Analysis and Design of Algorithms

Lecture 14
Branch and Bound

Lecturer: Tong Minh Duc

Nội dung

1. Lược đồ chung
2. Bài toán người du lịch
3. Bài toán cái túi

Nội dung

1. **Lược đồ chung**
2. Bài toán người du lịch
3. Bài toán cái túi

Giới thiệu

Phương pháp quay lui, vét cạn có thể giải các bài toán tối ưu, bằng cách lựa chọn phương án tối ưu trong tất cả các lời giải tìm được. Nhưng nhiều bài toán không gian các lời giải là quá lớn, nên áp dụng phương pháp pháp quay lui khó đảm bảo về thời gian cũng như kỹ thuật. Cho nên ta cần phải cải tiến thuật toán quay lui để hạn chế bớt việc duyệt các phương án. Có nhiều cách cải tiến, trong đó có phương pháp nhánh cận.

Phương pháp nhánh cận là một cải tiến của phương pháp quay lui, dùng để tìm lời giải tối ưu của bài toán. Ý tưởng chính của nó như sau :

Trong quá trình duyệt ta luôn giữ lại một phương án mẫu (có thể xem là lời giải tối ưu cục bộ – chẳng hạn có giá nhỏ nhất tại thời điểm đó). Đánh giá nhánh cận là phương pháp tính giá của phương án ngay trong quá trình xây dựng các thành phần của phương án theo hướng đang xây dựng có thể tốt hơn phương án mẫu hay không. Nếu không ta lựa chọn theo hướng khác.

Ý tưởng

Giả sử bài toán tối ưu cho là :

Tìm Min{f(x) : x $\in$ D};

Với X = $\left\{a = (a_1, \cdots, a_n) \in \prod_{i=1}^{n} A_i : P(x)\right\}$; $|A_i| < \infty; \forall i = \overline{1,n}$. P là một tính chất trên $\prod_{i=1}^{n} A_i$.

Nghiệm của bài toán nếu có sẽ được biểu diễn dưới dạng :x = (x_1,...,x_n)

Trong quá trình liệt kê theo phương pháp quay lui, ta xây dựng dần các thành phần của nghiệm.

Một bộ phận i thành phần (x_1, .., x_i) sẽ gọi là một lời giải (phương án) bộ phận cấp i. Ta gọi X_i là tập các lời giải bộ phận cấp i, $\forall i = \overline{1,n}$.

Đánh giá cận là tìm một hàm g xác định trên các X_i sao cho :

$g(x_1, \cdots, x_i) \le Min\{f(a) : a = (a_1, \cdots, a_n) \in X, x_i = a_i, \forall i = \overline{1,i}\}$

Bất đẳng thức này có nghĩa là giá trị $g(x_1, \cdots, x_i)$ không lớn hơn giá trị của các phương án mở rộng từ lời giải bộ phận $(x_1, \cdots, x_i)$.

Sau khi tìm được hàm đánh giá cận g, ta dùng g để giảm bớt chi phí duyệt các phương án theo phương pháp quay lui.

Giả sử x* là lời giải tốt nhất hiện có (phương án mẫu), còn f* là giá trị tốt nhất tương ứng f* = f(x*).

Nếu $g(x_1, \cdots, x_i)$ > f* thì :

f* < $g(x_1, \cdots, x_i) \le Min\{f(a) : a = (a_1, \cdots, a_n) \in X, x_i = a_i, \forall i = \overline{1,i}\}$

Nên chắc rằng các lời giải mở rộng từ $(x_1, \cdots, x_i)$ sẽ không tốt hơn phương án mẫu, do đó có thể bỏ đi không cần phát triển lời giải bộ phận $(x_1, \cdots, x_i)$ để tìm lời giải tối ưu của bài toán.

Lược đồ chung

Thủ tục quay lui sửa lại thành thủ tục nhánh cận

```
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 tối ưu ;
                        else
                        {
                                Xác định cận g(x1,···,xi) ;
                                if( g(x1,···,xi) ≤ f* )
                                        Try (i+1);
                        }
                        // Trả bài toán về trạng thái cũ
                }
```

2/22/2023 7

Thực chất của phương pháp nhánh cận là tìm kiếm theo chiều sâu trên cây liệt kê lời giải như phương pháp quay lui, chỉ khác có một điều là khi tìm được x_i mà đánh giá cận $g(x_1,\cdots,x_i) >$ f* thì ta cắt bỏ các nhánh con từ x_i đi xuống, mà quay lên ngay cha của nó là x_{i-1}.

Vấn đề là xác định hàm đánh giá cận như thế nào ?

2/22/2023 8

Nội dung

1. Lược đồ chung
2. **Bài toán người du lịch**
3. Bài toán cái túi

Bài toán

Một người du lịch muốn tham quan n thành phố $T_1,..,T_n$. 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 trở lại thành phố xuất phát.

Gọi C_{ij} là chi phí đi từ thành phố T_i đến T_j . Hãy tìm một hành trình thỏa yêu cầu bài toán sao cho chi phí là nhỏ nhất.

Ý tưởng

Gọi π là một hoán vị của {1,.., n} thì một hành trình thỏ yêu cầu bài toán có dạng : $T_{\pi(1)} \to T_{\pi(2)} \to \ldots \to T_{\pi(n)}$.

Nên có tất cả n! hành trình như thế.

Nếu ta cố định một thành phố xuất phát, chẳng hạn T_1, thì có (n-1)! hành trình.

Bài toán chuyển về dạng :

Tìm Min{f(a_2,.., a_n) : (a_2,.., a_n) là hoán vị của {2,..,n}}.

Với $f(a_1,\cdots,a_n) = C_{1,a_2} + C_{a_2,a_3} + \cdots + C_{a_{n-1},a_n} + C_{a_n 1}$

Cách giải bài toán sẽ kết hợp đánh giá nhánh cận trong quá trình liệt kê phương án của thuật toán quay lui.


```
Input       C = (Cij )
Output      - x* = (x1,...,xn)      // Hành trình tối ưu
            - f*  = f(x*)           // Giá trị tối ưu
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 tối ưu;
                  else
                  {
                        Xác định cận  g(x1,···,xi) ;
                        if( g(x1,···,xi)  ≤ f* )
                              Try (i+1);
                  }
                  // Trả bài toán về trạng thái cũ
            }
```

- Nếu ta cố định xuất phát tư T_1, ta duyệt vòng lặp từ $j = 2$.
- Đánh giá nhánh cận :

Đặt : CMin = Min{C_{ij} : i, j $\in$ {1,..,n}}

Giả sử vào bước i ta tìm được lời giả bộ phận cấp i là $(x_1,..,x_i)$, tức là đã đi qua đoạn đường $T_1 \rightarrow T_2 \rightarrow \ldots \rightarrow T_i$, tương ứng với chi phí :

$S_i = C_{1x_2} + C_{x_2x_3} + \cdots + C_{x_{i-1}x_i}$

Để phát triển hành trình bộ phận này thành một hành trình đầy đủ, ta còn phải đi qua n-i+1 đoạn đường nữa, gồm n-i thành phố còn lại và đoạn quay lại T_1.

Do chi phí mỗi một trong n-i+1 đoạn còn lại không nhỏ hơn CMin, nên hàm đánh giá cận có thể xác định như sau :

$g(x_1,\cdots,x_i) = S_i + (n-i+1)CMin$

- Điều kiện chấp nhận được của j là thành phố T_j chưa đi qua.

Ta dùng một mảng logic Daxet[] để biểu diễn trạng thài này

$$Daxet[j] = \begin{cases} 1; T_j \text{ đã được đi qua} \\ 0; T_j \text{ chưa được đi qua} \end{cases}$$

Mảng Daxet[ ] phải được bằng 0 tất cả.

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

Cập nhật trạng thái mới : Daxet[j] = 1.

Cập nhật lại chi phí sau khi tìm được x_i : S = S+ $C_{x_{i-1}x_i}$

- Cập nhật lời giải tối ưu :

Tính chi phí hành trình vừa tìm được :

Tong = S + C_{x_n1};

Nếu (Tong < f*) thì

Lgtu = x;

f* = Tong;

- Thao tác huỷ bỏ trạng thái : Daxet[j] = 0.

Trả lại chi phí cũ : S = S- $C_{x_{i-1}x_i}$

```
Try(i)≡
    for (j = 2 → n)
        if(!Daxet[j])
        {   x[i] = j;
            Daxet[j] = 1;
            S = S + C[x[i-1]][x[i]];
            if(i==n)        //Cap nhat toi uu
            {
                Tong = S + C[x[n]][x[1]];
                if(Tong < f*)
                {
                    Lgtu = x;
                    f* = Tong;
                }
            }
            else
            {
                g = S + (n-i+1)*Cmin; //Danh gia can
                if ( g < f*)
                        Try(i+1);
            }
            S = S - C[x[i-1]][x[i]];
            Daxet[j] = 0;
        }
```


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;
    }
}
```



```
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;
                }
}
```

Khởi tạo

Khởi động các biến :

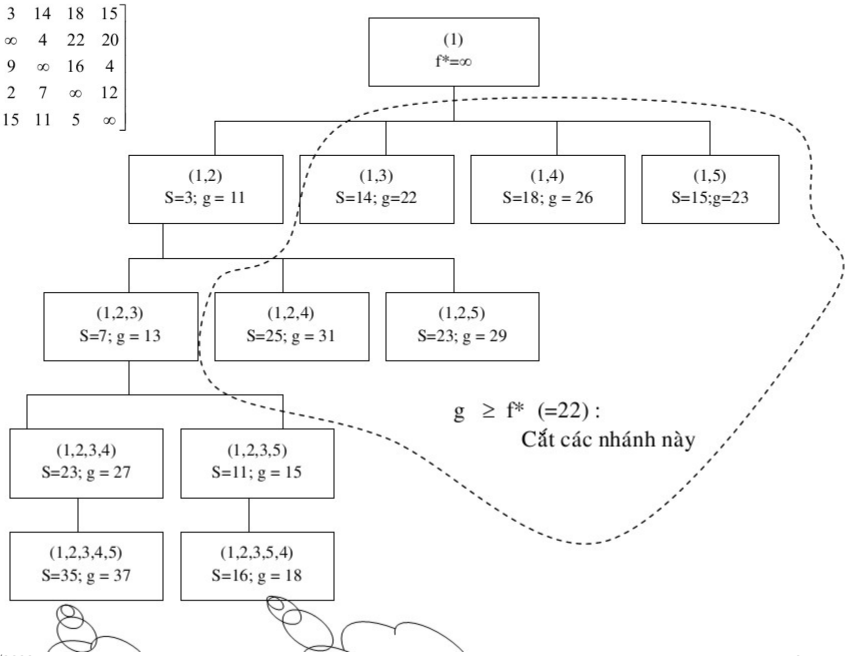
```
void Init()
{
        int i, j;
        Cmin = VC;//Chi phi nho nhat giua 2 thanh pho
        for(i = 1; i <= n; i++)
                Daxet[i] = 0;
        for(i = 1; i <= n; i++)
                for(j = 1; j <= n; j++)
                        if(Cmin>C[i][j])
                                Cmin = C[i][j];
        Gttu = VC;//Gia tri toi uu  f*
        S = 0;
        x[1] = 1; // Xuat phat tu dinh 1
}
```


Minh họa

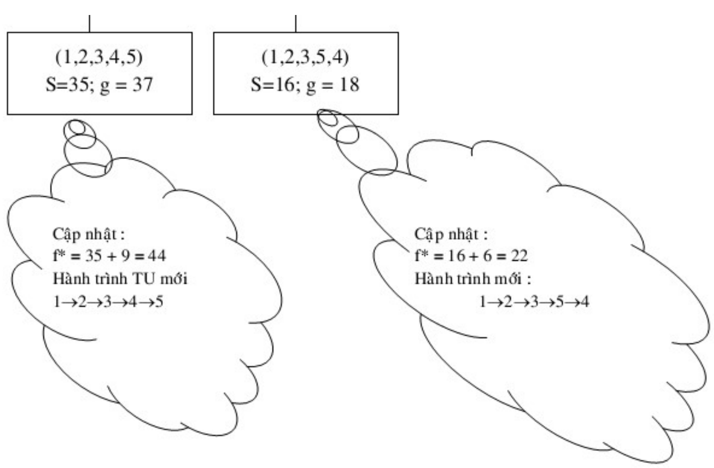
Ma trận chi phí

$$C = \begin{bmatrix} \infty & 3 & 14 & 18 & 15 \\ 3 & \infty & 4 & 22 & 20 \\ 17 & 9 & \infty & 16 & 4 \\ 6 & 2 & 7 & \infty & 12 \\ 9 & 15 & 11 & 5 & \infty \end{bmatrix}$$

$$C = \begin{bmatrix} \infty & 3 & 14 & 18 & 15 \\ 3 & \infty & 4 & 22 & 20 \\ 17 & 9 & \infty & 16 & 4 \\ 6 & 2 & 7 & \infty & 12 \\ 9 & 15 & 11 & 5 & \infty \end{bmatrix}$$



$$C = \begin{bmatrix} \infty & 3 & 14 & 18 & 15 \\ 3 & \infty & 4 & 22 & 20 \\ 17 & 9 & \infty & 16 & 4 \\ 6 & 2 & 7 & \infty & 12 \\ 9 & 15 & 11 & 5 & \infty \end{bmatrix}$$



Nội dung

1. Lược đồ chung
2. Bài toán người du lịch
3. **Bài toán cái túi**

Bài toán

Có n loại đồ vật, mỗi loại có số lượng không hạn chế. Đồ vật loại i, đặc trưng bởi trọng lượng Wi và giá trị sử dụng Vi, với mọi i $\in$ {1,..,n}.

Cần chọn các vật 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.

Ý tưởng

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

và : $f : D \to R^+$

$$(u_1, \cdots, u_n) \mapsto f(u_1, \cdots, u_n) = \sum_{i=1}^{n} u_i v_i \ ; (u_1, \cdots, u_n) \in D$$

Bài toán chiếc túi xách chuyển về bài toán sau :

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

Cho nên ta sẽ kết hợp đánh giá nhánh cận trong quá trình liệt kê các lời giải theo phương pháp quay lui.

Mô hình ban đầu có thể sử dụng như sau :

```
Try(i) ≡
    for(j = 1 → t)
        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 trên g;
                if( g(x_1,..., x_i) ≤ f*)
                    Try(i+1);
            }
            Trả lại trạng thái cũ cho bài toán;
        }
```

- Cách chọn vật :

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

  Ta chọn vật theo đơn giá giảm dần.

  Không mất tính tổng quát, ta giả sử các loại vật cho theo thứ tự giảm dần của đơn giá.

- Đánh giá cận trên :

  Giả sử đã tìm được lời giải bộ phận : $(x_1, \cdots, x_i)$. Khi đó :

- Giá trị của túi xách thu được : S = $\sum_{j=1}^{i} x_j v_j$ = S + $x_j v_j$.

- Tương ứng với trọng lượng các vật đã được xếp vào chiếc túi :

$$TL = \sum_{j=1}^{i} x_j w_j = TL + x_i w_i .$$

- Do đó, giới hạn trọng lượng của chiếc túi còn lại là : m – TL = m - $\sum_{j=1}^{i} x_j w_j$.

2/22/2023 27

Ta có :

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

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

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

Do đó, cận trên cho các lời giải bộ phận cấp i có thể xác định bởi :

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

- Theo biểu thức xác định cận trên g, các giá trị có thể chấp được cho x_{j+1} là :

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

2/22/2023 28

- Thao tác ghi nhận trạng thái mới khi xác định được x_i chẳng qua là cập nhật lại giá trị thu được và giới hạn trọng lượng mới của chiếc túi :

  $S = S + x_i v_i$

  $T = = T + x_i w_i$.
- Vì vậy, thao tác trả lại trạng thái cũ cho bài toán :

  $S = S - x_i v_i$

  $T = T - x_i w_i$.
- Cập nhật lời giải tối ưu :
  Khi tìm được một lời giải, ta 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.
- Các khởi tạo giá trị 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 chiếc túi.
  - TL = ; //Trọng lượng xếp vào chiếc túi từng bước.

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

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

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


Cài đặt

```
void Try(int i)
{
        int j, t, g;
        t = (int)((m-Tl)/w[i]);
        for (j = t; j >=0 ; j--)
        {
                x[i] = j;
                Tl = Tl + w[i]*x[i];    //Trong luong thu duoc
                S  = S  + v[i]*x[i];    //Gia tri thu duoc
                if(i==n)          //Cap nhat toi uu
                {
```

```
                    if(S > Gttu)
                    {
                            Gan();
                            Gttu = S;
                    }
            }
            else
            {
                    g = S + v[i+1]*(m-Tl)/w[i+1]; //Danh gia can
                    if ( g > Gttu)
                            Try(i+1);
            }
            Tl = Tl - w[i]*x[i];
            S  = S  - v[i]*x[i];
    }
}
```

```
void Try(int i)
{
        int j, t, g;
        t = (int)((m-Tl)/w[i]);
        for (j = t; j >=0 ; j--)
        {
                x[i] = j;
                Tl = Tl + w[i]*x[i];    //Trong luong thu duoc
                S  = S  + v[i]*x[i];    //Gia tri thu duoc
                if(i==n)          //Cap nhat toi uu
                {
                        if(S > Gttu)
                        {
                                Gan();
                                Gttu = S;
                        }
                }
                else
                {
                        g = S + v[i+1]*(m-Tl)/w[i+1]; //Danh gia can
                        if ( g > Gttu)
                                Try(i+1);
                }
                Tl = Tl - w[i]*x[i];
                S  = S  - v[i]*x[i];
        }
}
```

Cài đặt

- Khởi tạo

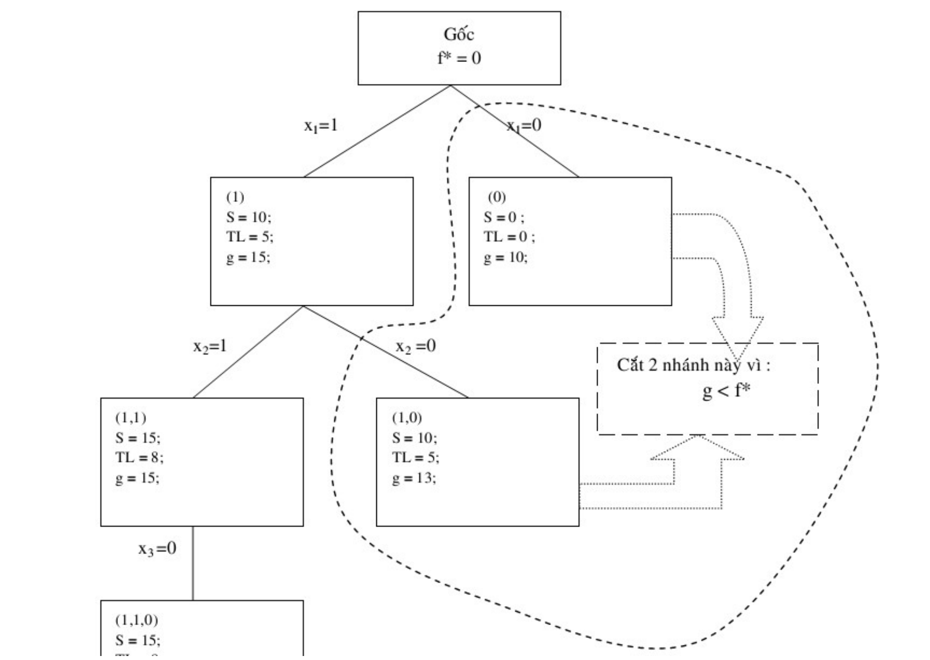
```
void Init()
{
        for (int i = 1; i <= n; i++)
        {
                Patu[i] = 0;
                x[i] = 0;
        }
        S = 0;
        Gttu = 0;
        Tl = 0;

}
```


Minh họa

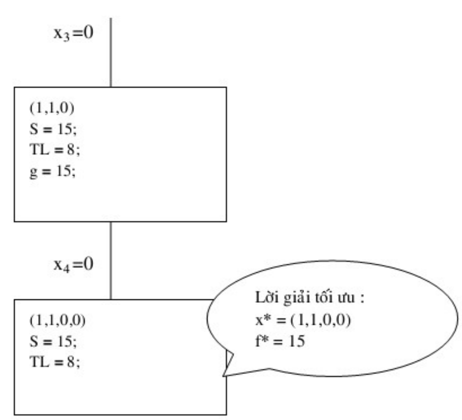
- Cho bài toán

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

m = 8



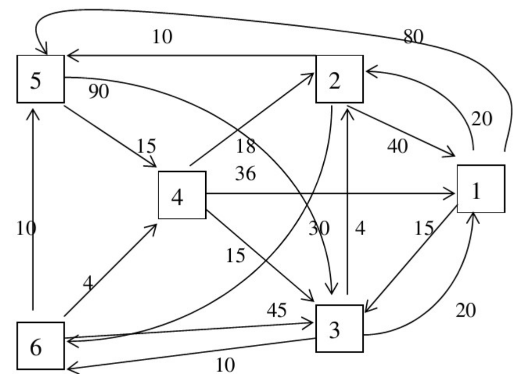
2/22/2023 37



2/22/2023 38

Bài tập

1. Thực hiện từng bước thuật toán nhánh cận cho bài toán người du lịch trên đồ thị sau.



Bài tập

2. Cài đặt thuật toán giải bài toán người du lịch (dựa trên thuật toán liệt kê các hoán vị) theo phương pháp nhánh cận. Đánh giá độ phức tạp thuật toán bằng lý thuyết, bằng thực nghiệm và so sánh.
3. Cài đặt thuật toán giải bài toán cái túi (dựa trên thuật toán liệt dãy nhị phân độ dài N) theo phương pháp nhánh cận. Đánh giá độ phức tạp thuật toán bằng lý thuyết, bằng thực nghiệm và so sánh