TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Toán rời rạc

Nguyễn Khánh Phương

Bộ môn Khoa học máy tính
E-mail: phuongnk@soict.hust.edu.vn

Phần thứ nhất

LÝ THUYẾT TỔ HỢP

Combinatorial Theory

Nguyễn Khánh Phương

Bộ môn Khoa học Máy tính,
Viện CNTT và Truyền thông,
Đại học Bách khoa Hà nội,
E-mail: phuongnk@soict.hust.edu.vn

Nội dung phần 1

Chương 0. Tập hợp

Chương 1. Bài toán đếm

Chương 2. Bài toán tồn tại

Chương 3. Bài toán liệt kê tổ hợp

Chương 4. Bài toán tối ưu tổ hợp

Nội dung chi tiết

1. **Giới thiệu bài toán**
2. Duyệt toàn bộ
3. Thuật toán nhánh cận

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

1. Giới thiệu bài toán

1.1 Bài toán tổng quát

1.2 Bài toán người du lịch

1.3 Bài toán cái túi

1.4 Bài toán đóng thùng

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

1.1. Bài toán tổng quát

- Trong rất nhiều vấn đề ứng dụng thực tế của tổ hợp, các cấu hình tổ hợp được gán cho một giá trị bằng số đánh giá giá trị sử dụng của cấu hình đối với mục đích sử dụng cụ thể nào đó.
- Khi đó xuất hiện bài toán: Hãy lựa chọn trong số các cấu hình tổ hợp chấp nhận được cấu hình có giá trị sử dụng tốt nhất. Các bài toán như vậy chúng ta sẽ gọi là bài toán tối ưu tổ hợp.

Phát biểu bài toán tối ưu tổ hợp

Dưới dạng tổng quát bài toán tối ưu tổ hợp có thể phát biểu như sau:

Tìm cực tiểu (hay cực đại) của hàm

$$f(x) \to \min (\max),$$

với điều kiện

$$x \in D,$$

trong đó D là tập hữu hạn phần tử.

Các thuật ngữ:

- $f(x)$ - hàm mục tiêu của bài toán,
- $x \in D$ - phương án
- D - tập các phương án của bài toán.
- Thông thường tập D được mô tả như là tập các cấu hình tổ hợp thoả mãn một số tính chất cho trước nào đó.
- Phương án $x^* \in D$ đem lại giá trị nhỏ nhất (lớn nhất) cho hàm mục tiêu được gọi là ***phương án tối ưu***, khi đó giá trị $f^* = f(x^*)$ được gọi là ***giá trị tối ưu*** của bài toán.

1. Giới thiệu bài toán

1.1 Bài toán tổng quát

1.2 Bài toán người du lịch

1.3 Bài toán cái túi

1.4 Bài toán đóng thùng

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Bài toán người du lịch (Traveling Salesman Problem – TSP)

- Một người du lịch muốn đi tham quan n thành phố T_1, T_2, ..., T_n.
- *Hành trình là cách đi xuất phát từ một thành phố nào đó đi qua tất cả các thành phố còn lại, mỗi thành phố đúng một lần, rồi quay trở lại thành phố xuất phát.*
- Biết c_{ij} là chi phí đi từ thành phố T_i đến thành phố T_j ($i, j = 1, 2,..., n$),
- Tìm hành trình với tổng chi phí là nhỏ nhất.

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Bài toán người du lịch

Ta có tương ứng 1-1 giữa một hành trình

$$T_{\pi(1)} \to T_{\pi(2)} \to \ldots \to T_{\pi(n)} \to T_{\pi(1)}$$

với một hoán vị $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ của n số tự nhiên 1, 2,..., n.

Đặt chi phí hành trình

$$f(\pi) = c_{\pi(1),\pi(2)} + \ldots + c_{\pi(n-1),\pi(n)} + c_{\pi(n),\pi(1)}.$$

Ký hiệu:

Π - tập tất cả các hoán vị của n số tự nhiên 1, 2, ..., n.

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Bài toán người du lịch

- Khi đó bài toán người du lịch có thể phát biểu dưới dạng bài toán tối ưu tổ hợp sau:

$$\min \{ f(\pi) : \pi \in \Pi \}.$$

- Có thể thấy rằng tổng số hành trình của người du lịch là $n!$, trong đó chỉ có $(n\text{-}1)!$ hành trình thực sự khác nhau (bởi vì có thể xuất phát từ một thành phố bất kỳ, nên có thể cố định một thành phố nào đó là thành phố xuất phát).

1. Giới thiệu bài toán

1.1 Bài toán tổng quát

1.2 Bài toán người du lịch

1.3 Bài toán cái túi

1.4 Bài toán đóng thùng

1.3. Bài toán cái túi (Knapsack Problem)

- Một nhà thám hiểm cần đem theo một cái túi có trọng lượng không quá b.
- Có n đồ vật có thể đem theo. Đồ vật thứ j có
  - trọng lượng là w_j và
  - giá trị sử dụng là v_j ($j = 1, 2, ..., n$).
- Hỏi rằng nhà thám hiểm cần đem theo các đồ vật nào để cho tổng giá trị sử dụng của các đồ vật đem theo là lớn nhất?

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Phát biểu bài toán

- Một phương án đem đồ của nhà thám hiểm có thể biểu diễn bởi vectơ nhị phân độ dài n: $x = (x_1, x_2, ..., x_n)$, trong đó $x_j = 1$ nếu đồ vật thứ j được đem theo và $x_j = 0$ nếu trái lại.
- Với phương án x, giá trị đồ vật đem theo là

$$f(x) = \sum_{j=1}^{n} v_j x_j,$$

tổng trọng lượng đồ vật đem theo là

$$g(x) = \sum_{j=1}^{n} w_j x_j$$

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

1.3. Bài toán cái túi

Bài toán cái túi có thể phát biểu dưới dạng bài toán tối ưu tổ hợp sau:

Trong số các vectơ nhị phân độ dài n thỏa mãn điều kiện $g(x) \le b$, hãy tìm vectơ x^* cho giá trị lớn nhất của hàm mục tiêu $f(x)$:

$$\max \{ f(x): x \in B^n, g(x) \le b \}.$$

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

1. Giới thiệu bài toán

1.1 Bài toán tổng quát

1.2 Bài toán người du lịch

1.3 Bài toán cái túi

1.4 Bài toán đóng thùng

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

1.4. Bài toán đóng thùng (Bin packing)

- Có n đồ vật với trọng lượng là w_1, w_2, ..., w_n. Cần tìm cách xếp các đồ vật này vào các cái thùng có cùng dung lượng là b sao cho số thùng cần sử dụng là nhỏ nhất có thể được.

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Phát biểu bài toán

- Ta có thể giả thiết là:

$$w_i \le b,\ i = 1, 2,.., n.$$

- Do đó số thùng cần sử dụng để chứa tất cả n đồ vật là không quá n. Vấn đề là cần số thùng ít nhất. Ta sẽ mở sẵn n cái thùng. Bài toán đặt ra là hãy xác định xem mỗi một trong số n đồ vật cần được xếp vào cái thùng nào trong số n cái thùng đã mở để cho số thùng chứa đồ là ít nhất.

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

1.4. Bài toán đóng thùng (Bin packing)

- Đưa vào biến Bun

  $x_{ij} = 1$, nếu đồ vật i được xếp vào thùng j,

  0, nếu trái lại.

Khi đó bài toán đóng thùng có thể phát biểu dưới dạng:

$$\sum_{j=1}^{n} sgn(\sum_{i=1}^{n} x_{ij}) \to \min,$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1,2,...,n$$

$$\sum_{i=1}^{n} w_i x_{ij} \le b, \quad j = 1,2,...,n;$$

$$x_{ij} \in \{0,1\}, \ i,j = 1,2,...,n.$$

$$\text{sgn}(x) = \begin{cases} -1 & \text{for } x < 0 \\ 0 & \text{for } x = 0 \\ 1 & \text{for } x > 0. \end{cases}$$

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Nội dung chi tiết

1. Giới thiệu bài toán
2. **Duyệt toàn bộ**
3. Thuật toán nhánh cận

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Mô tả phương pháp

- Một trong những phương pháp hiển nhiên nhất để giải bài toán tối ưu tổ hợp đặt ra là: Trên cơ sở các thuật toán liệt kê tổ hợp ta tiến hành duyệt từng phương án của bài toán, đối với mỗi phương án ta đều tính giá trị hàm mục tiêu tại nó, sau đó so sánh giá trị hàm mục tiêu tại tất cả các phương án được liệt kê để tìm ra phương án tối ưu.
- Phương pháp xây dựng theo nguyên tắc như vậy có tên gọi là phương pháp duyệt toàn bộ.

Ví dụ: Giải bài toán cái túi

- Xét bài toán cái túi biến 0-1:

$$\max\{f(x) = \sum_{j=1}^{n} v_j x_j : x \in D\},$$

$$\text{trong đó } D = \{x = (x_1, x_2, ..., x_n) \in B^n : \sum_{j=1}^{n} w_j x_j \le b\}$$

  - v_j, w_j, b là các số nguyên dương, j=1,2,…, n.
  - Cần có thuật toán liệt kê các phần tử của D

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Thuật toán quay lui
liệt kê các phương án chất đồ

- **Xây dựng S_k:**
  - S_1={ 0, t_1 }, với t_1=1 nếu $b \geq w_1$; t_1 = 0, nếu trái lại
  - Giả sử đã có phương án (x_1, …, x_{k-1}). Khi đó:
    - Dung lượng còn lại là:

    $b_{k-1} = b - w_1x_1 - \ldots - w_{k-1}x_{k-1}$

    - Giá trị của các đồ vật chất vào túi là

    $$f_{k-1} = v_1x_1 + \ldots + v_{k-1}x_{k-1}$$

    Do đó: S_k = {0, t_k}, với t_k=1 nếu $b_{k-1} \geq w_k$; t_k = 0, nếu trái lại
- **Mô tả S_k?**

```
for (y = 0; y++;y<= t_k )
```

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Chương trình trên C++

```
int x[20], xopt[20], c[20], w[20];
int  n,b, bk, fk, fopt;

void Nhapdl ( )
{
     <Nhập vào n, v, w, b>;
}
void Inkq ( )
{
    <Phương án tối ưu: xopt;
     Giá trị tối ưu: fopt>;
}
```

```
int main( )
{
    Nhapdl( );
    bk=b;
    fk= 0;
    fopt= 0;
    Try(1);
    InKq( );
}
```

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

```
void  Try(int k)
{
    int j, t;
   if (bk>=w[k])  t=1  else t=0;      //if: Trọng lượng còn lại của túi >= trọng lượng đồ vật k
   for (j= t; j--; j>=0)
   {
      x[k] = j;//j = 1 → x[k] = 1 tức là cho đồ vật k vào túi; j = 0→ x[k] = 0 tức là không cho đồ vật k vào túi
      bk= bk-w[k]*x[k];//bk: trọng lượng còn lại của túi
      fk= fk + v[k]*x[k];//fk: giá trị hiện tại của túi
       if  (k == n)  //nếu tất cả n đồ vật đều đã được xét
      {
          if (fk>fopt) {   //nếu giá trị hiện tại của túi > giá trị tốt nhất của túi hiện có
                xopt:=x; fopt:=fk;   //Cập nhật giá trị tốt nhất
          }
       }
      else Try(k+1);              //xét tiếp đồ vật thứ k+1
       bk= bk+w[k]*x[k];
       fk= fk - v[k]*x[k];
    }
}
```

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Bình luận

- Duyệt toàn bộ là khó có thể thực hiện được ngay cả trên những máy tính điện tử hiện đại nhất. Ví dụ để liệt kê hết

15! = 1 307 674 368 000

hoán vị trên máy tính điện tử với tốc độ tính toán 1 tỷ phép tính một giây, nếu để liệt kê một hoán vị cần phải làm 100 phép tính, thì ta cần một khoảng thời gian là 130767 giây > 36 tiếng đồng hồ!

20! ===> 7645 năm

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Bình luận

- Vì vậy cần phải có những biện pháp nhằm hạn chế việc tìm kiếm thì mới có hy vọng giải được các bài toán tối ưu tổ hợp thực tế. Tất nhiên để có thể đề ra những biện pháp như vậy cần phải nghiên cứu kỹ tính chất của bài toán tối ưu tổ hợp cụ thể.
- Nhờ những nghiên cứu như vậy, trong một số trường hợp cụ thể ta có thể xây dựng những thuật toán hiệu quả để giải bài toán đặt ra.

Bình luận

- Tuy nhiên phải nhấn mạnh rằng trong nhiều trường hợp (ví dụ trong các bài toán người du lịch, bài toán cái túi, bài toán đóng thùng) chúng ta chưa thể xây dựng được phương pháp hữu hiệu nào khác ngoài phương pháp duyệt toàn bộ.

Bình luận

- Khi đó, một vấn đề đặt ra là trong quá trình liệt kê lời giải ta cần tận dụng các thông tin đã tìm được để loại bỏ những phương án chắc chắn không phải là tối ưu.
- Trong mục tiếp theo chúng ta sẽ xét một sơ đồ tìm kiếm như vậy để giải các bài toán tối ưu tổ hợp mà trong tài liệu tham khảo được biết đến với tên gọi: thuật toán nhánh cận.

Nội dung chi tiết

1. Giới thiệu bài toán
2. Duyệt toàn bộ
3. **Thuật toán nhánh cận**

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

3. THUẬT TOÁN NHÁNH CẬN

(Branch and Bound Algorithm)

3. Thuật toán nhánh cận

3.1. Sơ đồ chung

3.2. Ví dụ

3.2.1. Bài toán người du lịch

3.2.2. Bài toán cái túi

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

3.1. Sơ đồ chung

- Thuật toán bao gồm hai thủ tục:
  - Phân nhánh (Branching Procedure)
  - Tính cận (Bounding Procedure)
- **Phân nhánh:** *Quá trình phân hoạch tập các phương án ra thành các tập con với kích thước càng ngày càng nhỏ cho đến khi thu được phân hoạch tập các phương án ra thành các tập con một phần tử*
- **Tính cận:** *Cần đưa ra cách tính cận cho giá trị hàm mục tiêu của bài toán trên mỗi tập con A trong phân hoạch của tập các phương án.*

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

3.1. Sơ đồ chung

- Ta sẽ mô tả tư tưởng của thuật toán trên mô hình bài toán tối ưu tổ hợp tổng quát sau

$$\min \{ f(x) : x \in D \},$$

trong đó D là tập hữu hạn phần tử.

- Giả thiết rằng tập D được mô tả như sau

$$D = \{x = (x_1, x_2, ..., x_n) \in A_1 \times A_2 \times ... \times A_n:$$

$$x \text{ thoả mãn tính chất } P\},$$

với $A_1, A_2, ..., A_n$ là các tập hữu hạn, còn P là tính chất trên tích Đề các $A_1 \times A_2 \times ... \times A_n$.

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Nhận xét

- Yêu cầu về mô tả của tập D là để có thể sử dụng thuật toán quay lui để liệt kê các phương án của bài toán.
- Bài toán

$$\max \{f(x): x \in D\}$$

là tương đương với bài toán

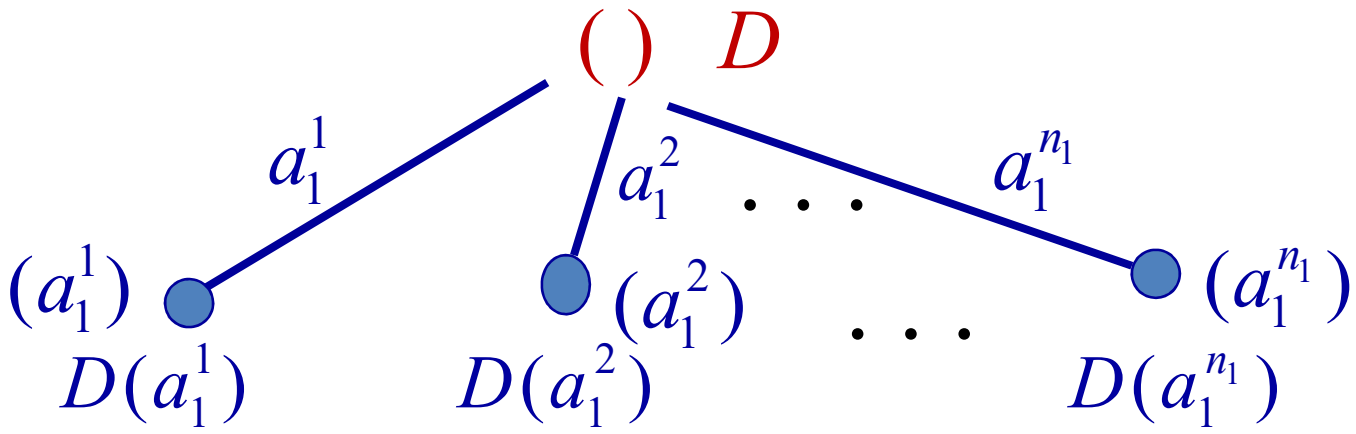
$$\min \{g(x): x \in D\}, \text{ trong đó } g(x) = -f(x)$$

Do đó ta có thể hạn chế ở việc xét bài toán min.

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Phân nhánh

Quá trình phân nhánh được thực hiện nhờ thuật toán quay lui:



trong đó $D(a_1^i) = \{x \in D : x_1 = a_1^i\}, i = 1, 2, ..., n_1$

là tập các phương án có thể phát triển từ pabp (a_1^i)

Ta có phân hoạch:

$$D = D(a_1^1) \cup D(a_1^2) \cup ... \cup D(a_1^{n_1})$$

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Phân nhánh

- Như vậy ta có thể đặt tương ứng mỗi phương án bộ phận $(a_1, a_2, ..., a_k)$ với một tập con các phương án của bài toán:

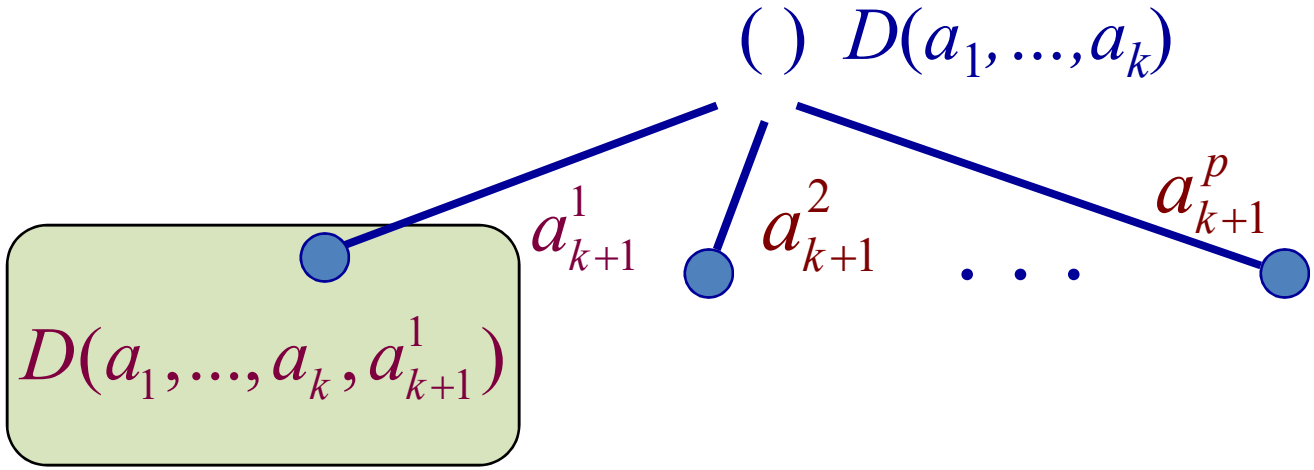
$$D(a_1,..., a_k)= \{ x \in D: x_i = a_i , i = 1,..., k \}.$$

- Ở bước tổng quát của thuật toán quay lui ta sẽ làm việc với phương án bộ phận $(a_1, a_2, ..., a_k)$ và xét các cách tiếp tục phát triển phương án này.
- Điều đó tương đương với việc phân hoạch tập D ra thành các tập con nhỏ hơn.

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Phân nhánh

- Quá trình phân nhánh có thể diễn tả như sau:



- Ta có phân hoạch:

$$D(a_1,\ldots,a_k)=\bigcup_{i=1}^{p} D(a_1,\ldots,a_k,a^i_{k+1})$$

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Tính cận

- Cần có hàm g xác định trên tập tất cả các phương án bộ phận của bài toán thoả mãn bất đẳng thức sau:

Giá trị hàm mục tiêu của mọi lời giải có k thành phần đầu tiên là $(a_1, a_2,\ldots,a_k)$

$$g(a_1,\ldots, a_k) \le \min\{f(x): x \in D(a_1, \ldots, a_k)\} \quad (*)$$

với mỗi phương án bộ phận cấp k $(a_1, a_2, \ldots, a_k)$, và với mọi $k = 1, 2, \ldots$

- Bất đẳng thức (*) có nghĩa là giá trị của hàm g tại phương án bộ phận $(a_1, a_2, \ldots, a_k)$ là không vượt quá giá trị nhỏ nhất của hàm mục tiêu của bài toán trên tập con các phương án

$$D(a_1,\ldots, a_k) = \{ x \in D: x_i = a_i , i = 1,\ldots, k \},$$

hay nói một cách khác, $g(a_1, a_2, \ldots, a_k)$ là **cận dưới** của giá trị hàm mục tiêu trên tập $D(a_1, a_2, \ldots, a_k)$.

Cắt nhánh nhờ sử dụng cận dưới

- Giả sử đã có hàm g. Ta xét cách sử dụng hàm này để giảm bớt khối lượng duyệt trong quá trình duyệt tất cả các phương án theo thuật toán quay lui.
- Trong quá trình liệt kê các phương án có thể đã thu được một số phương án của bài toán. Gọi x^* là phương án với giá trị hàm mục tiêu nhỏ nhất trong số các phương án đã tìm được, ký hiệu $f^* = f(x^*)$
- Ta sẽ gọi
  - x^* là phương án tốt nhất hiện có (phương án kỷ lục),
  - f^* là giá trị tốt nhất hiện có (giá trị kỷ lục).

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Cắt nhánh nhờ sử dụng cận dưới

Giả sử đã có f^* (*giá trị tốt nhất hiện có*), khi đó nếu

$$g(a_1, a_2, ..., a_k) > f^*$$

thì từ bất đẳng thức (*) suy ra

$$f^* < g(a_1,..., a_k) \le \min\{f(x): x \in D(a_1,...,a_k)\},$$

Vì thế tập $D(a_1,..., a_k)$ chắc chắn không chứa phương án tối ưu và có thể loại bỏ khỏi quá trình duyệt.

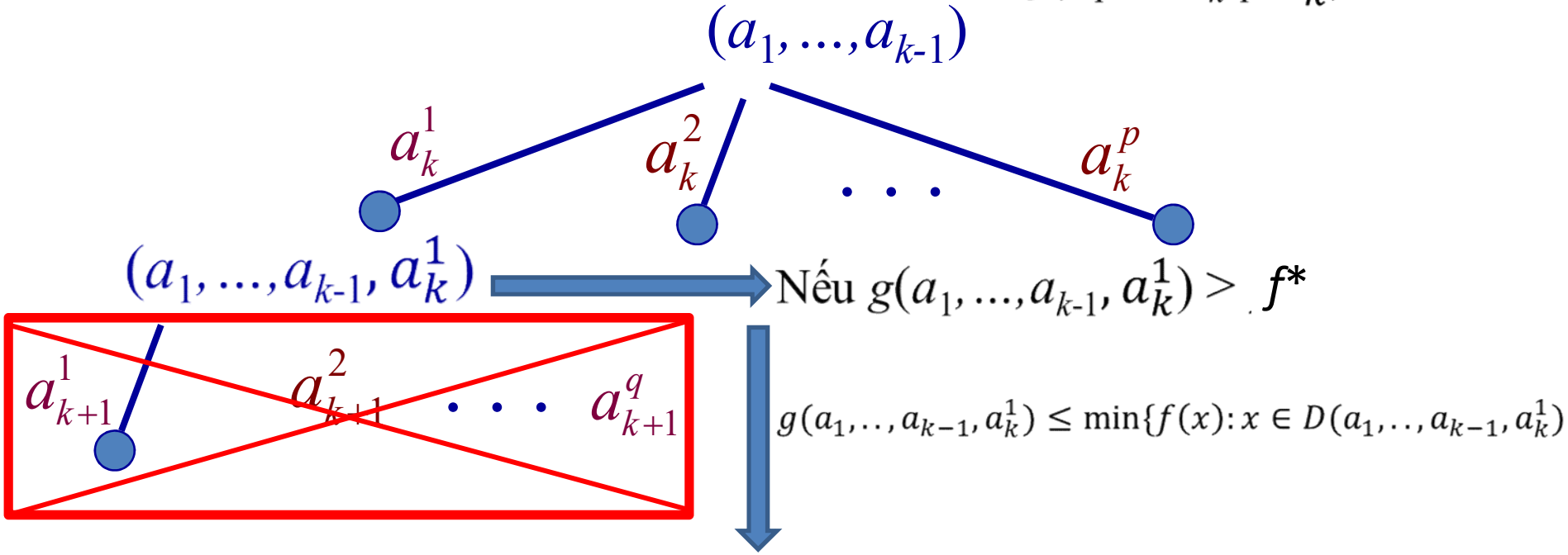
NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Cắt nhánh nhờ sử dụng cận dưới

f^* là giá trị hàm mục tiêu tốt nhất hiện có

Nếu $g(a_1, \ldots, a_{k-1}, a_k^2) > f^*$

Nếu $g(a_1, \ldots, a_{k-1}, a_k^p) > f^*$



→ mọi lời giải của bài toán có k thành phần đầu tiên là $(a_1, \ldots, a_{k-1}, a_k^1)$ chắc chắn đều có giá trị hàm mục tiêu > f^* → có thể loại bỏ toàn bộ nhánh này ra khỏi quá trình duyệt

$g(a_1, \ldots, a_k)$ là cận dưới của phương án bộ phận $(a_1, \ldots, a_k)$

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Thuật toán nhánh cận

```
void  Branch(int k)
{
    //Xây dựng x_k từ phương án bộ phận (x_1, x_2, ..., x_{k-1})
    for  a_k∈A_k
       if   (a_k∈S_k )
       {
          x_k =  a_k;
          if   (k == n)  < Cập nhật kỷ lục>;
          else if   (g(x_1,..., x_k)  ≤ f*) Branch(k+1);
        }
}

void  BranchAndBound ( )
{
    f* = +∞;
    // Nếu biết p/án x* nào đó thì đặt f* = f(x*)
    Branch(1);
    if  (f*  <  +∞)
            <f* là giá trị tối ưu, x* là p/án tối ưu >
   else  <  bài toán không có phương án >;
}
```

Chú ý rằng nếu trong thủ tục Branch ta thay câu lệnh

if $(k == n)$ < Cập nhật kỷ lục>;
else
if $(g(a_1,..., a_k) \le f^*)$ Branch(k+1);

bởi

if $(k == n)$ **then** < Cập nhật kỷ lục>;
else Branch(k+1);

thì ta thu được thuật toán **duyệt toàn bộ.**

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Chú ý:

$$g(a_1,..., a_k) \le \min\{f(x): x \in D(a_1, ..., a_k)\} \quad (*)$$

- Việc xây dựng hàm g phụ thuộc vào từng bài toán tối ưu tổ hợp cụ thể. Thông thường ta cố gắng xây dựng nó sao cho:
  - Việc tính giá trị của g phải đơn giản hơn việc giải bài toán tối ưu tổ hợp ở vế phải của (*).
  - Giá trị của $g(a_1,..., a_k)$ phải sát với giá trị của vế phải của (*).
- Rất tiếc là hai yêu cầu này trong thực tế thường đối lập nhau.

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

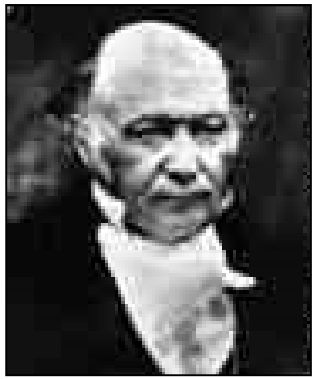
3. Thuật toán nhánh cận

3.1. Sơ đồ chung

3.2. Ví dụ

3.2.1. Bài toán người du lịch

3.2.2. Bài toán cái túi



Sir William Rowan Hamilton
1805 - 1865

Bài toán người du lịch (Traveling Salesman Problem – TSP)

- Một người du lịch muốn đi tham quan n thành phố 1, 2, …,n.
- *Hành trình là cách đi xuất phát từ một thành phố nào đó đi qua tất cả các thành phố còn lại, mỗi thành phố đúng một lần, rồi quay trở lại thành phố xuất phát.*
- Biết c_{ij} là chi phí đi từ thành phố i đến thành phố j ($i, j = 1, 2,..., n$),
- Tìm hành trình với tổng chi phí là nhỏ nhất.

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

3.2.1. Bài toán người du lịch

Cố định thành phố xuất phát là 1, bài toán người du lịch dẫn về bài toán:

- Tìm cực tiểu của hàm

$$f(1,x_2,\ldots, x_n) = c[1,x_2]+c[x_2,x_3]+\ldots+c[x_{n-1},x_n] + c[x_n,1]$$

với điều kiện

$(x_2, x_3, \ldots, x_n)$ là hoán vị của các số 2, ..., n.

Hàm cận dưới

- Ký hiệu

  $c_{min} = \min \{ c[i, j] , i, j = 1, 2, ..., n, i \neq j \}$

  là chi phí đi lại nhỏ nhất giữa các thành phố.

- Cần đánh giá cận dưới cho phương án bộ phận $(1, u_2, ..., u_k)$ tương ứng với hành trình bộ phận đã đi qua k thành phố:

$$1 \rightarrow u_2 \rightarrow \ldots \rightarrow u_{k-1} \rightarrow u_k$$

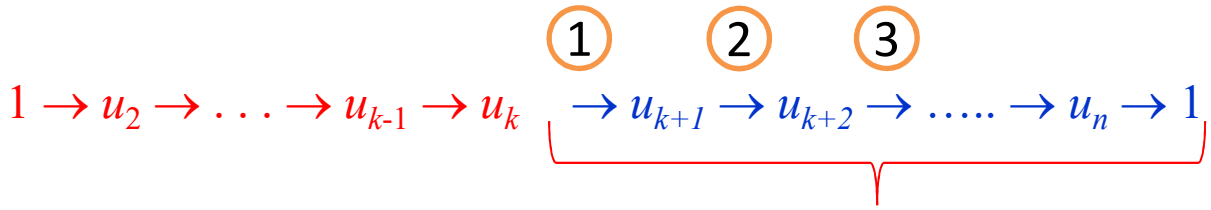
NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Hàm cận dưới

- Chi phí phải trả theo hành trình bộ phận này là

  $\sigma = c[1,u_2] + c[u_2, u_3] + \ldots + c[u_{k-1}, u_k]$.

- Để phát triển thành hành trình đầy đủ:



ta còn phải đi qua n-k+1 đoạn đường nữa, mỗi đoạn có chi phí không ít hơn c_{min}, nên cận dưới cho phương án bộ phận $(1, u_2, \ldots, u_k)$ có thể tính theo công thức:

$$g(1, u_2, \ldots, u_k) = \sigma + (n\text{-}k\text{+}1)\, c_{min}\,.$$

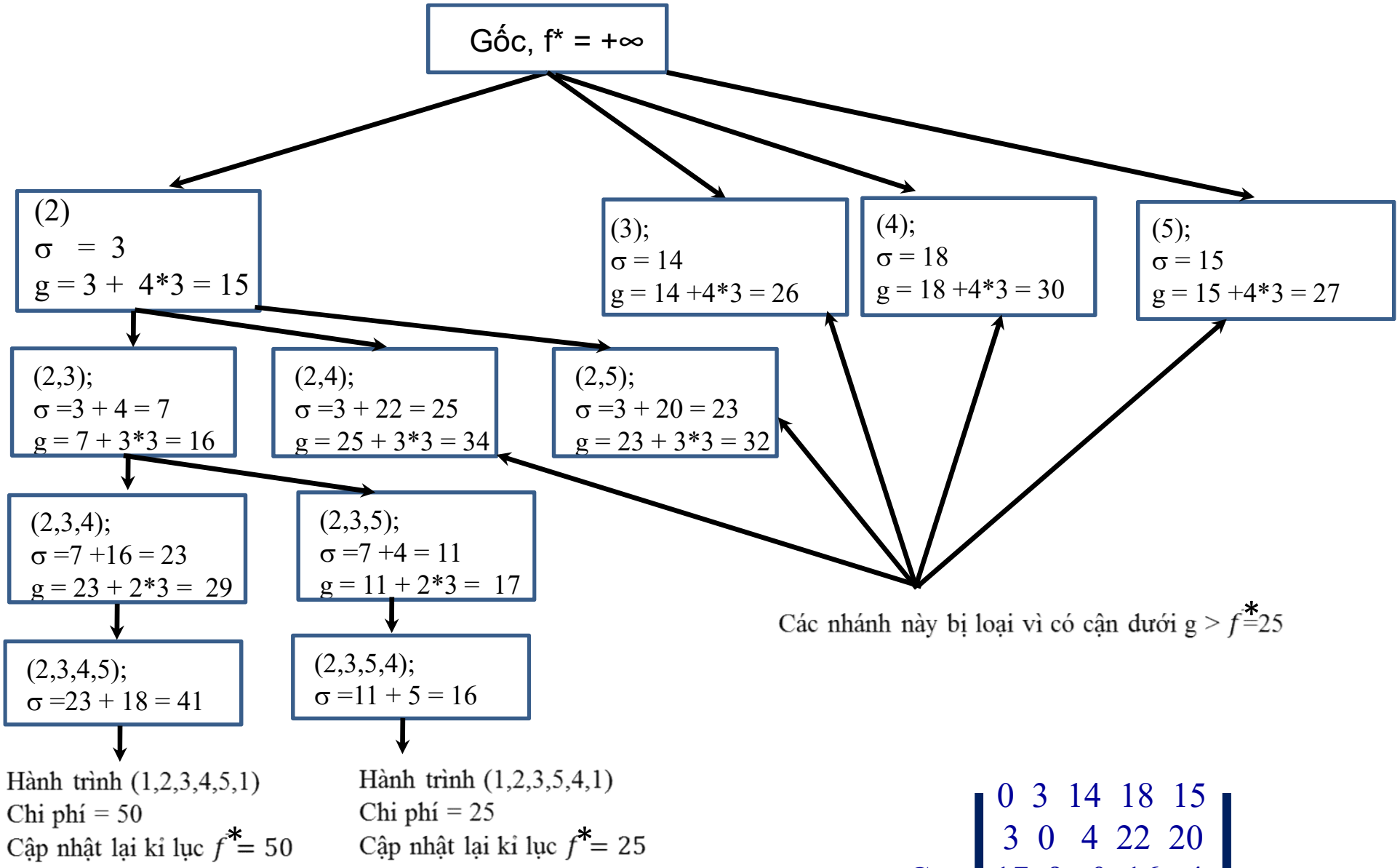
Ví dụ 1

Cho 5 thành phố {1, 2, 3, 4, 5}. Giải bài toán người du lịch xuất phát từ thành phố 1 với ma trận chi phí sau:

$$C = \begin{vmatrix} 0 & 3 & 14 & 18 & 15 \\ 3 & 0 & 4 & 22 & 20 \\ 17 & 9 & 0 & 16 & 4 \\ 9 & 20 & 7 & 0 & 18 \\ 9 & 15 & 11 & 5 & 0 \end{vmatrix}$$

Ví dụ 1

- Ta có $c_{min} = 3$. Quá trình thực hiện thuật toán được mô tả bởi cây tìm kiếm lời giải.
- Thông tin được ghi trong các ô trên hình vẽ theo thứ tự sau:
  - các thành phần của phương án bộ phận,
  - σ là chi phí theo hành trình bộ phận,
  - g – cận dưới của phương án bộ phận.



$$C = \begin{bmatrix} 0 & 3 & 14 & 18 & 15 \\ 3 & 0 & 4 & 22 & 20 \\ 17 & 9 & 0 & 16 & 4 \\ 9 & 20 & 7 & 0 & 18 \\ 9 & 15 & 11 & 5 & 0 \end{bmatrix}$$

Kết quả

Kết thúc thuật toán, ta thu được:

- phương án tối ưu (1, 2, 3, 5, 4, 1) tương ứng với hành trình

$$1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 4 \rightarrow 1 ,$$

- chi phí nhỏ nhất là 25.

Cài đặt

```
void  Branch(int k)
{
    for (int v = 2; v<=n;v++) {
       if (visited[v] == FALSE) {
             x_k = v; visited[v] = TRUE;
             f = f + c(x_{k-1},x_k);
             if (k == n) //Cập nhật kỉ lục:
              {  if (f + c(x_n,x_1) < f*) f* = f + c(x_n,x_1); }
             else {
                    g = f + (n-k + 1)*cmin; //tính cận
                    if (g < f* ) Branch(k + 1);
                  }
        }
        f = f - c(x_{k-1},x_k);
        visited[v] = FALSE;
    }
}
```

```
void  Branch(int k)
{
    //Xây dựng x_k từ phương án bộ phận (x_1, x_2, ..., x_{k-1})
    for  a_k∈A_k
       if   (a_k∈S_k )
       {
          x_k =  a_k;
          if   (k == n)  < Cập nhật kỷ lục>;
          else if   (g(x_1,..., x_k)  ≤ f*) Branch(k+1);
        }
}
```

Cài đặt

```
void  BranchAndBound()
{
     f* = +∞;
     for (v = 1; v<=n;v++) visited[v] = FALSE;
     f=0; x1 = 1; visited[x1] = TRUE;
     Branch(2);
     return f*;
}
```

```
void  BranchAndBound ( )
{
    f* = +∞;
    // Nếu biết p/án x* nào đó thì đặt f* = f(x*)
    Branch(1);
    if  (f  <  +∞)
            <f* là giá trị tối ưu, x* là p/án tối ưu >
   else  <  bài toán không có phương án >;
}
```

Ví dụ 2

Cho 5 thành phố {1, 2, 3, 4, 5}. Giải bài toán người du lịch xuất phát từ thành phố 1 theo phương pháp nhánh cận với ma trận chi phí sau:

0	4	10	2	17
4	0	4	11	9
10	4	0	12	3
2	11	12	0	13
17	9	3	6	0

3. Thuật toán nhánh cận

3.1. Sơ đồ chung

3.2. Ví dụ

3.2.1. Bài toán người du lịch

3.2.2. Bài toán cái túi

3.2.2. Bài toán cái túi (Knap sack)

- Có n loại đồ vật.
- Đồ vật loại j có
  - trọng lượng w_j và
  - giá trị sử dụng là v_j ($j = 1, 2,..., n$) .
- Cần chất các đồ vật này vào một cái túi có trọng lượng là b sao cho tổng giá trị sử dụng của các đồ vật chất trong túi là lớn nhất.

3.2.2. Bài toán cái túi (Knap sack)

- Đưa vào biến số

  x_j – số lượng đồ vật loại j được chất vào túi, j=1,2, ..., n

- Mô hình toán học của bài toán có dạng sau: Tìm

$$f^* = \max\{ f(x) = \sum_{j=1}^{n} v_j x_j : \sum_{j=1}^{n} w_j x_j \le b,\ x_j \in Z_+,\ j = 1,2,...,n \}$$

trong đó Z_+ là tập các số nguyên không âm

Bài toán cái túi biến nguyên

- Kí hiệu D là tập các phương án của bài toán:

$$D = \{ x = (x_1,...,x_n) : \sum_{j=1}^{n} w_j x_j \le b,\ x_j \in Z_+,\ j = 1,2,...,n \}$$

Xây dựng hàm cận trên

- Giả thiết rằng các đồ vật được đánh số sao cho bất đẳng thức sau được thoả mãn:

$$v_1 / w_1 \geq v_2 / w_2 \geq \ldots \geq v_n / w_n .$$

(có nghĩa là các đồ vật được xếp theo thứ tự không tăng của giá trị một đơn vị trọng lượng)

- Để xây dựng hàm tính cận trên, cùng với bài toán cái túi (KP) ta xét bài toán cái túi biến liên tục (KPC) sau đây: Tìm

$$g^* = \max \{ f(x) = \sum_{j=1}^{n} v_j x_j : \sum_{j=1}^{n} w_j x_j \leq b, x_j \geq 0, j = 1, 2, \ldots, n \}$$

Xây dựng hàm cận trên

Mệnh đề. *Phương án tối ưu của bài toán KPC là vectơ* $(x^* = x_1^*, x_2^*, .., x_n^*)$ *với các thành phần được xác định bởi công thức:*

$x_1^* = b/w_1, x_2^* = x_3^* = \cdots . = x_n^* = 0$

và giá trị tối ưu là $g^* = v_1 b / w_1$.

Chứng minh. Xét $x = (x_1, ..., x_n)$ là một phương án tuỳ ý của bài toán KPC. Khi đó

$$v_j \le (v_1 / w_1)\, w_j\,, j = 1, 2, ..., n$$

do $x_j \ge 0$, ta suy ra

$$v_j x_j \le (v_1 / w_1)\, w_j x_j\,, j = 1, 2, ..., n.$$

- Từ đó ta có

$$\sum_{j=1}^{n} v_j x_j \le \sum_{j=1}^{n} (v_1 / w_1) w_j x_j$$

$$= (v_1 / w_1) \sum_{j=1}^{n} w_j x_j$$

$$\le (v_1 / w_1) b = g^*$$

Mệnh đề được chứng minh.

Tính cận trên

- Bây giờ, giả sử ta có phương án bộ phận cấp k: $(u_1, u_2, \ldots, u_k)$. Khi đó giá trị sử dụng của các đồ vật đang có trong túi là

$$\sigma_k = v_1u_1 + v_2u_2 + \ldots + v_ku_k$$

và trọng lượng còn lại của cái túi là

$$b_k = b - (w_1u_1 + w_2u_2 + \ldots + w_ku_k)$$

- Ta có:

$$\max\{f(x) : x \in D, x_j = u_j, j = 1,2,\ldots,k\}$$

$$= \max \{\sigma_k + \sum_{j=k+1}^{n} v_j x_j : \sum_{j=k+1}^{n} w_j x_j \le b_k,\ x_j \in Z_+, j = k+1, k+2, \ldots, n\}$$

$$\le \sigma_k + \max \{\sum_{j=k+1}^{n} v_j x_j : \sum_{j=k+1}^{n} w_j x_j \le b_k,\ x_j \ge 0, j = k+1, k+2, \ldots, n\}$$

$$= \sigma_k + \boxed{v_{k+1} b_k / w_{k+1}}.$$

Với trọng lượng còn lại b_k: lấy toàn bộ là đồ vật (k+1) là đồ vật có giá trị nhất để cho vào túi

- Vậy ta có thể tính cận trên cho phương án bộ phận $(u_1, u_2, \ldots, u_k)$ bởi công thức

$$g(u_1, u_2, \ldots, u_k) = \sigma_k + v_{k+1}\, b_k / w_{k+1}.$$

Tính cận trên

- **Chú ý:** Khi tiếp tục xây dựng thành phần thứ k+1 của lời giải, các ứng cử viên cho x_{k+1} sẽ là 0, 1, ..., $[b_k / w_{k+1}]$.
- Do có kết quả của mệnh đề, khi chọn giá trị cho x_{k+1} ta sẽ duyệt các ứng cử viên theo thứ tự giảm dần.

Ví dụ

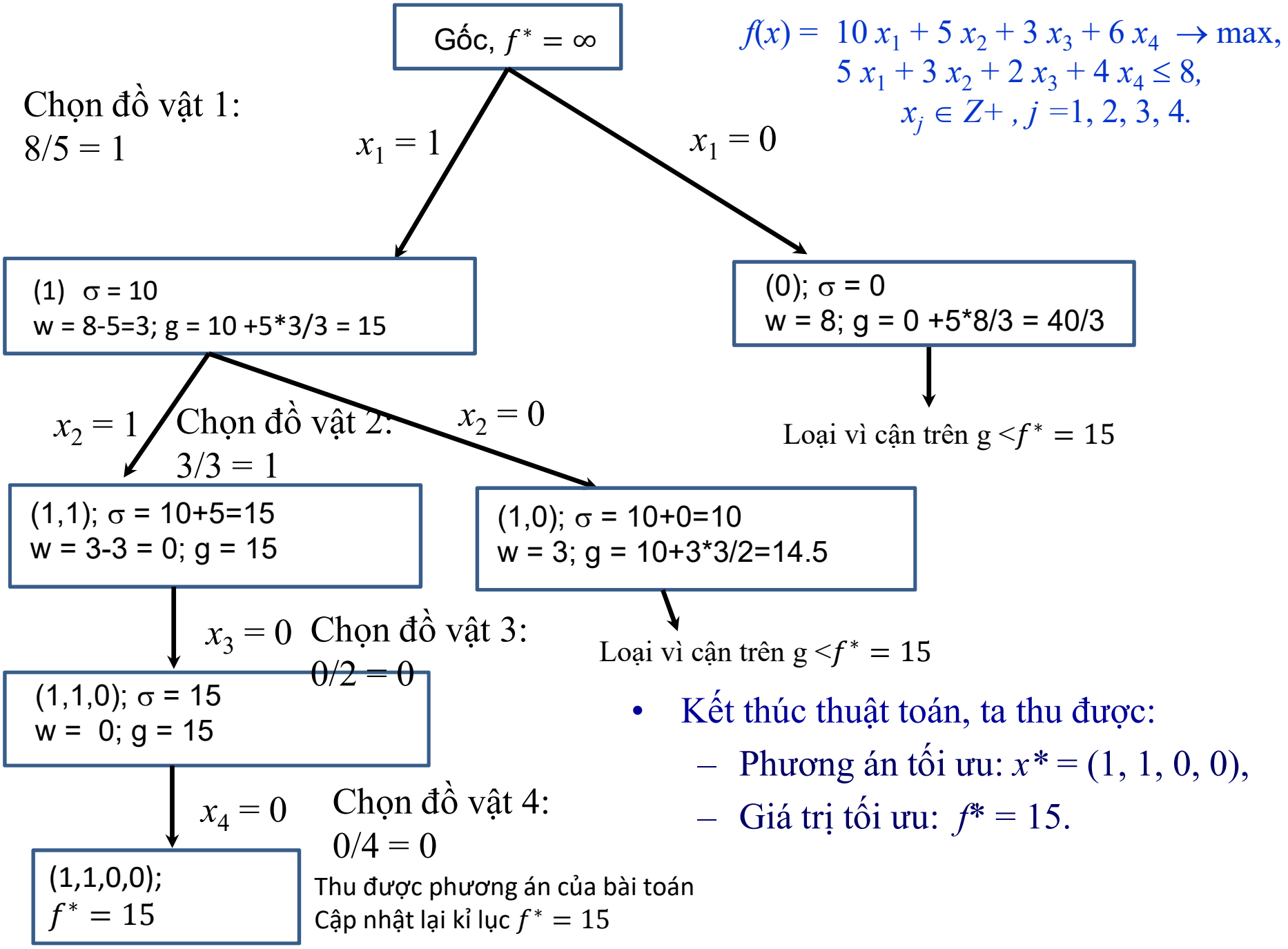
Giải bài toán cái túi sau theo thuật toán nhánh cận vừa trình bày

$$f(x) = 10\,x_1 + 5\,x_2 + 3\,x_3 + 6\,x_4 \rightarrow \max,$$
$$5\,x_1 + 3\,x_2 + 2\,x_3 + 4\,x_4 \le 8,$$
$$x_j \in Z_+ , j = 1, 2, 3, 4.$$

Chú ý: Trong ví dụ đang xét, các đồ vật đã được xếp theo thứ tự không tăng của giá trị một đơn vị trọng lượng.

$$\frac{10}{5} = 2 > \frac{5}{3} \approx 1,66 > \frac{3}{2} = 1,5 = \frac{6}{4}$$

- Quá trình giải bài toán được mô tả trong cây tìm kiếm. Thông tin về một phương án bộ phận trên cây được ghi trong các ô trên hình vẽ tương ứng theo thứ tự sau:
  - các thành phần của phương án bộ phận,
  - σ - giá trị của các đồ vật đang chất trong túi,
  - w – trọng lượng còn lại của túi,
  - g – cận trên của phương án bộ phận.



- Kết thúc thuật toán, ta thu được:
  - Phương án tối ưu: $x^* = (1, 1, 0, 0)$,
  - Giá trị tối ưu: $f^* = 15$.

Ví dụ

Giải bài toán cái túi sau theo thuật toán nhánh cận

$$5x_1 + 8x_2 + x_3 \rightarrow \max$$

$$2x_1 + 3x_2 + x_3 \leq 13$$

$$x_1, x_2, x_3 \geq 0, \text{ nguyên}$$

Chú ý: Trong ví dụ đang xét, các đồ vật chưa được sắp xếp theo thứ tự không tăng của giá trị một đơn vị trọng lượng.

Thứ tự các đồ vật sắp xếp lại là: 2, 1, 3