

NỘI DUNG

- Sơ đồ chung nhánh và cận
- Bài toán lộ trình xe buýt đón trả khách
- Bài toán lộ trình xe tải giao hàng

ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

3

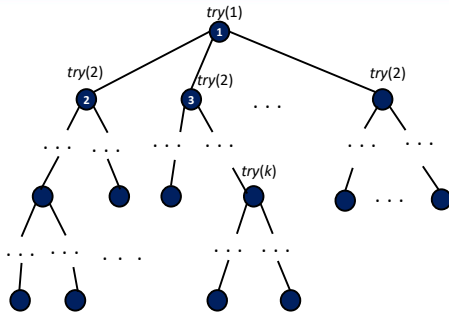
SƠ ĐỒ CHUNG QUAY LUI, NHÁNH VÀ CẬN

- Thuật toán quay lui cho phép ta giải các bài toán liệt kê tổ hợp và bài toán tối ưu tổ hợp
- Phương án được mô hình hóa bằng một dãy các biến quyết định X_1, X_2, \dots, X_n
- Cần tìm cho mỗi biến X_i một giá trị từ 1 tập rời rạc A_i cho trước sao cho
 - Các ràng buộc của bài toán được thỏa mãn
 - Tối ưu một hàm mục tiêu cho trước
- Tìm kiếm quay lui
 - Duyệt qua tất cả các biến (ví dụ thứ tự từ X_1, X_2, \dots, X_n), với mỗi biến X_k
 - Duyệt lần lượt qua tất cả các giá trị có thể gán cho X_k với mỗi giá trị v
 - Kiểm tra ràng buộc
 - Gán cho X_k
 - Nếu $k = n$ thì ghi nhận một phương án
 - Ngược lại, xét tiếp biến X_{k+1}

ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

4

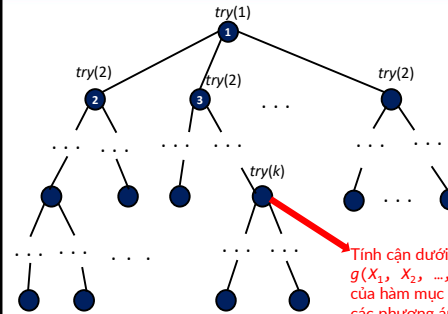
SƠ ĐỒ CHUNG QUAY LUI, NHÁNH VÀ CẬN



Bài toán liệt kê

```
try(k) { // thử các giá trị có thể gán cho  $X_k$ 
  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  $D$ ]
}
```

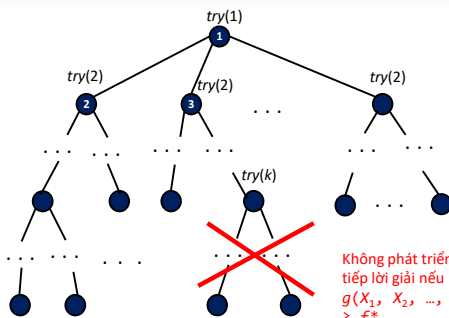
SƠ ĐỒ CHUNG QUAY LUI, NHÁNH VÀ CẬN



Bài toán tìm min (ký hiệu f^* : kỷ lục)

```
try(k) { // thử các giá trị có thể gán cho  $X_k$ 
  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  $g(X_1, X_2, \dots, X_k) < f^*$  then
        try( $k+1$ );
    }
  }
  [Recover the data structure  $D$ ]
}
```

SƠ ĐỒ CHUNG QUAY LUI, NHÁNH VÀ CẬN



Bài toán tìm min (ký hiệu f^* : kỷ lục)

```
try(k) { // thử các giá trị có thể gán cho  $X_k$ 
  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  $g(X_1, X_2, \dots, X_k) < f^*$  then
        try( $k+1$ );
    }
  }
  [Recover the data structure  $D$ ]
}
```

BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

- Một đội gồm K xe tải giống nhau cần được phân công để vận chuyển hàng hóa pepsi từ kho trung tâm (điểm 0) đến các điểm giao hàng 1,2,..., N và quay về kho. Khoảng cách di chuyển từ điểm i đến điểm j là $d(i,j)$
- Mỗi xe tải có tải trọng Q (mỗi chuyến chỉ vận chuyển tối đa Q thùng)
- Mỗi điểm giao hàng i có lượng hàng yêu cầu là $d[i]$ thùng, $i = 1, \dots, N$.
- Cần xây dựng phương án vận chuyển sao cho
 - Mỗi xe đều phải được phân công vận chuyển
 - Mỗi điểm giao chỉ được giao bởi đúng 1 xe
 - Tổng lượng hàng trên xe không vượt quá tải trọng của xe đó
 - Tổng độ dài lộ trình các xe là nhỏ nhất
- Lưu ý: có thể không dùng hết cả K xe.

BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

- Ví dụ $N = 3, K = 2, Q = 10, d[1] = 3, d[2] = 2, d[3] = 1$
→ có 6 phương án vận chuyển sau

Route[1] = 0 - 1 - 0	Route[1] = 0 - 1 - 2 - 0
Route[2] = 0 - 2 - 3 - 0	Route[2] = 0 - 3 - 0
Route[1] = 0 - 1 - 3 - 0	Route[1] = 0 - 2 - 0
Route[2] = 0 - 2 - 0	Route[2] = 0 - 3 - 1 - 0
Route[1] = 0 - 1 - 0	Route[1] = 0 - 2 - 1 - 0
Route[2] = 0 - 3 - 2 - 0	Route[2] = 0 - 3 - 0

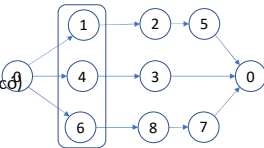
BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

- Ví dụ $N = 3, K = 2, Q = 4, d[1] = 3, d[2] = 2, d[3] = 1$
→ có 4 phương án vận chuyển sau

Route[1] = 0 - 1 - 0	Route[1] = 0 - 1 - 2 - 0
Route[2] = 0 - 2 - 3 - 0	Route[2] = 0 - 3 - 0
Route[1] = 0 - 1 - 3 - 0	Route[1] = 0 - 2 - 0
Route[2] = 0 - 2 - 0	Route[2] = 0 - 3 - 1 - 0
Route[1] = 0 - 1 - 0	Route[1] = 0 - 2 - 1 - 0
Route[2] = 0 - 3 - 2 - 0	Route[2] = 0 - 3 - 0

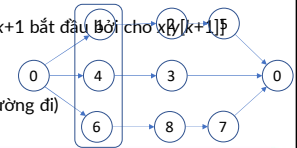
BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

- Thiết kế dữ liệu
 - $y[k]$ điểm giao đầu tiên của xe thứ k ($y[k] \in \{0, 1, 2, \dots, N\}$, với $k = 1, 2, \dots, K$)
 - $y[k] = 0$ nghĩa là xe k sẽ không được dùng để lập lộ trình
 - $x[i]$ là điểm tiếp theo của điểm giao i trên lộ trình ($x[i] \in \{0, 1, 2, \dots, N\}$, với $i = 1, 2, \dots, N$)
 - Do các xe giống hệt nhau nên giả định $y[k] \leq y[k+1]$, $k = 1, 2, \dots, K-1$
 - Nếu $y[k] > 0$ thì $y[k+1] > y[k]$
 - Các biến gắn với phương án bộ phận:
 - $visited[v] = \text{true}$ nếu v đã được thăm bởi 1 xe nào đó
 - $load[k]$: tổng lượng hàng trên xe k
 - f : tổng độ dài các quãng đường của lộ trình bộ phận hiện có
 - f^* : kỷ lục (độ dài quãng đường của phương án tốt nhất hiện có)



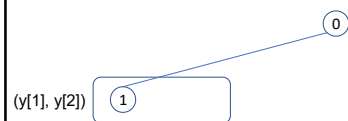
BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

- Chiến lược duyệt
 - Bắt đầu bằng việc duyệt bộ giá trị cho $(y[1], \dots, y[K])$
 - Với mỗi bộ giá trị đầy đủ của $(y[1], \dots, y[K])$, bắt đầu duyệt bộ giá trị cho $x[1], \dots, x[N]$ xuất phát từ $x[y[1]]$
 - Mỗi khi thử giá trị $x[v] = u$ cho xe thứ k thì
 - Nếu $u > 0$ (chưa phải điểm xuất phát) thử duyệt tiếp giá trị cho $x[u]$ vẫn trên chuyến xe thứ k
 - Nếu $u = 0$ (điểm xuất phát) thì
 - Nếu $k = K$ (đã đủ hết các chuyến cho K) xe và điểm giao nào cũng được thăm thì ghi nhận 1 phương án
 - Ngược lại, thử duyệt tiếp giá trị cho chuyến của xe $k+1$ bắt đầu bởi $x[y[k+1]]$
 - Biến nbR: ghi nhận số xe thực sự được lập lộ trình giao hàng
 - Biến segments
 - Ghi nhận số chặng (đoạn nối giữa 2 điểm liên tiếp trên đường đi)
 - Khi $segments = N + nbR$ thì thu được phương án đầy đủ



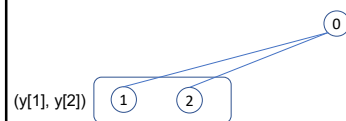
BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

- $K = 2, N = 6$



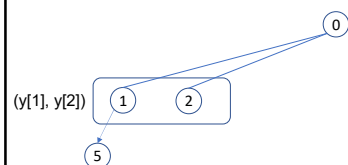
BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

- $K = 2, N = 6$



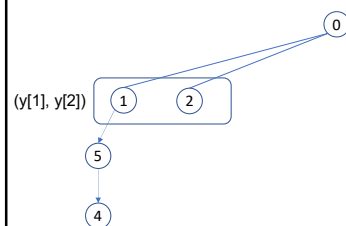
BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

- $K = 2, N = 6$



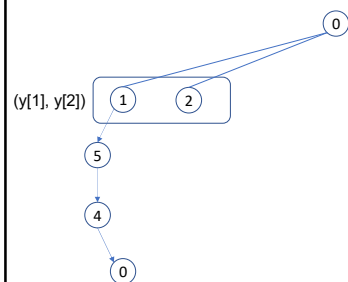
BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

- $K = 2, N = 6$



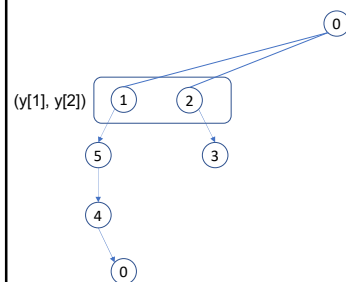
BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

- $K = 2, N = 6$



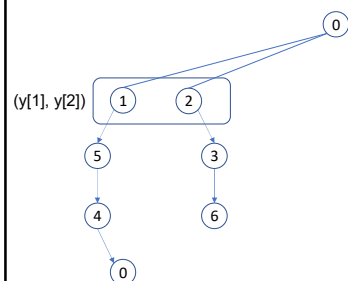
BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

- $K = 2, N = 6$



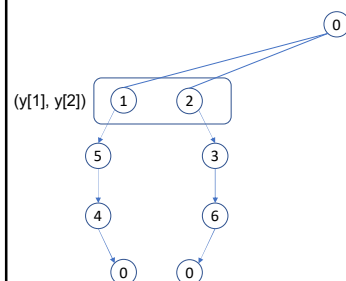
BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

- $K = 2, N = 6$



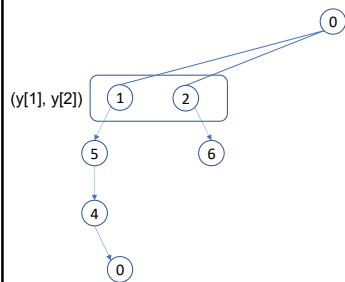
BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

- $K = 2, N = 6$



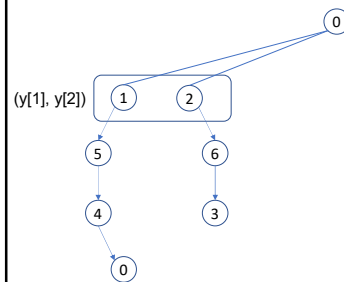
BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

- $K = 2, N = 6$



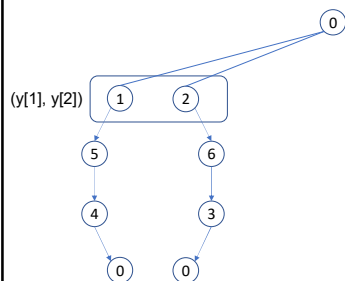
BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

- $K = 2, N = 6$



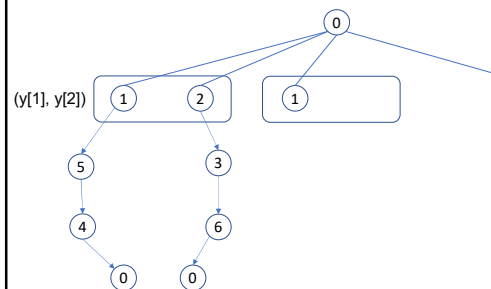
BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

- $K = 2, N = 6$



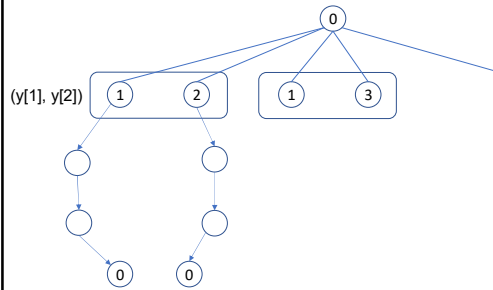
BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

- $K = 2, N = 6$



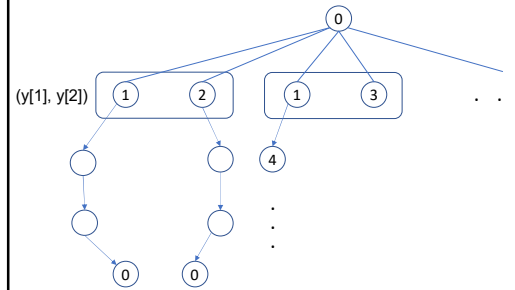
BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

- $K = 2, N = 6$



BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

- $K = 2, N = 6$



BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

```

TRY_X(s, k) { // thử giá trị cho x[s]
  if (s == 0) then {
    if k < K then
      TRY_X(y[k+1], k+1);
    return
  }
  [...]
}

checkX(v, k) {
  if v > 0 and visited[v]
    then return false;
  if load[k] + d[v] > Q
    then return false;
  return true;
}

for v = 0 to n do {
  if checkX(v, k) then {
    x[s] = v; visited[v] = true; f = f + c[s, v];
    load[k] = load[k] + d[v]; segments = segments + 1;
    if v > 0 then { if f + (n + nbr - segments) * Cmin < f* then TRY_X(v, k); }
  } else {
    if k = K then {
      if segments = n + nbr then updateBest();
    } else {
      if f + (n + nbr - segments) * Cmin < f* then TRY_X(y[k+1], k+1);
    }
  }
  visited[v] = false; f = f - c[s, v];
  load[k] = load[k] - d[v]; segments = segments - 1;
}

```

BÀI TOÁN LỘ TRÌNH XE TẢI GIAO HÀNG

```

checkY(v, k) {
  if v = 0 then return true;
  if load[k] + d[v] > Q then
    return false;
  if visited[v] = true then
    return false;
  return true;
}

solve() {
  f = 0; f* = +∞; y[0] = 0;
  for v = 1 to n do
    visited[v] = false;
  TRY_Y(1);
  output(f*);
}

TRY_Y(k) { // thử giá trị cho y[k]
  s = 0;
  if y[k-1] > 0 then s = y[k-1] + 1;
  for v = s to n do {
    if checkY(v, k) then {
      y[k] = v;
      if v > 0 then segments = segments + 1;
      visited[v] = true; f = f + c[0, v]; load[k] = load[k] + d[v];
      if k < K then TRY_Y(k+1);
    } else { nbr = segments; TRY_X(y[1], 1); }
    load[k] = load[k] - d[v]; visited[v] = false; f = f - c[0, v];
    if v > 0 then segments = segments - 1;
  }
}

```



HUST

 hust.edu.vn  fb.com/dhbkhn

THANK YOU !