



1


Thành viên



Võ Nhật Thanh
19522245



Trần Trung Tín
19522351



Lê Trần Trọng Khiêm
19521689

2

Mục tiêu của nhóm

Các bạn hiểu, cài đặt được, nhận diện bài toán và áp dụng

3

3

Nội dung thuyết trình

4

1

Đặt vấn đề

4

Ứng dụng

2

Định nghĩa branch and bound

5

Tổng kết

3

Ưu và nhược điểm

6

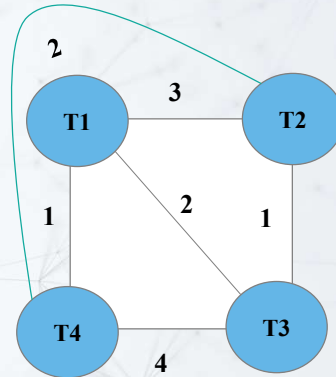
Bài tập về nhà

4

Đặt vấn đề

Bài toán người giao hàng

Tìm **chu trình ngắn nhất** đi qua mỗi thành phố đúng **một lần**



5

Các hướng tiếp cận

- ▶ Vết cặn
 - ▶ Vết cặn quay lui
 - ▶ Một số phương pháp khác
- Nhược điểm chung: tốn kém chi phí

Ý tưởng

Loại bỏ các bước đi không cần thiết

➡ **Branch and bound**



6

Định nghĩa branch and bound

7

- Phương pháp tối ưu bài toán tổ hợp và rời rạc
- Xây dựng cây để tìm phương án tối ưu (sử dụng cận để hạn chế nhánh)
- Là thuật toán quay lui
 - Tiếp tục đào sâu tạo ra cấu hình tốt hơn cấu hình đã lưu trữ
 - Nhờ B&B ta có thể quay sớm hơn backtracking cổ điển

7

Mô hình chung



Tối ưu $P = (X, f)$

P là vấn đề cần tối ưu

F là hàm tính cận

X là tập các giải pháp

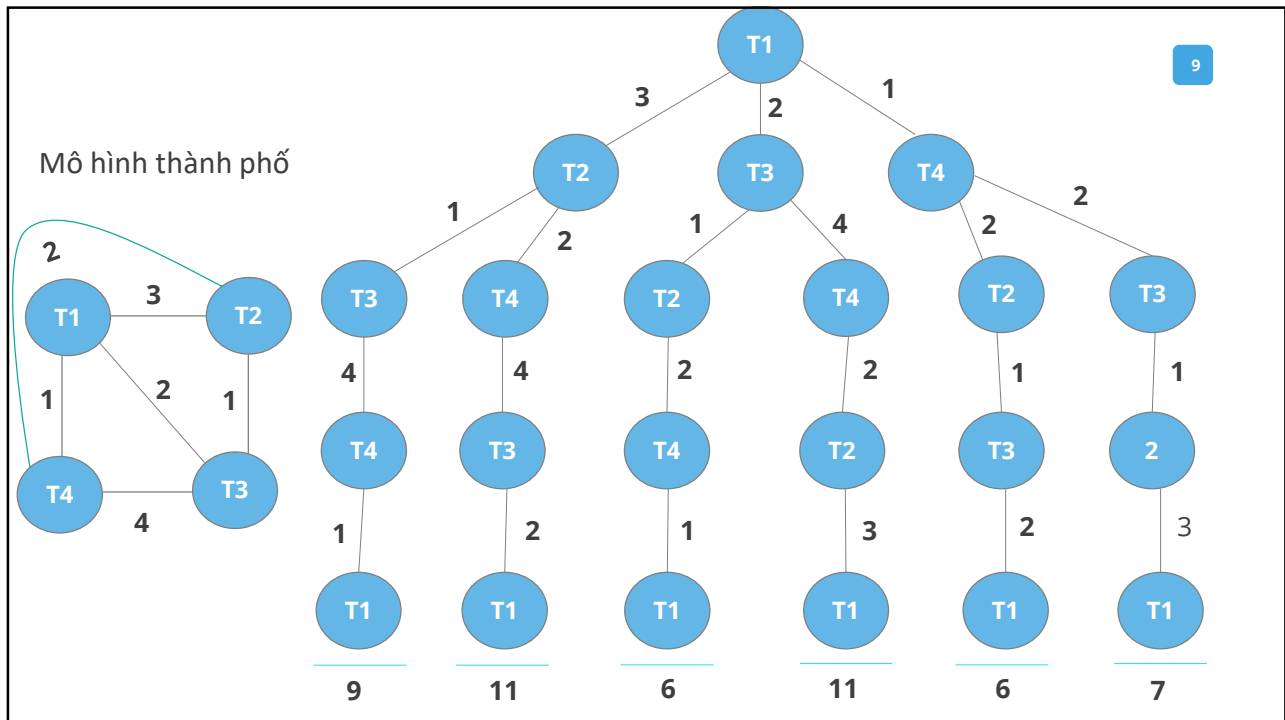


Tập giải pháp X

$X = \{ a = (a_1, a_2, \dots, a_{n-1}, a_n) \in \prod_{i=1}^n A_i \mid a \in P(x) \}$

A_i là trạng thái không gian tìm kiếm

8



9

	1	2	3	4
1	∞	3	2	1
2	3	∞	1	2
3	2	1	∞	4
4	1	2	4	∞

	1	2	3	4	
1	∞	2	1	0	1
2	2	∞	0	1	1
3	1	0	∞	3	1
4	0	1	3	∞	1
	0	0	0	0	4 + 0 = 4

→ Tổng chi phí giảm

$C(s) = C(R) + A[i][j] + r$

C(s) : Tổng chi phí
C(R): Chi phí nút trước
A[i][j]: Chi phí vị trí ban đầu tại A[i][j]
r : chi phí mất mát

10

11

Path 1-2

	1	2	3	4
1	∞	2	1	0
2	2	∞	0	1
3	1	0	∞	3
4	0	1	3	∞

	1	2	3	4	
1	∞	∞	∞	∞	0
2	∞	∞	0	0	0
3	0	∞	∞	1	1
4	0	∞	3	∞	0
	0	0	0	1	2
$C(2) = 4 + 2 + 2 = 8$					

Path 1-3

	1	2	3	4	
1	∞	∞	∞	∞	
2	2	∞	∞	0	
3	∞	0	∞	3	
4	0	1	∞	∞	
	0	0	0	0	
$C(3) = 4 + 1 + 1 = 6$					

Path 1-4

	1	2	3	4	
1	∞	∞	∞	∞	
2	2	∞	0	∞	
3	0	0	∞	∞	
4	∞	0	2	∞	
	1	0	0	0	
$C(3) = 4 + 0 + 2 = 6$					

Path 1-3

	1	2	3	4
1	∞	∞	∞	∞
2	1	∞	∞	0
3	∞	0	∞	3
4	0	1	∞	∞

Path 1-3-4

	1	2	3	4	
1	∞	∞	∞	∞	0
2	0	∞	∞	∞	1
3	∞	∞	∞	∞	0
4	∞	0	∞	∞	1
	0	0	0	0	2

$C(3-4) = 6 + 3 + 2 = 11$

Path 1-3-2

	1	2	3	4	
1	∞	∞	∞	∞	0
2	∞	∞	∞	0	0
3	∞	∞	∞	∞	0
4	0	∞	∞	∞	0
	0	0	0	0	0

$C(3-2) = 6 + 0 + 0 = 6$

13

13

Path 1-4

	1	2	3	4
1	∞	∞	∞	∞
2	1	∞	0	∞
3	0	0	∞	∞
4	∞	0	2	∞

Path 1-4-3

	1	2	3	4	
1	∞	∞	∞	∞	0
2	0	∞	∞	∞	1
3	0	0	∞	∞	0
4	∞	∞	∞	∞	0
	0	0	0	0	1

$C(4-3) = 6 + 2 + 1 = 9$

Path 1-4-2

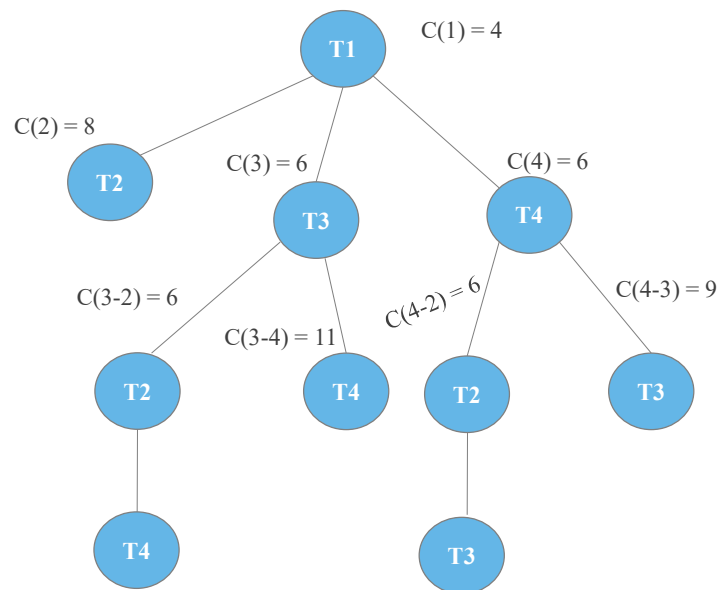
	1	2	3	4	
1	∞	∞	∞	∞	0
2	1	∞	0	∞	0
3	0	∞	∞	∞	0
4	∞	∞	∞	∞	0
	0	0	0	0	0

$C(4-2) = 6 + 0 + 0 = 6$

14

14

15



15

16

Path 1-3-2

	1	2	3	4
1	∞	∞	∞	∞
2	∞	∞	∞	0
3	∞	∞	∞	∞
4	0	∞	∞	∞

Path 1-4-2

	1	2	3	4
1	∞	∞	∞	∞
2	1	∞	0	∞
3	0	∞	∞	∞
4	∞	∞	∞	∞

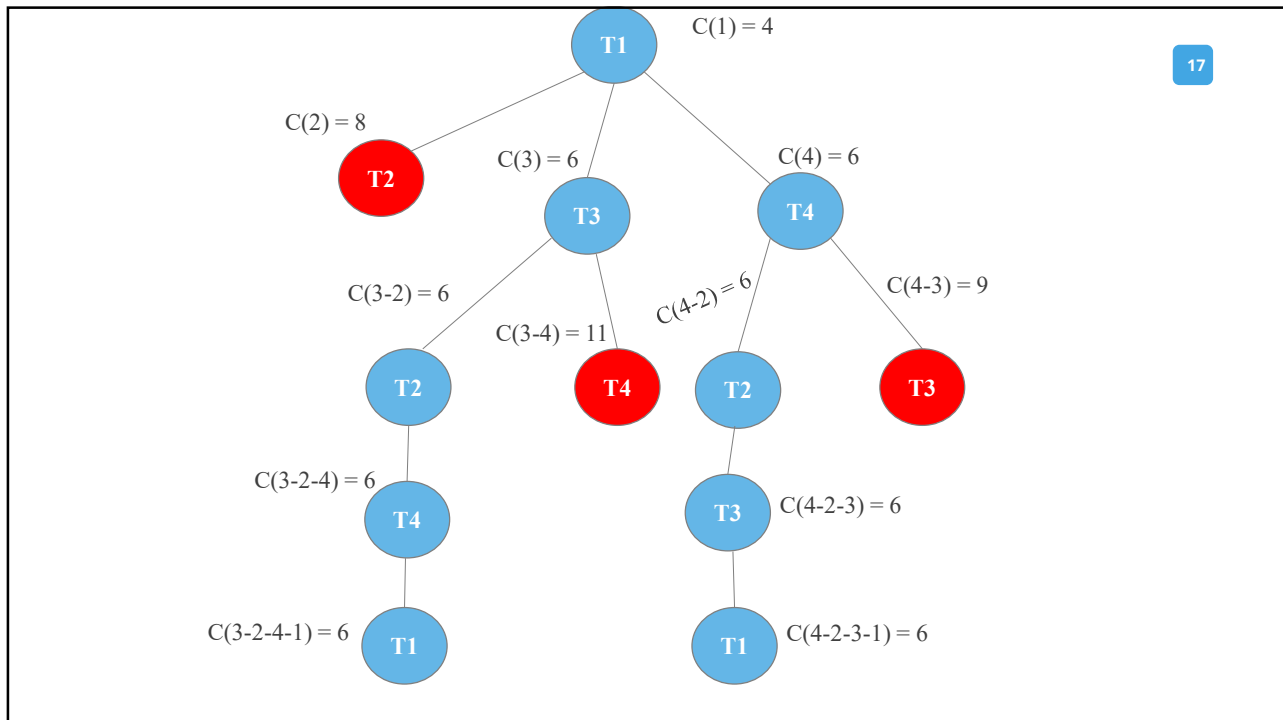
Path 1-3-2-4

	1	2	3	4	
1	∞	∞	∞	∞	0
2	∞	∞	∞	∞	0
3	∞	∞	∞	∞	0
4	0	∞	∞	∞	0
	0	0	0	0	0
$C(4-2) = 6 + 0 + 0 = 6$					

Path 1-4-2-3

	1	2	3	4	
1	∞	∞	∞	∞	0
2	∞	∞	∞	∞	0
3	0	∞	∞	∞	0
4	∞	∞	∞	∞	0
	0	0	0	0	0
$C(4-2) = 6 + 0 + 0 = 6$					

16

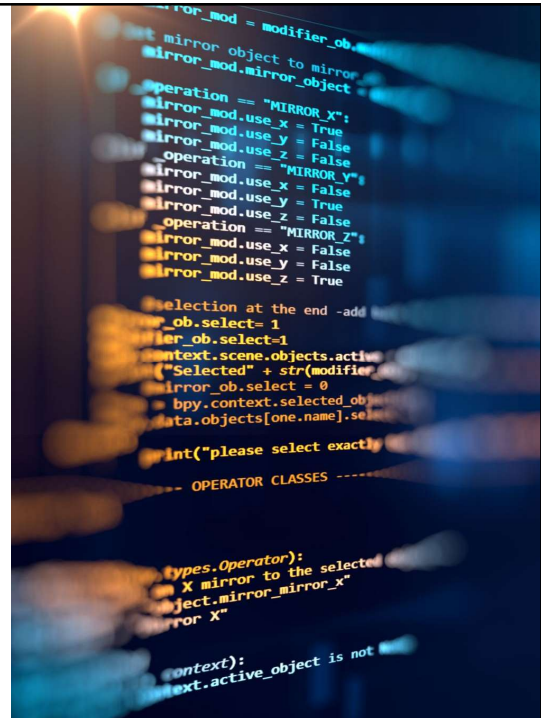


17

Mã giả

```

Try (i)
for (j = 1 -> n):
if (Chấp nhận được):
    Xác định xi theo j;
    Ghi nhận trạng thái mới;
    if (i == n)
        Cập nhật lời giải (phương án) tối ưu;
    else:
        Xác định cận g (x1,...,xi)
        if g (x1,...,xi) ≤ f*
            Try (i+1);
//Trả bài toán về trạng thái cũ
  
```



18


19



Đặc trưng quan trọng
nhất của B&B là gì?

19

20



Giống và khác nhau
giữa B&B và Greedy?

20

Ưu điểm

21



Giảm chi phí

Giảm chi phí nhờ việc cắt bỏ các bước đi không cần thiết

Nhược điểm



Phụ thuộc hàm

Việc tối ưu phụ thuộc vào hàm g , mỗi bài toán sẽ có một hàm g khác nhau



Điều kiện hàm g

- Việc tính giá trị của g phải đơn giản hơn việc giải bài toán tổ hợp tìm $\min\{f(a): a = (a_1, \dots, a_n) \text{ thuộc } X, x_i = a_i, i = 1, \dots, n\}$
- Giá trị của $g(a_1, a_2, \dots, a_k)$ phải sát với các giá trị của \min

21

Ứng dụng



Bài toán phân công

Phân công công việc sao cho chi phí thực hiện là thấp nhất



Input

Danh sách công nhân và công việc
Ma trận chi phí thực hiện



Output

Bảng phân công công việc thỏa yêu cầu



22

Các hướng tiếp cận

Brute force

Thuật toán Hungarian

Độ phức tạp trong trường hợp xấu nhất đạt $O(n^3)$

Thuật toán Hungarian

Tối ưu bằng việc loại bỏ các bước đi cần thiết



23

Đối với hàm chi phí

Chọn công nhân trước

Với mỗi công nhân, chọn công việc có chi phí tối thiểu từ danh sách công việc chưa được giao

Chọn công việc trước

Với mỗi công việc, chọn công nhân có chi phí thực hiện công việc thấp nhất từ danh sách công nhân chưa được phân công



24

Các bước thực hiện

	Job 1	Job 2	Job 3	Job 4
A	9	2	7	8
B	6	4	3	7
C	5	8	1	8
D	7	6	9	4

	Job 1	Job 2	Job 3	Job 4
A	9	2	7	8
B	6	4	3	7
C	5	8	1	8
D	7	6	9	4

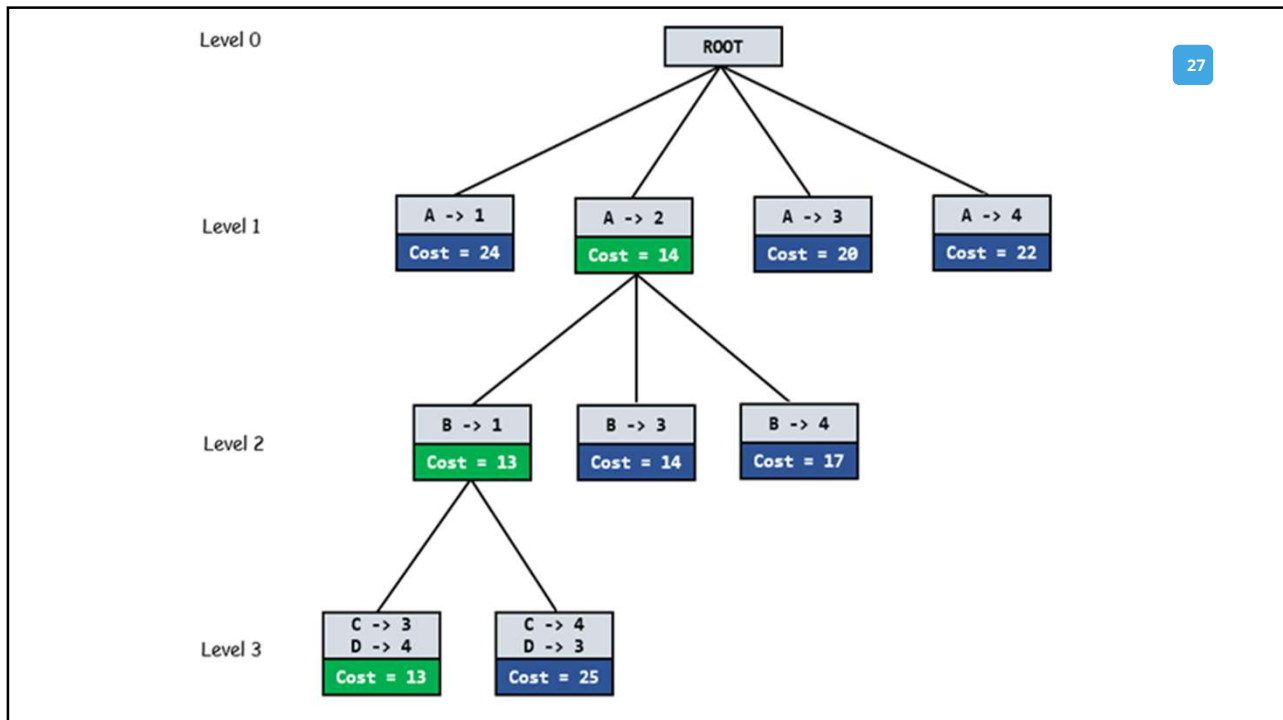
25

Các bước thực hiện

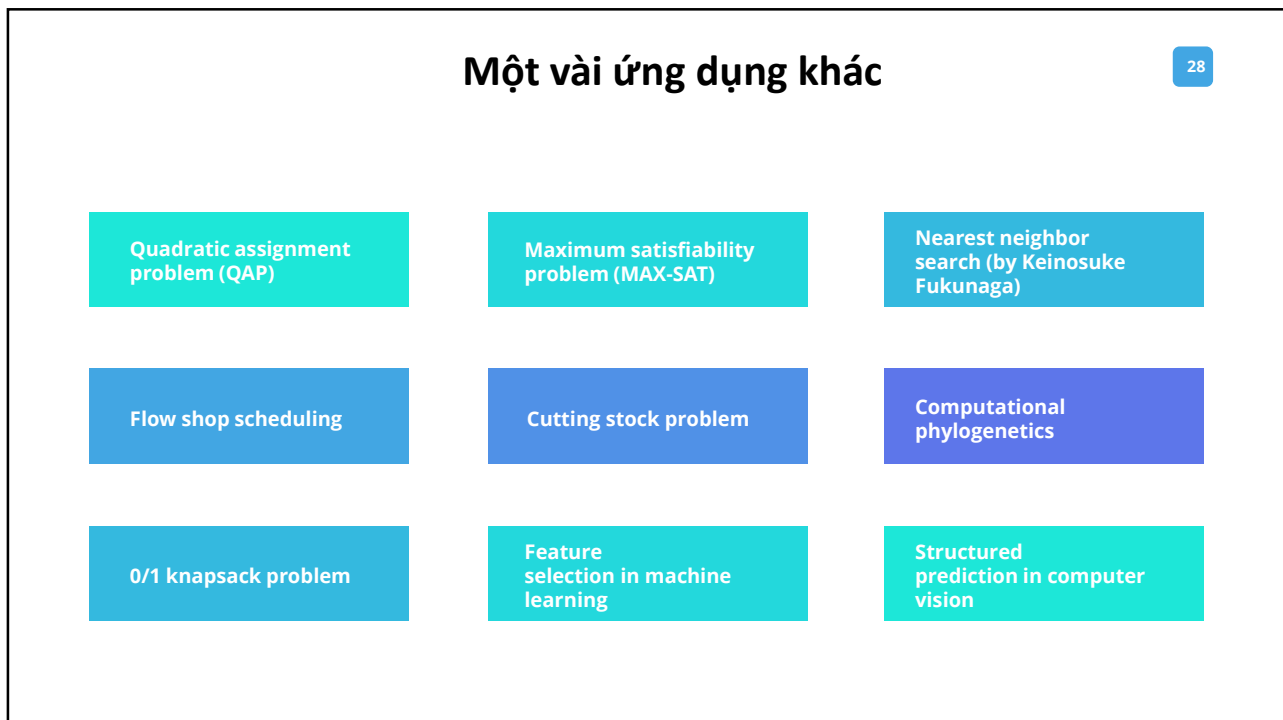
	Job 1	Job 2	Job 3	Job 4
A	9	2	7	8
B	6	4	3	7
C	5	8	1	8
D	7	6	9	4

	Job 1	Job 2	Job 3	Job 4
A	9	2	7	8
B	6	4	3	7
C	5	8	1	8
D	7	6	9	4

26



27



28

Kết luận

29

- 1 Giảm chi phí do bỏ đi các bước cần thiết
- 2 Bị phụ thuộc vào hàm tính cận
- 3 Là thuật toán quay lui
- 4 Tối ưu bằng việc quyết định quay lui sớm hơn
- 5 Là kĩ thuật đánh giá việc tiếp tục đào sâu tạo ra cấu hình tốt hơn cấu hình tốt nhất mà ta lưu trữ

29

ANY QUESTIONS?

30

30



Cảm ơn các bạn đã
lắng nghe!