Recursion, Backtracking, Branch and Bound (3)

Truong Ngoc Tuan

Nội dung

- Recursion
- Backtracking
- Generating Method
- Branch and Bound

Nội dung

- Branch and Bound
  - Lược đồ chung
  - Bài toán người đi du lịch
  - Bài toán cái túi

Nội dung

- Branch and Bound
  - Lược đồ chung
  - Bài toán người đi du lịch
  - Bài toán cái túi

Branch and Bound

❑ Mở đầu

- Phương pháp backtracking (vét cạn) có thể giải các bài toán tối ưu, bằng cách lựa chọn phương án tối ưu nhất trong các lời giải tìm được.
- Nhưng nhiều bài toán không gian lời giải quá lớn, không thể áp dụng được

=> ***Cần cải tiến thuật toán backtrack để hạn chế bớt việc duyệt các trường hợp***

Branch and Bound

❑ Mở đầu

- Phương pháp Branch and Bound là một cải tiến của phương pháp Backtrack
- ***Ý tưởng:***
  - Trong quá trình duyệt, luôn giữ lại 1 phương án mẫu (có thể xem là tối ưu ở thời điểm hiện tại)
  - Đánh giá phương án ngay trong thời điểm xây dựng các thành phần
  - Nếu tốt hơn thì lựa chọn, ngược lại chọn hướng khác

```
void Try(int i) {
    for (j = i->n ) {
        if(chấp nhận được){
            xác định x_i theo j;
            ghi nhận trạng thái mới
            if(i == n)
                Cập nhật Lời giải tối ưu
            else {
                Xác định cận g(x_1,…, x_i)
                if (g(x_1,…, x_i) <= f*)
                    Try (i+1);
            }
            Trả bài toán về trạng thái cũ
        }
    }
}
```

Branch and Bound

❑ Nhận xét:

- Thực chất phương pháp nhánh cận là tìm kiếm theo chiều sâu trên cây liệt kê lời giải như phương pháp quay lui
- Khi tìm được x_i mà đánh giá cận $g(x_1, \ldots, x_n) > f^*$ thì cắt bỏ các nhánh con từ x_i, quay ngược lên nhánh cha của nó là x_{i-1}

=> ***Xác định hàm đánh giá cận như thế nào?***

Nội dung

- Branch and Bound
  - Lược đồ chung
  - Bài toán người đi du lịch
  - Bài toán cái túi

Branch and Bound

❑ Bài toán

- Một người đi du lịch muốn tham quan N thành phố $\mathbf{T_1}$, .., $\mathbf{T_n}$. Xuất phát từ một thành phố nào đó, người du lịch muốn đi qua tất cả các thành phố còn lại, mỗi thành phố đi qua đúng 1 lần rồi quay lại thành phố xuất phát
- Gọi C_{ij} là chi phí đi từ thành phố $\mathbf{T_i}$ đến $\mathbf{T_j}$. Hãy tìm một hành trình thỏa mãn yêu cầu bài toán sao cho chi phí nhỏ nhất

Branch and Bound

❑ Ý tưởng

- Số lượng hoán vị của tập {1, .., n) là n!. Do đó có tất cả n! hành trình có thể xảy ra
- Nếu cố định 1 thành phố xuất phát, VD là $\mathbf{T_1}$ thì sẽ có (n-1)! hành trình

Branch and Bound

❑ Ý tưởng

- Bài toán chuyển về dạng
  - Tìm Min{f(a_2, …, a_n) với: (a_2, …, a_n) là hoán vị của {2, …, n}}
  - Trong đó

    $f(a_1, \ldots, a_n) = C_{1,a2} + C_{a2,a3} + \ldots + C_{an-1,an} + C_{an,1}$
- Cách giải bài toán sẽ kết hợp đánh giá nhánh cận trong quá trình liệt kê phương án của thuật toán backtrack

Branch and Bound

❑ Ý tưởng

- Input: $C = (C_{ij})$
- Output: $x^* = (x_1, \ldots, x_n)$ // hành trình tối ưu

  $f^* = f(x^*)$ // giá trị tối ưu
- Lời giải: Slide 07

Branch and Bound

❑ Ý tưởng

- Nếu cố định xuất phát từ T_1, duyệt lặp từ j=2
- Đánh giá nhánh cận
  - Đặt C_{min} = Min {$C_{i,j}$ i,j -> {1,…,n}}
  - Giả sử tại bước i đã tìm được lời giải bộ phận cấp i là $(x_1,\ldots x_i)$.
  - Đã đi qua đoạn đường $T_1 \rightarrow T_2 \rightarrow \ldots \rightarrow T_i$ với chi phí $Si = C_{1,x2} + C_{x2,x3} + .. + C_{xi-1,xi}$

Branch and Bound

❑ Ý tưởng

- Ta cần phải đi qua n-i+1 đoạn đường nữa
- Đánh giá nhánh cận đoạn còn lại
  - $g(x_1,\ldots,x_i) = S_i + (n\text{-}i+1)C_{min}$
- Điều kiện chấp nhận được của j là thành phố T_j chưa đi qua. Dùng mảng đánh dấu *visited[j]*

Branch and Bound

❑ Ý tưởng

- Xác định x_i theo j bằng lệnh gán: $x_i = j$

  Cập nhật trạng thái mới: visited[j] = 1

  Cập nhật lại chi phí: $S = S + C_{xi-1,xi}$

Branch and Bound

❑ Ý tưởng

- Cập nhật lời giải tối ưu:

  $SUM = S + C_{xn,1}$

  Nếu (SUM < f*) thì

  Lgtu = x;

  f* = SUM

Branch and Bound

❑ Ý tưởng

- Thao tác hủy trạng thái

  visited[j] = 0

- Trả lại chi phí cũ:

  $S = S - C_{xi-1,xi}$

Branch and Bound

- Cài đặt

```
void Try(int i)
{
        int j, Tong, g;
        for (j = 2; j <= n; j++)
                if(!Daxet[j])
                {
                        x[i] = j;
                        Daxet[j] = 1;
                        S = S + C[x[i-1]][x[i]];
                        if(i==n)        //Cap nhat hanh trinh toi uu
                        {
                                Tong = S + C[x[n]][x[1]];
                                if(Tong < Gttu)
                                {
                                        Gan(Httu,x,n);
                                        Gttu = Tong;
                                }
                        }
                        else
                        {
                                g = S + (n-i+1)*Cmin; //Danh gia can
                                if ( g < Gttu)
                                        Try(i+1);
                        }
                        S = S - C[x[i-1]][x[i]];
                        Daxet[j] = 0;
                }
}
```

https://ideone.com/dzNLuc

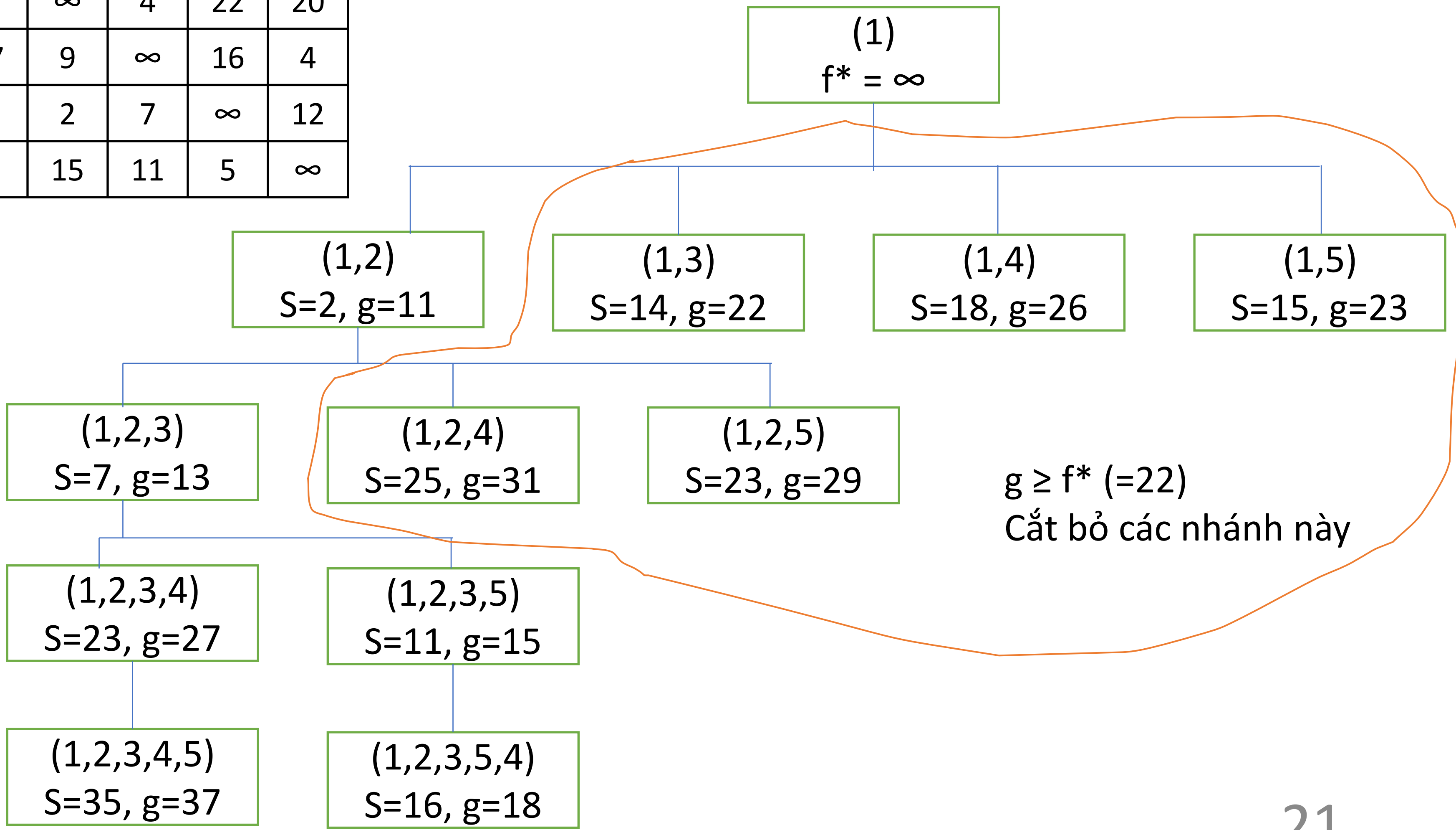
Branch and Bound

- Minh họa

	1	2	3	4	5
1	∞	3	14	18	15
2	3	∞	4	22	20
3	17	9	∞	16	4
4	6	2	7	∞	12
5	9	15	11	5	∞

Branch and Bound

	1	2	3	4	5
1	∞	3	14	18	15
2	3	∞	4	22	20
3	17	9	∞	16	4
4	6	2	7	∞	12
5	9	15	11	5	∞



Nội dung

- Branch and Bound
  - Lược đồ chung
  - Bài toán người đi du lịch
  - Bài toán cái túi

Branch and Bound

❑ Bài toán

- Có n đồ vật mỗi loại có số lượng không giới hạn. Đồ vật loại i có trọng lượng W_i và giá trị V_i.
- Cần chọn các vậy này đặt vào một chiếc túi xách có giới hạn trọng lượng M, sao cho tổng giá trị sử dụng các vật được chọn là lớn nhất

Branch and Bound

❑ Ý tưởng

- Đặt

$D = \{u = (u_1, \dots, u_n) \in N^n: \sum_{i=1}^{n} u_i w_i \le m\}$

$f : D \rightarrow R^{+}$

$(u_1, \dots, u_n) \rightarrow f(u_1, \dots, u_n) = \sum_{i=1}^{n} u_i w_i$

- Bài toán cái túi xách chuyển về bài toán sau:

Tìm x* ∈ D : f* = f(x*) = {f(u): u ∈ D}

Branch and Bound

❑ Ý tưởng

- Cách chọn đồ vật

  Xét mảng đơn giá: $Dg = \left(\frac{v_1}{w_1}, \dots, \frac{v_n}{w_n}\right)$

- Ta chọn đồ vật theo giá trị giảm dần

Branch and Bound

❑ Ý tưởng

- Đánh giá cận trên:

  Giả sử đã tìm được lời giải bộ phận: $(x_1, \ldots, x_n)$.

  Khi đó:

  - Giá trị $S = \sum_{j=1}^{i} x_j v_j = S + x_j v_j$

Branch and Bound

❑ Ý tưởng

- Đánh giá cận trên:

  Giả sử đã tìm được lời giải bộ phận: $(x_1,\ldots,x_n)$.

  Khi đó:

  - Trọng lượng $\mathrm{TL} = \sum_{j=1}^{i} x_j w_j = TL + x_j w_j$

Branch and Bound

❑ Ý tưởng

- Đánh giá cận trên:

  Giả sử đã tìm được lời giải bộ phận: $(x_1,\ldots,x_n)$.

  Khi đó:

  - Giới hạn trọng lượng còn lại

$$\mathrm{M} - \mathrm{TL} = M - \sum_{j=1}^{i} x_j w_j$$

Branch and Bound

❑ Ý tưởng

- Ta có

$$Max\{f(u): u=(u_1,\cdots,u_n)\in D; u_j = x_j, \forall j=\overline{1,i}\} =$$

$$Max\left\{S+\sum_{j=i+1}^{n} u_j v_j : \sum_{j=i+1}^{n} u_j w_j \le m_i\right\} =$$

$$S + Max\left\{\sum_{j=i+1}^{n} u_j v_j : \sum_{j=i+1}^{n} u_j w_j \le m_i\right\} \le S + v_{i+1} * \left(\frac{m_i}{w_{i+1}}\right)$$

- Lời giải bộ phận thứ i:

$$g(x_1,\ldots,x_n) = S + v_{i+1} * \left(\frac{m_i}{w_{i+1}}\right)$$

- Giá trị có thể chấp nhận được cho x_{j+1} là:

$$t = 0 \rightarrow \left(\frac{m_i}{w_{i+1}}\right)$$

Branch and Bound

❑ Ý tưởng

- Ghi nhận trạng thái khi xác định được x_i

  $S = S + x_i v_i$

  $T = T + x_i w_i$

- Trả lại trạng thái cũ cho bài toán

  $S = S - x_i v_i$

  $T = T - x_i w_i$

Branch and Bound

❑ Ý tưởng

- Khi tìm được lời giải, so sánh lời giải này với lời giải mà ta coi là tốt nhất vào thời điểm hiện tại để chọn lời giải tối ưu
- Khởi tạo ban đầu

x = 0;* *// Lời giải tối ưu của bài toán*

f = f(x*) = 0;* *// Giá trị tối ưu*

S = 0; *// Giá trị thu được từng bước của túi*

TL = 0; *// Trọng lượng xếp vào túi từng bước*

Branch and Bound

❑ Cài đặt

Input $m,$

$v=(v_1, \ldots, v_n) : v_i \in R, \forall i;$

$w=(w_1,\ldots, w_n) : w_i \in R, \forall i;$

Output $x^* = (x_1,\ldots, x_n) : x_i \in N, \forall i;$

$$f^* = f(x^*) = Max\left\{\sum_{i=1}^{n} u_i v_i : \sum_{i=1}^{n} u_i w_i \le m, u_i \in N, \forall i \in \overline{1,n}\right\}$$

Branch and Bound

- Cài đặt

```
Try(i)≡
    t = (m-TL)/w_i ;
    for (j = t; j >=0 ; j--)
    {
        x_i = j;
        TL = TL + w_i*x_i ;
        S  = S  + v_i*x_i;
        if(i==n)        //Cap nhat toi uu
        {
            if(S > f*)
            {
                x* = x;
                f* = S;
            }
        }
        else
        {
            g = S + v_{i+1}*(m-TL)/w_{i+1}; //Danh gia can
            if ( g > f*)
                Try(i+1);
        }
        TL = TL – w_i*x_i;
        S  = S  - v_i*x_i;
    }
```

https://ideone.com/b3rVJw

Branch and Bound

❑ Minh họa

$m = 8$

i	1	2	3	4
w	5	3	2	4
v	10	5	3	6


Q
A


THANK
YOU