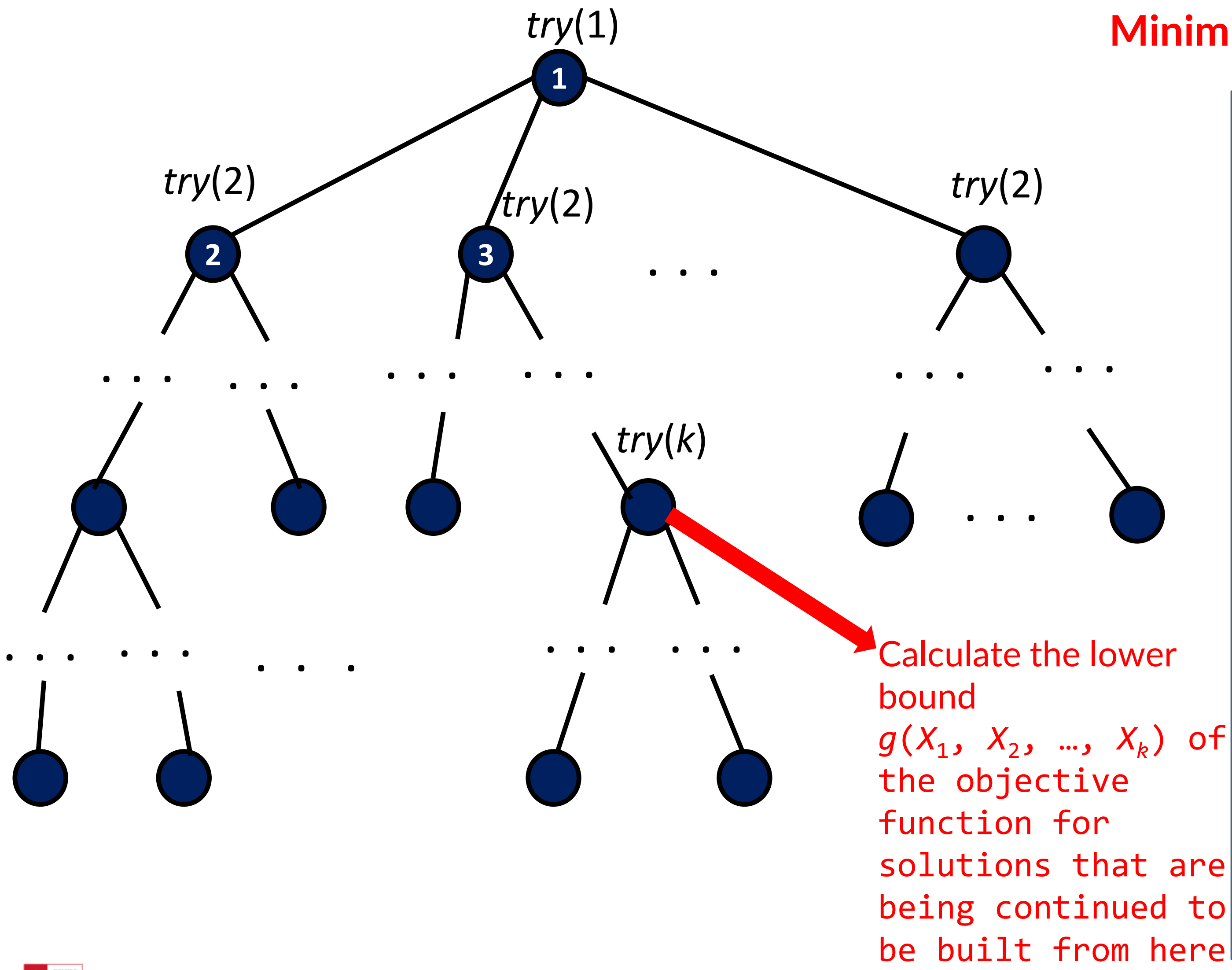
GENERAL DIAGRAM OF BACKTRACKING, BRANCHING AND BOUND

- The backtracking algorithm allows us to solve combinatorial enumeration problems and combinatorial optimization problems
- The alternative is modeled by a sequence of decision variables $X_1, X_2, \ldots, X_n$
- Need to find for each variable X_i a value selected from a given discrete set A_i such that
  - The constraints of the problem are satisfied
  - Optimize a given objective function
- Backtracking algorithm
  - Traverse through all variables (e.g. order from $X_1, X_2, \ldots, X_n$), for each variable X_k :
    - Traverse through all possible values that could be assigned to X_k, for each value v:
      - Check constraints
      - Assign $X_k = v$
      - If $k = n$ then record a solution to the problem
      - Otherwise, consider the variable X_{k+1}



Enumeration problem

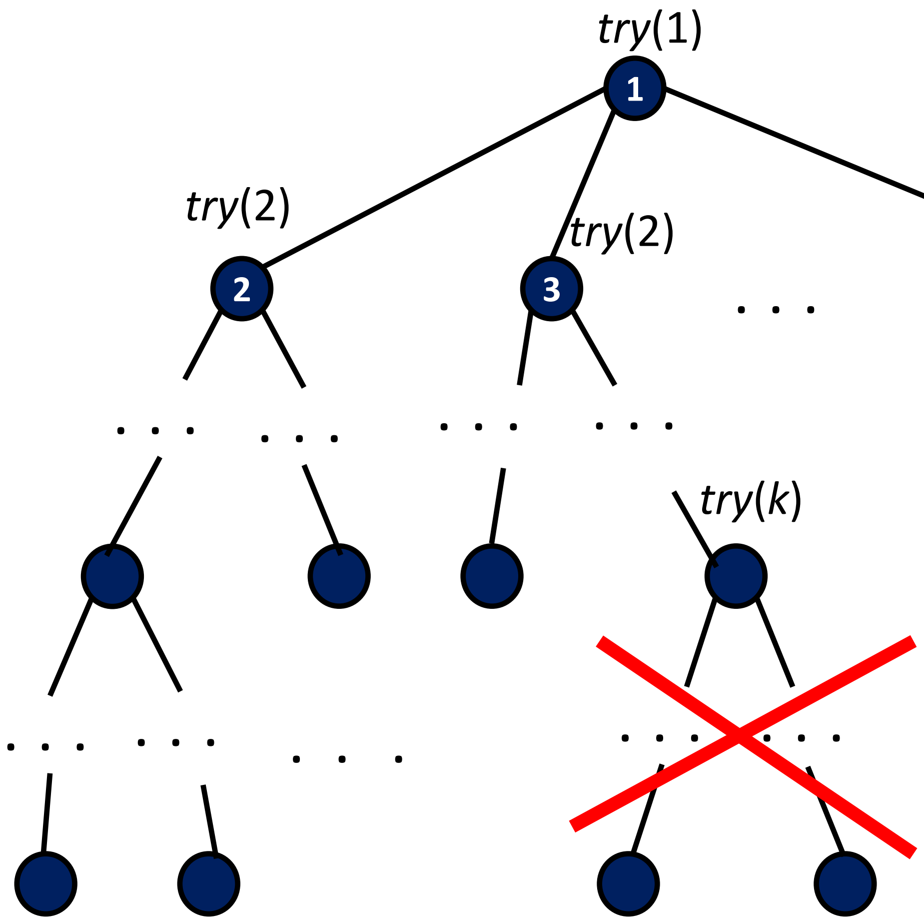
```
try(k){ //Try out the possible values assigned to X_k
  for v in A_k do {
     if check(v,k){
        X_k = v;
        [Update a data structure D]
        if k = n then solution();
        else {
            try(k+1);
        }
        [Recover the data structure D]
     }
  }
}
```



Minimize optimization problem (Denote f^* : optimal value)

```
try(k){//Try out the possible values assigned to X_k
  for v in A_k do {
     if check(v,k){
        X_k = v;
        [Update a data structure D]
        if k = n then updateBest();
        else {
            if g(X_1, X_2, …, X_k) < f* then
                try(k+1);
        }
        [Recover the data structure D]
     }
  }
}
```

GENERAL DIAGRAM OF BACKTRACKING, BRANCHING AND BOUND



Do not further build the solution if $g(X_1, X_2, \ldots, X_k) \geq f^*$

Minimize optimization problem (Denote f^* : optimal value)

```
try(k){//Try out the possible values assigned to X_k
  for v in A_k do {
     if check(v,k){
        X_k = v;
        [Update a data structure D]
        if k = n then updateBest();
        else {
            if g(X_1, X_2, …, X_k) < f* then
                try(k+1);
        }
        [Recover the data structure D]
     }
  }
}
```

The problem of bus routes picking up and dropping off passengers

- A bus departing from point 0 needs to build a route that could serve n passengers and return to point 0. Passenger i has: the pick-up point is i and the drop-off point is $i + n$ ($i = 1, 2, \ldots, n$). The bus has K seats to serve passengers. The travel distance from point i to point j is d(i, j), with $i, j = 0, 1, 2, \ldots, 2n$. Calculate the route for the bus so that the total distance traveled is minimal, and the number of passengers on the bus never exceeds K.

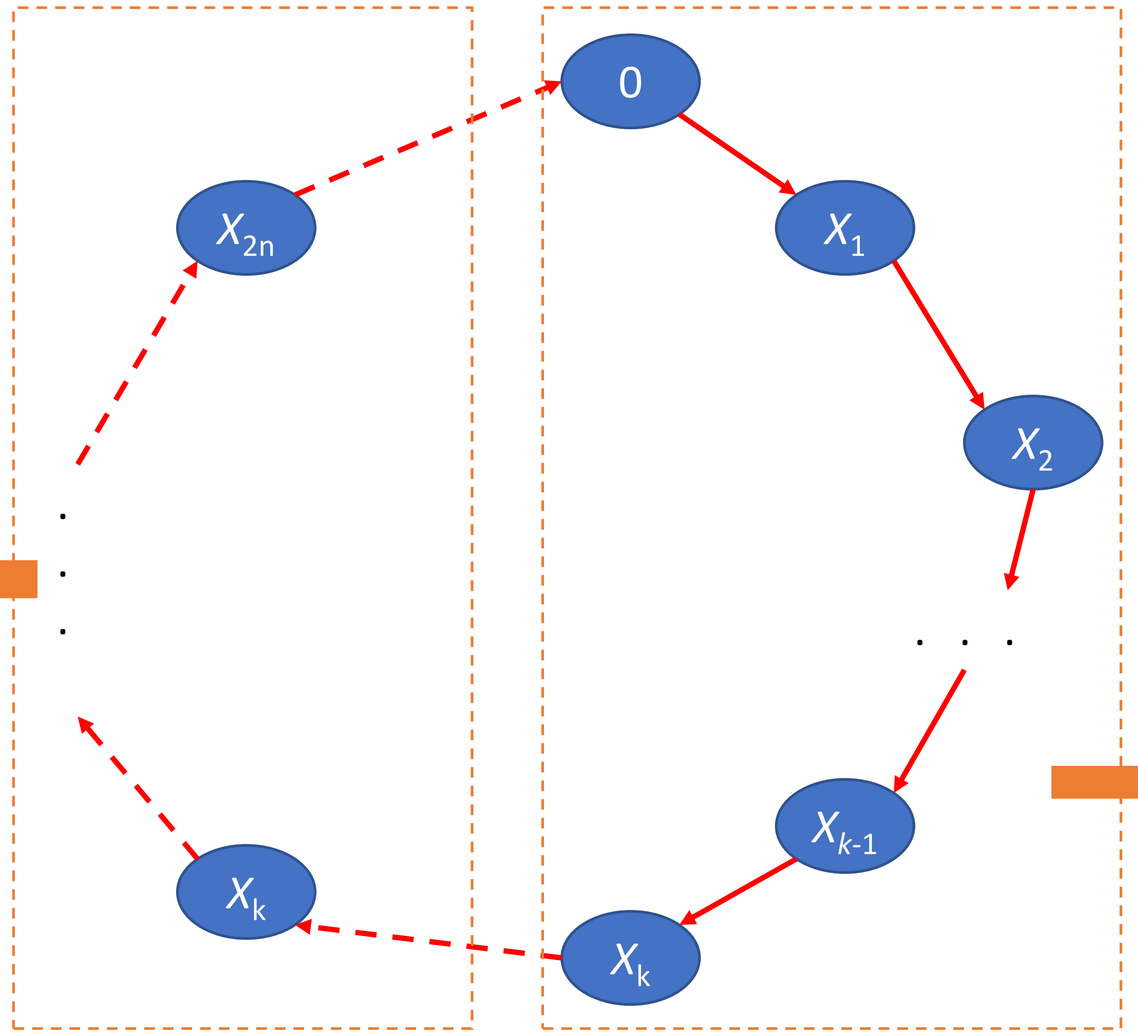
The problem of bus routes picking up and dropping off passengers

- A bus departing from point 0 needs to build a route that could serve n passengers and return to point 0. Passenger i has: the pick-up point is i and the drop-off point is $i + n$ ($i = 1, 2, \ldots, n$). The bus has K seats to serve passengers. The travel distance from point i to point j is $d(i, j)$, with $i, j = 0, 1, 2, \ldots, 2n$. Calculate the route for the bus so that the total distance traveled is minimal, and the number of passengers on the bus never exceeds K.
- Branch and bound algorithm
  - Modelling problem: $X_1, X_2, \ldots, X_{2n}$ is the sequence of pick-up and drop-off points on the bus route (a permutation of 1, 2, …, 2n).
  - *Cmin*: the smallest distance among the distances between 2 points
  - Marker array: visited[v] = true means point *v* has appeared on the route and visited[v] = false, otherwise
  - load: number of passengers present in the vehicle
    - When the route reaches the pick-up point, the load increases by 1, and when it reaches the drop-off point, the load decreases by 1
  - *f*: length of the partial route
  - *f**: shortest route length that has been found

The problem of bus routes picking up and dropping off passengers

- **Analyze the lower bound**

- Untraveled route, including $2n+1-k$ segments, each segment has length $\geq Cmin$
- The length of the complete route developing further from X_k will be $\geq f + Cmin*(2n+1-k)$



The partial route that has gone through:

- k segments
- Length f

The problem of bus routes picking up and dropping off passengers

```
try(k){
  for v = 1 to 2n do {
     if check(v,k){
        Xk = v;
        f = f + d(Xk-1,Xk); visited[v] = true;
        if v ≤ n then load += 1; else load -= 1;
        if k = 2n then updateBest();
        else {
            if f + Cmin*(2n+1-k) < f* then
                try(k+1);
        }
        if v ≤ n then load -= 1; else load += 1;
        f = f - d(Xk-1,Xk); visited[v] = false;
     }
  }
}
```

```
check(v,k){
  if visited[v] = true then return false;
  if v > n then {
    if visited[v-n] = false then return false;
  }else{
    if load + 1 > K then return false;
  }
  return true;
}
```

```
updateBest(){
  if f + d(X2n,0) < f* then {
    f* = f + d(X2n,0);
  }
}
```

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !