Chương 4

BÀI TOÁN TỐI ƯU TỔ HỢP

Nguyễn Đức Nghĩa

Nội dung

- 1. Phát biểu bài toán
- 2. Duyệt toàn bộ
- 3. Thuật toán nhánh cận

1. Phát biểu 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 Đức Nghĩa

- 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.

Toán rời rạc

Nguyễn Đức Nghĩa

Phát biểu bài toán

- Díi d¹ng tæng qu¸t bµi to¸n tèi u tæ hîp cã thÓ ph¸t biÓu nh sau:

T×m cùc tiÓu (hay cùc ®¹i) cña phiÕm hµm

$$f(x) \to \text{min (max)},$$

víi ®iÒu kiÖn

$$x \in D,$$

trong ®ã *D* lµ tËp h÷u h¹n phÇn tö.

Các thuật ngữ

- $f(x)$ - hµm môc tiªu cña bµi to¸n,
- $x \in D$ - ph¬ng ¸n
- D - tËp c¸c ph¬ng ¸n cña bµi to¸n.
- Th«ng thêng tËp D ®îc m« t¶ nh lµ tËp c¸c cÊu h×nh tæ hîp tho¶ m·n mét sè tÝnh chÊt cho tríc nµo ®ã.
- Ph¬ng ¸n $x^* \in D$ ®em l¹i gi¸ trÞ nhá nhÊt (lín nhÊt) cho hµm môc tiªu ®îc gäi lµ ***ph¬ng ¸n tèi u***, khi ®ã gi¸ trÞ $f^* = f(x^*)$ ®îc gäi lµ ***gi¸ trÞ tèi u*** cña bµi to¸n.

Nguyễn Đức Nghĩa

1. Phát biểu 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 Đức Nghĩa

Bµi to¸n ngêi du lÞch

(Traveling Salesman Problem – TSP)

- Mét ngêi du lÞch muèn ®i tham quan *n* thµnh phè $T_1, T_2, \ldots, T_n$.
- *Hành trình là cách đi xuÊt ph¸t tõ mét thµnh phè nµo ®ã ®i qua tÊt c¶ c¸c thµnh phè cßn l¹i, mçi thµnh phè ®óng mét lÇn, råi quay trë l¹i thµnh phè xuÊt ph¸t.*
- BiÕt c_{ij} lµ chi phÝ ®i tõ thµnh phè T_i ®Õn thµnh phè T_j ($i, j = 1, 2,\ldots, n$),
- T×m hµnh tr×nh víi tæng chi phÝ lµ nhá nhÊt.

Nguyễn Đức Nghĩa

Nguyễn Đức Nghĩa

Sơ lược về lịch sử

- The origins of the TSP are obscure. In the 1920's, the mathematician and economist Karl Menger publicized it among his colleagues in Vienna.
- In the 1930's, the problem reappeared in the mathematical circles of Princeton.
- In the 1940's, it was studied by statisticians (Mahalanobis (1940), Jessen (1942), Gosh (1948), Marks (1948)) in connection with an agricultural application and the mathematician Merill Flood popularized it among his colleagues at the RAND Corporation. Eventually, the TSP gained notoriety as the prototype of a hard problem in combinatorial optimization: examining the tours one by one is out of the question because of their large number, and no other idea was on the horizon for a long time.
- New history with George Dantzig, Ray Fulkerson, and Selmer Johnson's 1954 breakthrough.

- Ta cã t¬ng øng 1-1 gi÷a một hµnh tr×nh

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

víi mét ho¸n vÞ $\pi = (\pi(1), \pi(2),\ldots, \pi(n))$ cña n sè tù nhiªn 1, 2,..., n.

§Æt

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

Ký hiÖu:

Π - tËp tÊt c¶ c¸c ho¸n vÞ cña n sè tù nhiªn 1, 2,..., n.

Toán rời rạc

Nguyễn Đức Nghĩa

- Khi ®ã bµi to¸n ngêi du lÞch cã thÓ ph¸t biÓu díi d¹ng bµi to¸n tèi u tæ hîp 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-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. Phát biểu 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 Đức Nghĩa

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 a_j và
  - giá trị sử dụng là c_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 Đức Nghĩa

Phát biểu bài toán

- Mét ph¬ng ¸n ®em ®å cña nhµ th¸m hiÓm cã thÓ biÓu diÔn bëi vect¬ nhÞ ph©n ®é dµi *n*: $x = (x_1, x_2, \ldots, x_n)$, trong ®ã $x_j = 1$ nÕu ®å vËt thø *j* ®îc ®em theo vµ $x_j = 0$ nÕu tr¸i l¹i.
- Víi ph¬ng ¸n *x*, gi¸ trÞ ®å vËt ®em theo lµ

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

tæng träng lîng ®å vËt ®em theo lµ

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

Bài toán cái túi

- Bµi to¸n c¸i tói cã thÓ ph¸t biÓu díi d¹ng bµi to¸n tèi u tæ hîp sau:

  Trong sè c¸c vect¬ nhÞ ph©n ®é dµi *n* tho¶ m·n ®iÒu kiÖn $g(x) \leq b$, h·y t×m vect¬ x^* cho gi¸ trÞ lín nhÊt cña hµm môc tiªu $f(x)$:

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

1. Phát biểu 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**

Bµi to¸n ®ãng thïng
(*Bin Packing*)

- Có n đồ vật với trọng lượng là $w_1, w_2, \ldots, w_n$. 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.

Phát biểu bài toán

- Ta cã thÓ gi¶ thiÕt lµ

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

- Do ®ã sè thïng cÇn sö dông ®Ó chøa tÊt c¶ c¸c ®å vËt lµ kh«ng qu¸ *n*. VÊn ®Ò lµ cÇn sè thïng Ýt nhÊt. Ta sÏ më s½n *n* c¸i thïng. Bµi to¸n ®Æt ra lµ h·y x¸c ®Þnh xem mçi mét trong sè *n* ®å vËt cÇn ®îc xÕp vµo c¸i thïng nµo trong sè *n* c¸i thïng ®· më ®Ó cho sè thïng chøa ®å lµ Ýt nhÊt.

Bài toán đóng thùng

- 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:

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

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

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

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

Nguyễn Đức Nghĩa

2. DUYỆT TOÀN BỘ

Nguyễn Đức Nghĩa

NỘI DUNG

2.1. Mô tả phương pháp

2.2. Ví dụ áp dụng: Bài toán cái túi

Nguyễn Đức Nghĩa

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.

Nguyễn Đức Nghĩa

Nguyễn Đức Nghĩa

NỘI DUNG

2.1. Mô tả phương pháp

2.2. Ví dụ áp dụng: Bài toán cái túi

Nguyễn Đức Nghĩa

Ví dụ: Giải bài toán cái túi

- XĐt bµi to¸n c¸i tói:

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

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

- c_j, w_j, b là các số nguyên dương, $j=1,2,\ldots,n$.

- CÇn cã thuËt to¸n liÖt kª c¸c phÇn tö cña *D*

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 $(x_1, \ldots, x_{k-1})$. 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

  $$f_{k-1}= c_1x_1 + \ldots + c_{k-1}x_{k-1}$$

  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 to t_k do

Nguyễn Đức Nghĩa

Chương trình trên Pascal

```
type
arrint= array[1..20] of integer;
var
    x, xopt, c, w: arrint;
    n,b, bk, fk, fopt: integer;

procedure Nhapdl;
var i: integer;
begin
    {Nhập vào n, c, w, b}
end;
```

```
procedure Inkq;
var j;
begin
    {Phương án tối ưu: xopt;
     Giá trị tối ưu: fopt }
end;
```

Nguyễn Đức Nghĩa

```
procedure KP(i: integer);
var j, t: integer;
begin
   if bk>=w[i] then t:=1  else t:=0;
   for j := t downto 0 do begin
      x[i] := j; bk:= bk-w[i]*x[i];
      fk:= fk + c[i]*x[i];
      if  i = n then  begin
         if fk>fopt then begin
               xopt:=x; fopt:=fk;
         end
      end else KP(i+1);
      bk:= bk+w[i]*x[i];
      fk:= fk - c[i]*x[i];
   end;
end;
```

```
BEGIN {Main program}
   Nhapdl;
   bk:=b;
   fk:= 0;
   fopt:= 0;
   KP(1);
   InKq
END.
```

Nguyễn Đức Nghĩa

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 Đức Nghĩa

- V× vËy cÇn ph¶i cã nh÷ng biÖn ph¸p nh»m h¹n chÕ viÖc t×m kiÕm th× míi cã hy väng gi¶i ®îc c¸c bµi to¸n tèi u tæ hîp thùc tÕ. TÊt nhiªn ®Ó cã thÓ ®Ò ra nh÷ng biÖn ph¸p nh vËy cÇn ph¶i nghiªn cøu kü tÝnh chÊt cña bµi to¸n tèi u tæ hîp cô thÓ.
- Nhê nh÷ng nghiªn cøu nh vËy, trong mét sè trêng hîp cô thÓ ta cã thÓ x©y dùng nh÷ng thuËt to¸n hiÖu qu¶ ®Ó gi¶i bµi to¸n ®Æt ra.

- Tuy nhiªn ph¶i nhÊn m¹nh r»ng trong nhiÒu trêng hîp (vÝ dô trong c¸c bµi to¸n ngêi du lÞch, bµi to¸n c¸i tói, bµi to¸n ®ãng thïng) chóng ta cha thÓ x©y dùng ®îc ph¬ng ph¸p h÷u hiÖu nµo kh¸c ngoµi ph¬ng ph¸p duyÖt toµn bé.

Nguyễn Đức Nghĩa

- Khi ®ã, mét vÊn ®Ò ®Æt ra lµ trong qu¸ tr×nh liÖt kª lêi gi¶i ta cÇn tËn dông c¸c th«ng tin ®· t×m ®îc ®Ó lo¹i bá nh÷ng ph¬ng ¸n ch¾c ch¾n kh«ng ph¶i lµ tèi u.
- Trong môc tiÕp theo chóng ta sÏ xÐt mét s¬ ®å t×m kiÕm nh vËy ®Ó gi¶i c¸c bµi to¸n tèi u tæ hîp mµ trong tµi liÖu tham kh¶o ®îc biÕt ®Õn víi tªn gäi: thuËt to¸n nh¸nh cËn.

Nguyễn Đức Nghĩa

3. THUẬT TOÁN NHÁNH CẬN

(Branch and Bound Algorithm)

NỘI DUNG

- **3.1. Sơ đồ chung**
- 3.2. Bài toán cái túi
- 3.3. Bài toán người du lịch

Nguyễn Đức Nghĩa

Nguyễn Đức Nghĩa

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.*

Toán rời rạc

Sơ đồ chung

- Ta sÏ m« t¶ tëng cña thuËt to¸n trªn m« h×nh bµi to¸n tèi u tæ hîp tæng qu¸t sau

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

trong ®ã D lµ tËp h÷u h¹n phÇn tö.

Gi¶ thiÕt r»ng tËp D ®îc m« t¶ nh sau

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

$$x \text{ tho¶ m·n tÝnh chÊt } P\},$$

víi $A_1, A_2, ..., A_n$ lµ c¸c tËp h÷u h¹n, cßn P lµ tÝnh chÊt cho trªn tÝch §Òcac $A_1 \times A_2 \times \dots \times A_n$.

Nguyễn Đức Nghĩa

Nhận xét

- Nhận thấy rằng, các bài toán vừa trình bày ở mục 1 đều có thể mô tả dưới dạng bài toán trên.
- 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

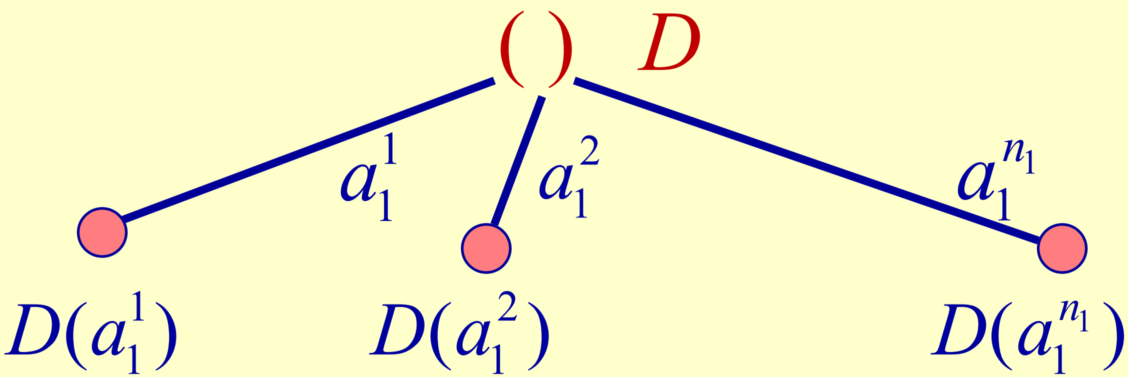
$$\min \{g(x): x \in D\}, \text{ trong đó } g(x) = -f(x)$$

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

Nguyễn Đức Nghĩa

Ph©n nh¸nh

- Qu¸ tr×nh ph©n nh¸nh ®îc thùc hiÖn nhê thuËt to¸n quay lui:



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

lµ tËp c¸c ph¬ng ¸n cã thÓ ph¸t triÓn tõ pabp (a_1^i)

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

Toán rời rạc

Nguyễn Đức Nghĩa

Nguyễn Đức Nghĩa

Ph©n nh¸nh

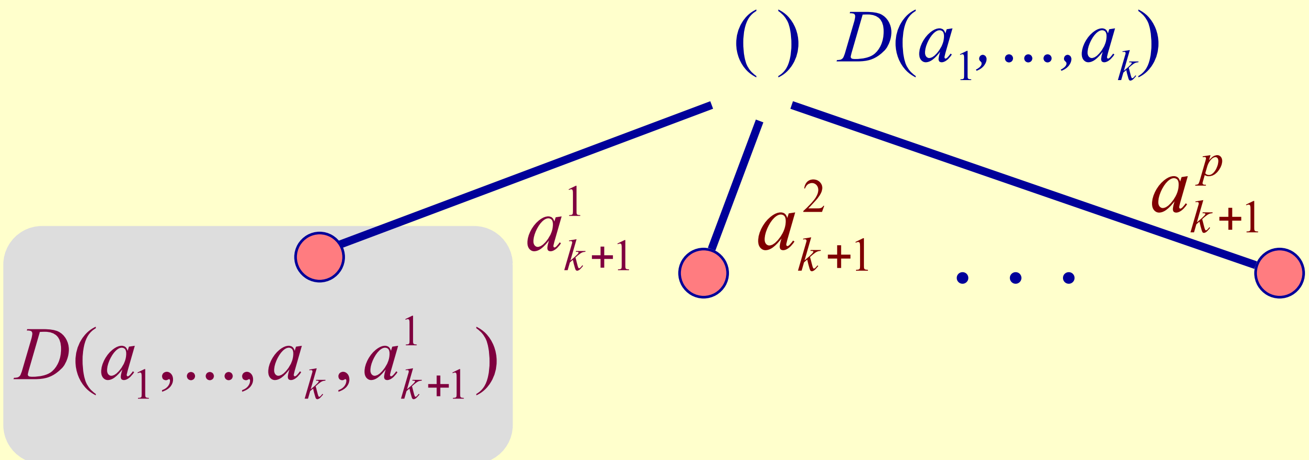
- Như vậy ta có thể đặt tương ứng mỗi phương án bộ phận $(a_1, a_2, ..., a_k)$ với một tập con các phương án của bài toán:

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

- 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.

Ph©n nh¸nh

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



- Ta cã ph©n ho¹ch:

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

Nguyễn Đức Nghĩa

Tính cận

- Cần có hµm *g* x¸c ®Þnh trªn tËp tÊt c¶ c¸c ph¬ng ¸n bé phËn cña bµi to¸n tho¶ m·n bÊt ®¼ng thøc sau:

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

  víi mỗi lêi gi¶i bé phËn $(a_1, a_2, \ldots, a_k)$, vµ víi mäi $k = 1, 2, \ldots$

Toán rời rạc

Nguyễn Đức Nghĩa

- BÊt ®¼ng thøc (*) cã nghÜa lµ gi¸ trÞ cña hµm *g* t¹i ph¬ng ¸n bé phËn (a_1, a_2, ..., a_k) lµ kh«ng vît qu¸ gi¸ trÞ nhá nhÊt cña hµm môc tiªu cña bµi to¸n trªn tËp con c¸c ph¬ng ¸n

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

hay nãi mét c¸ch kh¸c, $g(a_1, a_2, . . . , a_k)$ lµ **cËn díi** cña gi¸ trÞ hµm môc tiªu trªn tËp $D(a_1, a_2, \ldots, a_k)$.

- V× lÏ ®ã, hµm g ®îc gäi lµ hµm cËn díi, vµ gi¸ trÞ $g(a_1, a_2, . . . , a_k)$ ®îc gäi lµ cËn díi cña tËp $D(a_1, a_2, ..., a_k)$.
- Do cã thÓ ®ång nhÊt tËp $D(a_1,..., a_k)$ víi ph¬ng ¸n bé phËn $(a_1,..., a_k)$, nªn ta còng gäi gi¸ trÞ $g(a_1,..., a_k)$ lµ cËn díi cña ph¬ng ¸n bé phËn $(a_1,..., a_k)$.

Cắt nhánh nhờ sử dụng cận dưới

- Gi¶ sö ®· cã hµm *g*. Ta xÐt c¸ch sö dông hµm nµy ®Ó gi¶m bít khèi lîng duyÖt trong qu¸ tr×nh duyÖt tÊt c¶ c¸c ph¬ng ¸n theo thuËt to¸n quay lui.
- Trong qu¸ tr×nh liÖt kª c¸c ph¬ng ¸n cã thÓ ®· thu ®îc mét sè ph¬ng ¸n cña bµi to¸n. Gäi ¯*x* lµ ph¬ng ¸n víi gi¸ trÞ hµm môc tiªu nhá nhÊt trong sè c¸c ph¬ng ¸n ®· t×m ®îc, ký hiÖu ¯*f* = *f*(¯*x*).
- Ta sÏ gäi
  - ¯*x* lµ ph¬ng ¸n tèt nhÊt hiÖn cã,
  - cßn ¯*f* lµ kû lôc.

Toán rời rạc

Nguyễn Đức Nghĩa

Cắt nhánh nhờ sử dụng cận dưới

- Gi¶ sö ®· cã $\bar{f}$, khi ®ã nÕu

$$g(a_1, a_2, \ldots, a_k) > \bar{f},$$

th× tõ bÊt ®¼ng thøc (*) suy ra

$$\bar{f} < g(a_1,\ldots, a_k) \leq \min\{f(x): x \in D(a_1,\ldots,a_k)\},$$

- V× thÕ tËp $D(a_1,\ldots, a_k)$ ch¾c ch¾n kh«ng chøa ph¬ng ¸n tèi u và có thể loại bỏ khỏi quá trình duyệt.

Thuật toán nhánh cận

procedure Branch(k);

(* Phát triển phương án bộ phận (x_1, x_2, ..., x_{k-1}) *)

begin

    for $a_k \in A_k$ **do**

      if $a_k \in S_k$ **then**

      begin

          $x_k := a_k$;

          if $(k = n)$ **then** < Cập nhật kỷ lục>

          else

          if $g(x_1, \ldots, x_k) \le \overline{f}$ **then** Branch(k+1)

      end;

end;

Thuật toán nhánh cận

procedure BranchAndBound;

begin

$\bar{f}$:= +∞;

(* Nếu biết p/án $\bar{x}$ nào đó thì đặt $\bar{f} = f(\bar{x})$ *)

Branch(1);

if $\bar{f}$ < +∞ then

< $\bar{f}$ là giá trị tối ưu, $\bar{x}$ là p/án tối ưu >

else < bài toán không có phương án >**;**

end;

Chú ý: Sơ đồ duyệt toàn bộ

- Chó ý r»ng nÕu trong thñ tôc Branch ta thay c©u lÖnh

  if (k = n) **then** < CËp nhËt kû lôc>

  else

  if $g(a_1,..., a_k)$ $\leq$ ¯f **then** Branch(k+1)

  bëi

  if (k = n) **then** < CËp nhËt kû lôc>

  else Branch(k+1)

  th× ta thu ®îc thuËt to¸n **duyÖt toµn bé.**

Nguyễn Đức Nghĩa

Chú ý:

- 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.

NỘI DUNG

- 3.1. Sơ đồ chung
- **3.2. Bài toán cái túi**
- 3.3. Bài toán người du lịch

Nguyễn Đức Nghĩa

Nguyễn Đức Nghĩa

Bài toán cái túi

- Có n loại đồ vật.
- Đồ vật loại j có
  - trọng lượng a_j và
  - giá trị sử dụng là c_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.

Bài toán cái túi (KP)

- 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

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

trong ®ã Z_+ lµ tËp c¸c sè nguyªn kh«ng ©m.

- Ký hiÖu *D* lµ tËp c¸c ph¬ng ¸n cña bµi to¸n:

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

- Gi¶ thiÕt r»ng c¸c ®å vËt ®îc ®¸nh sè sao cho bÊt ®¼ng thøc sau ®îc tho¶ m·n

$$c_1 / a_1 \ge c_2 / a_2 \ge ... \ge c_n / a_n .$$

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

Xây dựng hàm cận trên

- §Ó x©y dùng hµm tÝnh cËn trên, cïng víi bµi to¸n c¸i tói (KP) ta xÐt bµi to¸n c¸i tói biÕn liªn tôc (KPC) sau đây: T×m

$$g^* = \max\left\{ f(x) = \sum_{j=1}^{n} c_j x_j : \sum_{j=1}^{n} a_j x_j \le b,\ x_j \ge 0,\ j = 1, 2, \ldots, n \right\}$$

- **MÖnh ®Ò.** *Ph¬ng ¸n tèi u cña bµi to¸n KPC lµ vect¬ $\bar{x} = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$ víi c¸c thµnh phÇn ®îc x¸c ®Þnh bëi c«ng thøc:*

$$\bar{x}_1 = b / a_1,\ \bar{x}_2 = \bar{x}_3 = \ldots = \bar{x}_n = 0.$$

vµ gi¸ trÞ tèi u lµ $g^* = c_1 b / a_1$.

- **Chøng minh.** XĐt $x = (x_1,\ldots, x_n)$ lµ mét ph¬ng ¸n tuú ý cña bµi to¸n KPC. Khi ®ã

$$c_j \leq (c_1 / a_1)\, a_j,\ j = 1, 2, \ldots, n$$

do $x_j \geq 0$, ta suy ra

$$c_j x_j \leq (c_1 / a_1)\, a_j x_j,\ j = 1, 2, \ldots, n$$

- Từ đó ta có

$$\sum_{j=1}^{n} c_j x_j \leq \sum_{j=1}^{n} (c_1/a_1) a_j x_j$$

$$= (c_1/a_1) \sum_{j=1}^{n} a_j x_j$$

$$\leq (c_1/a_1) b = g^*$$

- Mệnh đề được chứng minh.

Toán rời rạc

Nguyễn Đức Nghĩa

- B©y giê, gi¶ sö ta cã ph¬ng ¸n bé phËn cÊp *k*: (u_1, u_2, ..., u_k).
- Khi ®ã gi¸ trÞ sö dông cña c¸c ®å vËt ®ang cã trong tói lµ

$$\sigma_k = c_1u_1 + c_2u_2 + . . . + c_ku_k$$

vµ träng l¬îng cßn l¹i cña c¸i tói lµ

$$b_k = b – (a_1u_1 + a_2u_2 + . . . + a_ku_k).$$

Toán rời rạc

Nguyễn Đức Nghĩa

Tính cận trên

- Ta cã

$$
\begin{aligned}
&\max\{f(x): x\in D, x_j = u_j, j=1,2,\dots,k\}\\
&= \max \ \{\sigma_k + \sum_{j=k+1}^{n} c_j x_j : \sum_{j=k+1}^{n} a_j x_j \le b_k,\ x_j \in Z_+, j=k+1, k+2,\dots,n\}\\
&\le \sigma_k + \max \ \{\sum_{j=k+1}^{n} c_j x_j : \sum_{j=k+1}^{n} a_j x_j \le b_k,\ x_j \ge 0, j=k+1, k+2,\dots,n\}\\
&= \sigma_k + c_{k+1} b_k / a_{k+1}.
\end{aligned}
$$

- VËy ta cã thÓ tÝnh cËn trªn cho ph¬ng ¸n bé phËn (u_1, u_2, ..., u_k) bëi c«ng thøc

$$g(u_1, u_2, \dots, u_k) = \sigma_k + c_{k+1}\, b_k / a_{k+1}.$$

- **Chó ý:** Khi tiÕp tôc x©y dùng thµnh phÇn thø $k+1$ cña lêi gi¶i, c¸c ứng cö viên cho x_{k+1} sÏ lµ 0, 1, ..., $[b_k / a_{k+1}]$.
- Do cã kÕt qu¶ cña mÖnh ®Ò, khi chän gi¸ trÞ cho x_{k+1} ta sÏ duyÖt c¸c ứng cö viên theo thø tù gi¶m dÇn.

Nguyễn Đức Nghĩa

Ví dụ

- Gi¶i bµi to¸n c¸i tói sau theo thuËt to¸n nh¸nh cËn võa tr×nh bµy

$$f(x) = 10x_1 + 5x_2 + 3x_3 + 6x_4 \to \max,$$

$$5x_1 + 3x_2 + 2x_3 + 4x_4 \le 8,$$

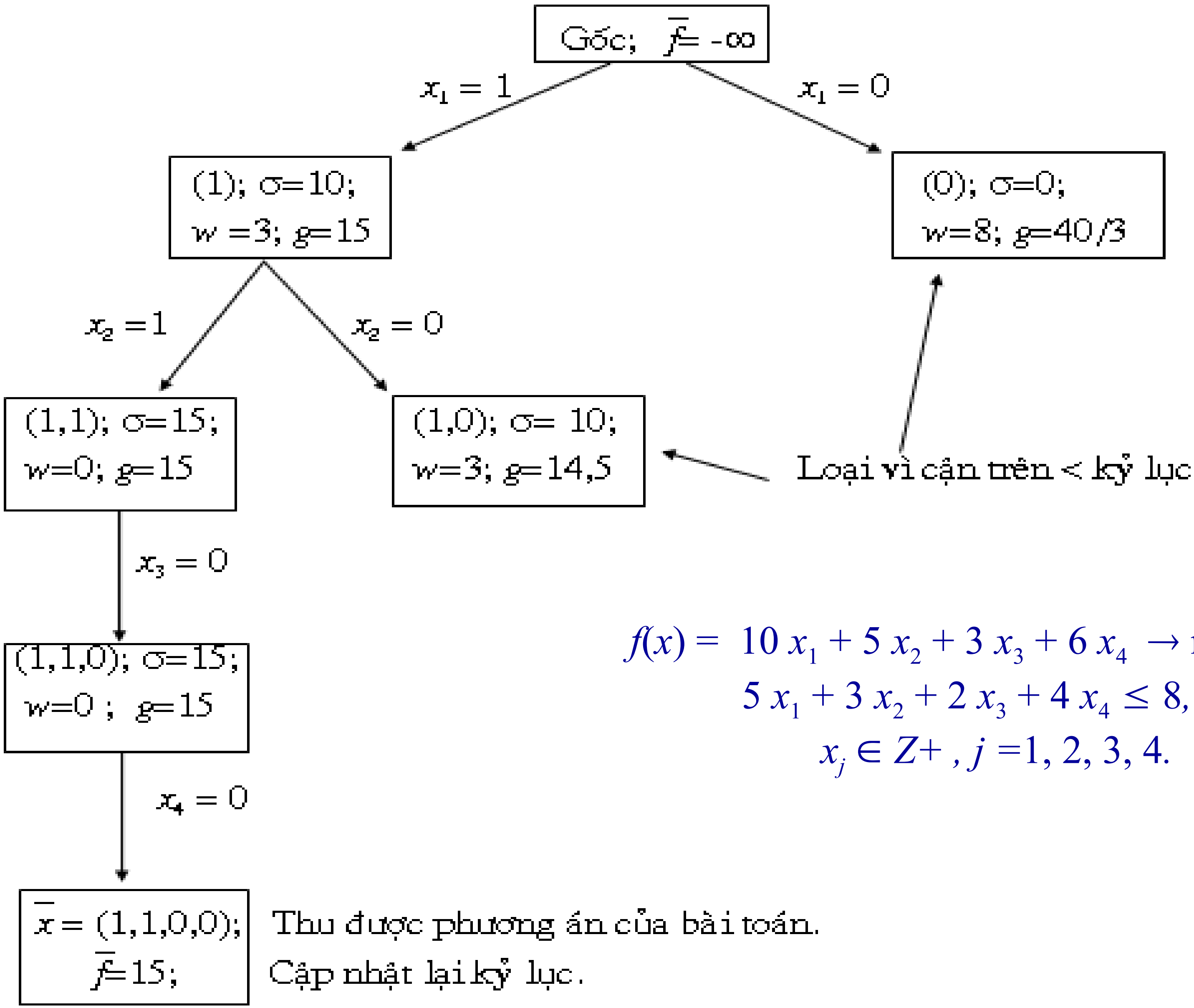
$$x_j \in Z_+,\ j = 1, 2, 3, 4.$$

- *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.*

- Qu¸ tr×nh gi¶i bµi to¸n ®îc m« t¶ trong c©y t×m kiÕm trong h×nh 1. Th«ng tin vÒ mét ph¬ng ¸n bé phËn trªn c©y ®îc ghi trong c¸c « trªn h×nh vÏ t¬ng øng theo thø tù sau:
  - c¸c thµnh phÇn cña ph¬ng ¸n,
  - σ - gi¸ trÞ cña c¸c ®å vËt ®ang chÊt trong tói,
  - *w* - träng lîng cßn l¹i cña tói
  - *g* - cËn trªn.

Nguyễn Đức Nghĩa

Nguyễn Đức Nghĩa



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

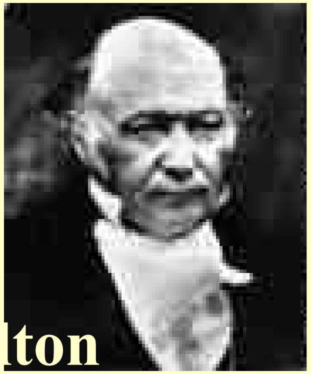
- KÕt thóc thuËt to¸n, ta thu ®îc:
  - Ph¬ng ¸n tèi u: x^* = (1, 1, 0, 0),
  - Gi¸ trÞ tèi u: f^* = 15.

Toán rời rạc

Nguyễn Đức Nghĩa

NỘI DUNG

- 3.1. Sơ đồ chung
- 3.2. Bài toán cái túi
- **3.3. Bài toán người du lịch**



- Cè ®Þnh thµnh phè xuÊt ph¸t lµ T_1, bµi to¸n ngêi du lÞch dÉn vÒ bµi to¸n:

  T×m cùc tiÓu cña hµm

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

víi ®iÒu kiÖn

  $(1, x_2, x_3, \dots, x_n)$ lµ ho¸n vÞ cña c¸c sè 1,2, ..., n.

Nguyễn Đức Nghĩa

Toán rời rạc

Hàm cận dưới

- Ký hiÖu

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

lµ chi phÝ ®i l¹i nhá nhÊt gi÷a c¸c thµnh phè.

- CÇn ®¸nh gi¸ cËn díi cho ph¬ng ¸n bé phËn (1, u_2, ..., u_k) t¬ng øng víi hµnh tr×nh bé phËn qua k thµnh phè:

$$T_1 \rightarrow T(u_2) \rightarrow \ldots \rightarrow T(u_{k-1}) \rightarrow T(u_k).$$

Nguyễn Đức Nghĩa

Hàm cận dưới

- Chi phÝ ph¶i tr¶ theo hµnh tr×nh bé phËn nµy lµ

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

- §Ó ph¸t triÓn thµnh hµnh tr×nh ®Çy ®ñ, ta cßn ph¶i ®i qua n-k+1 ®o¹n ®êng n÷a, mçi ®o¹n cã chi phÝ kh«ng Ýt h¬n c_{min}, nªn cËn díi cho ph¬ng ¸n bé phËn (1, u_2, ..., u_k) cã thÓ tÝnh theo c«ng thøc

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

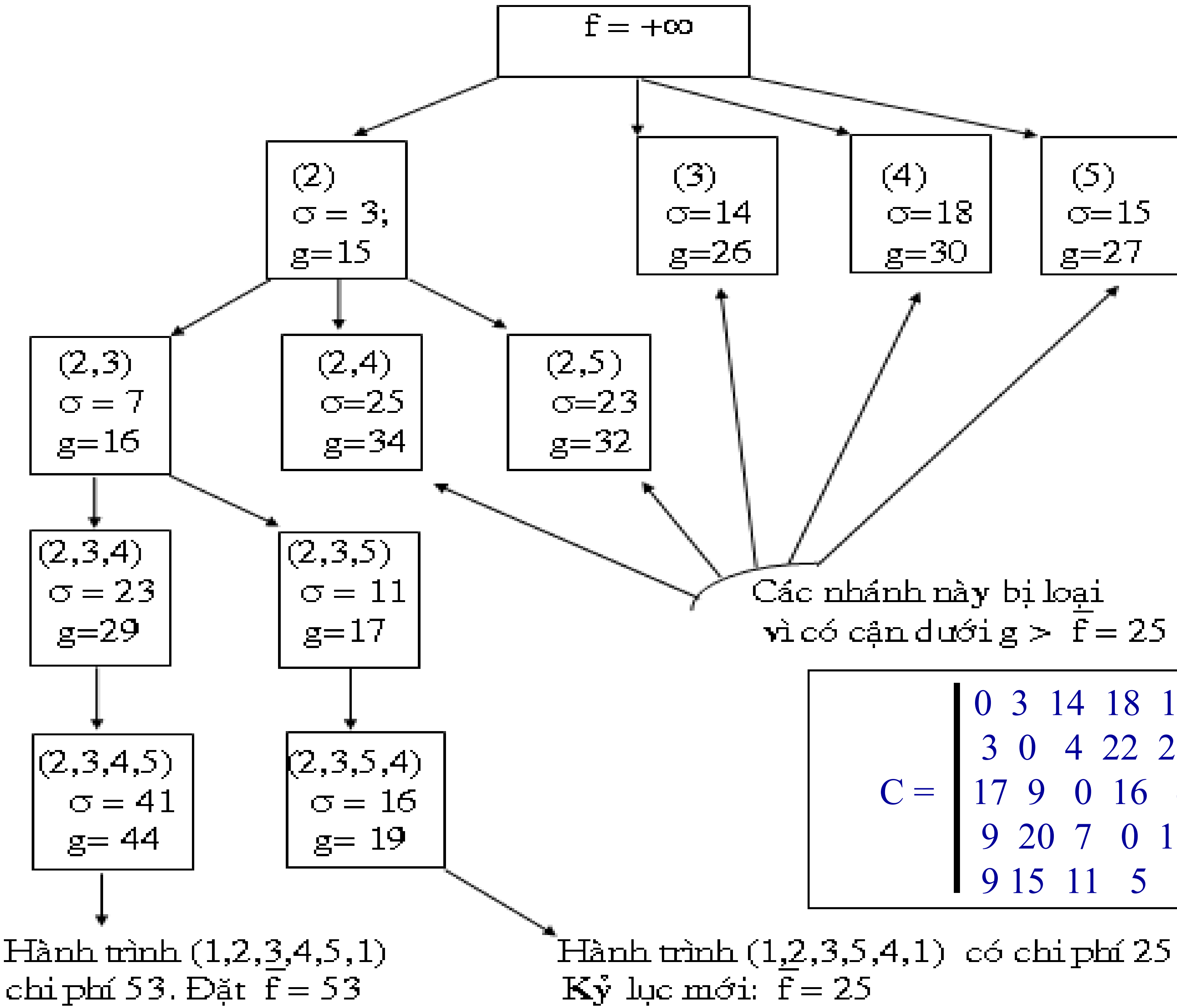
Ví dụ

- Giải bài toán người du lịch với ma trận chi phí sau:

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

Nguyễn Đức Nghĩa

- Ta cã c_{min} = 3. Qu¸ tr×nh thùc hiÖn thuËt to¸n ®îc m« t¶ bëi c©y t×m kiÕm lêi gi¶i.
- Th«ng tin ®îc ghi trong c¸c « trªn h×nh vÏ theo thø tù sau:
  - c¸c thµnh phÇn cña ph¬ng ¸n,
  - σ lµ chi phÝ theo hµnh tr×nh bé phËn
  - *g* - cËn díi.



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

Kết quả

- Kết thúc thuật toán, ta thu được phương án tối ưu (1, 2, 3, 5, 4, 1) tương ứng với hành trình

$$T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_5 \rightarrow T_4 \rightarrow T_1 ,$$

- Chi phí nhỏ nhất là 25.

Nguyễn Đức Nghĩa

Nguyễn Đức Nghĩa

Kỷ lục
về giải bài toán người du lịch

Kỷ lục
(Kích thước TSP giải được)

1954	1962	1977	1987	1987	1987	1994	1998	2001	2004
49	33	120	532	666	2392	7397	13509	15112	24978

http://www.tsp.gatech.edu/index.html

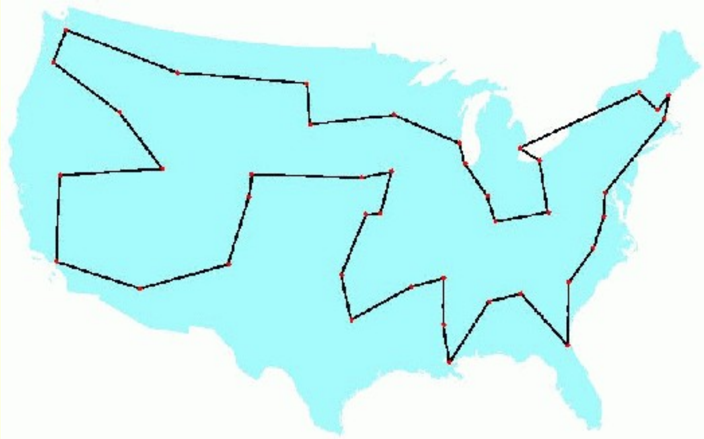
Nguyễn Đức Nghĩa

Year	Research Team	Size of Instance
1954	G. Dantzig, R. Fulkerson, and S. Johnson	49 cities
1971	M. Held and R.M. Karp	64 cities
1975	P.M. Camerini, L. Fratta, and F. Maffioli	67 cities
1977	M. Grötschel	120 cities
1980	H. Crowder and M.W. Padberg	318 cities
1987	M. Padberg and G. Rinaldi	532 cities
1987	M. Grötschel and O. Holland	666 cities
1987	M. Padberg and G. Rinaldi	2,392 cities
1994	D. Applegate, R. Bixby, V. Chvátal, and W. Cook	7,397 cities
1998	D. Applegate, R. Bixby, V. Chvátal, and W. Cook	13,509 cities
2001	D. Applegate, R. Bixby, V. Chvátal, and W. Cook	15,112 cities
2004	D. Applegate, R. Bixby, V. Chvátal, W. Cook, and K. Helsgaun	24,978 cities

Nguyễn Đức Nghĩa

The First Big TSP

Dantzig, Ray Fulkerson, and Selmer Johnson (1954) published a description of a method for solving the TSP and illustrated the power of this method by solving an instance with 49 cities, an impressive size at that time. They created this instance by picking one city from each of the 48 states in the U.S.A. (Alaska and Hawaii became states only in 1959) and adding Washington, D.C.; the costs of travel between these cities were defined by road distances. Rather than solving this problem, they solved the 42-city problem obtained by removing Baltimore, Wilmington, Philadelphia, Newark, New York, Hartford, and Providence. As it turned out, an optimal tour through the 42 cities used the edge joining Washington, D.C. to Boston; since the shortest route between these two cities passes through the seven removed cities, this solution of the 42-city problem yields a solution of the 49-city problem.



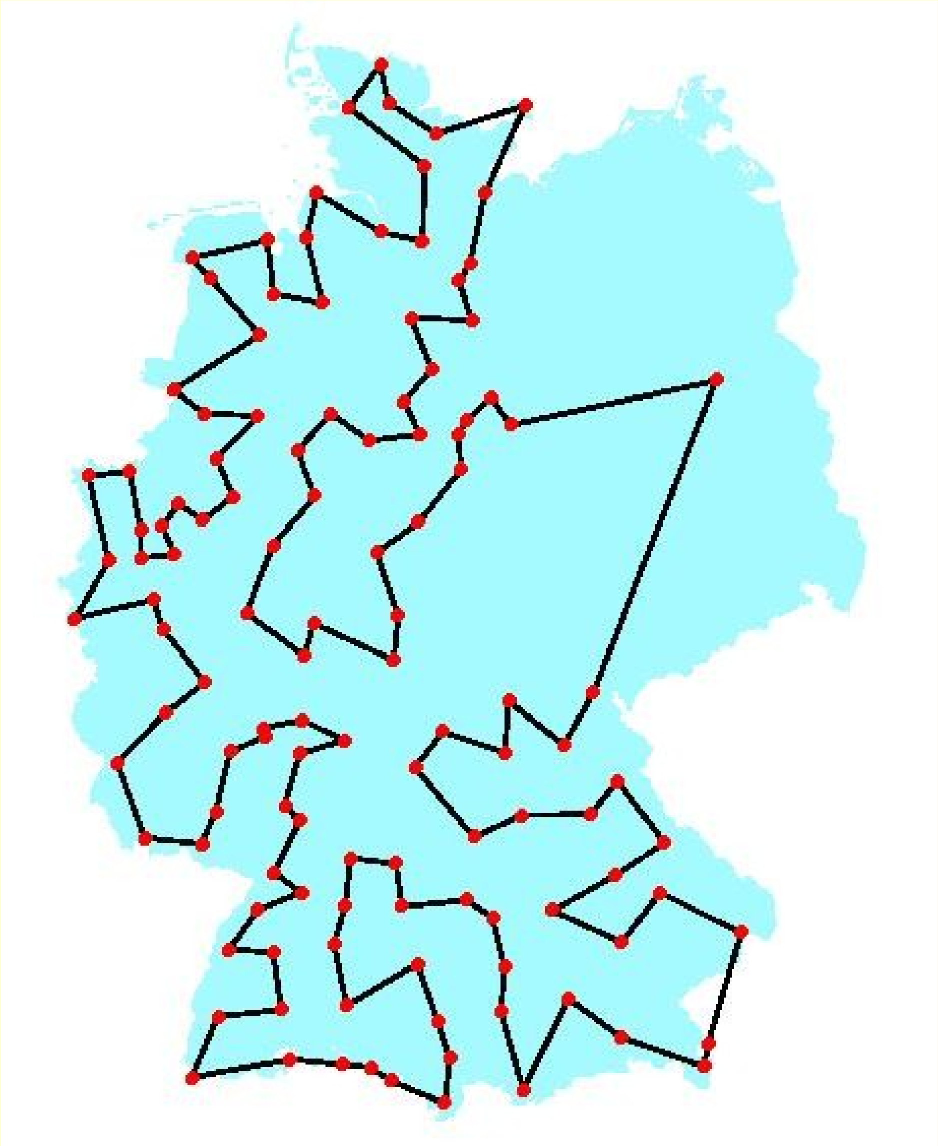
Nguyễn Đức Nghĩa

Procter and Gamble's Contest

- Proctor and Gamble ran a contest in 1962. The contest required solving a TSP on a specified 33 cities. There was a tie between many people who found the optimum. An early TSP researcher, Professor Gerald Thompson of Carnegie Mellon University, was one of the winners.

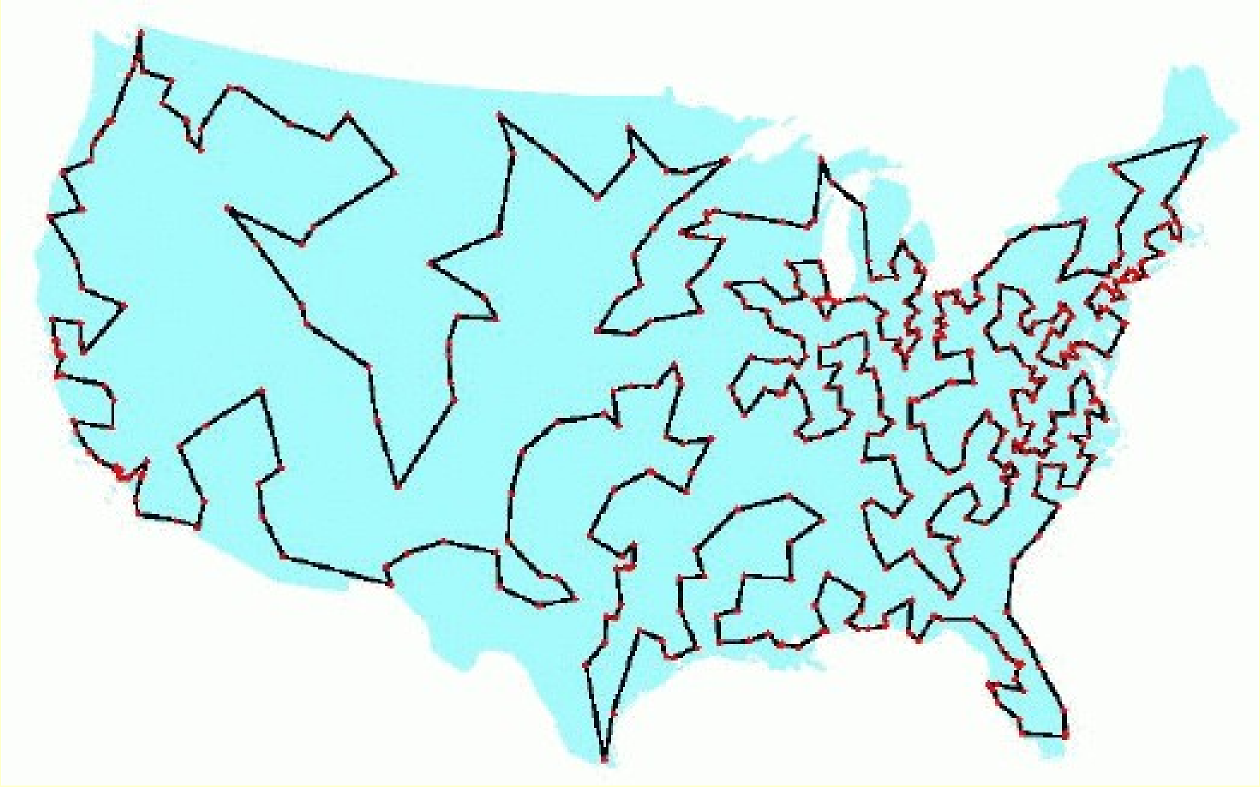
120 Western German Cities

- Groetschel (1977) found the optimal tour of 120 cities from what was then West Germany.



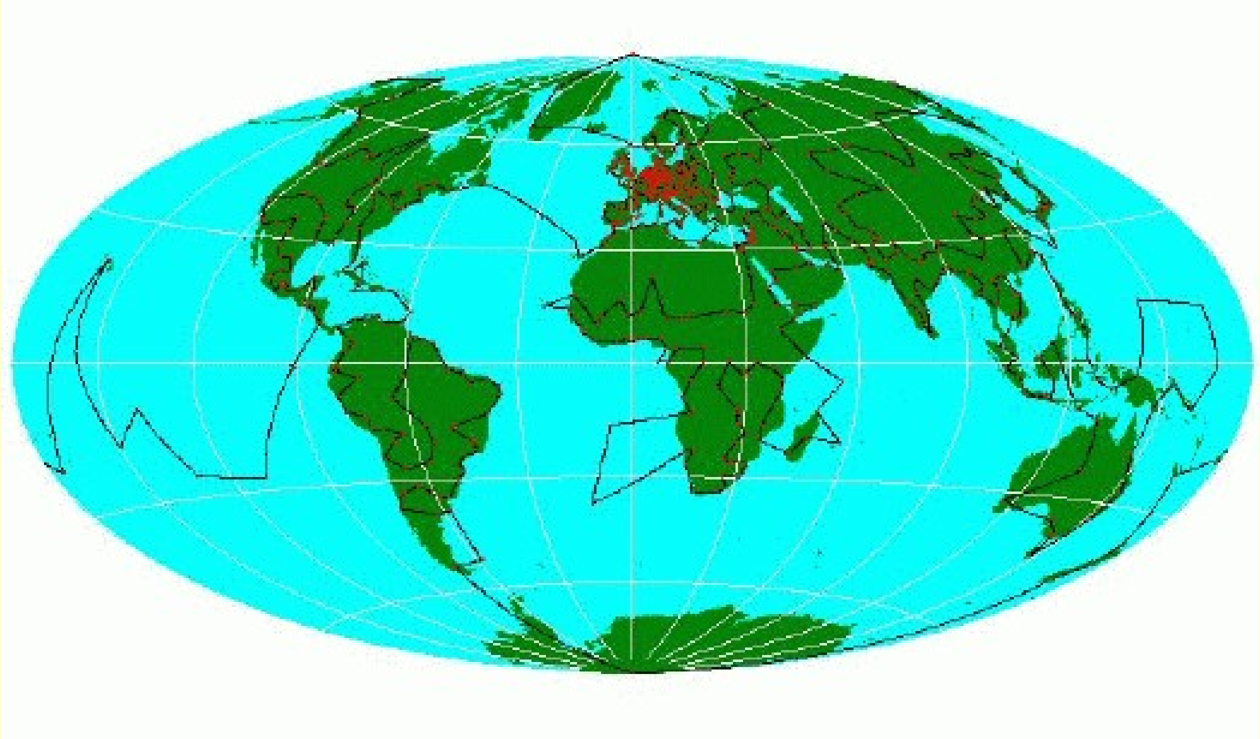
532 Locations in America

- Padberg and Rinaldi (1987) found the optimal tour of 532 AT&T switch locations in the USA.



666 Cities Worldwide

- Groetschel and Holland (1987) found the optimal tour of 666 interesting places in the world.

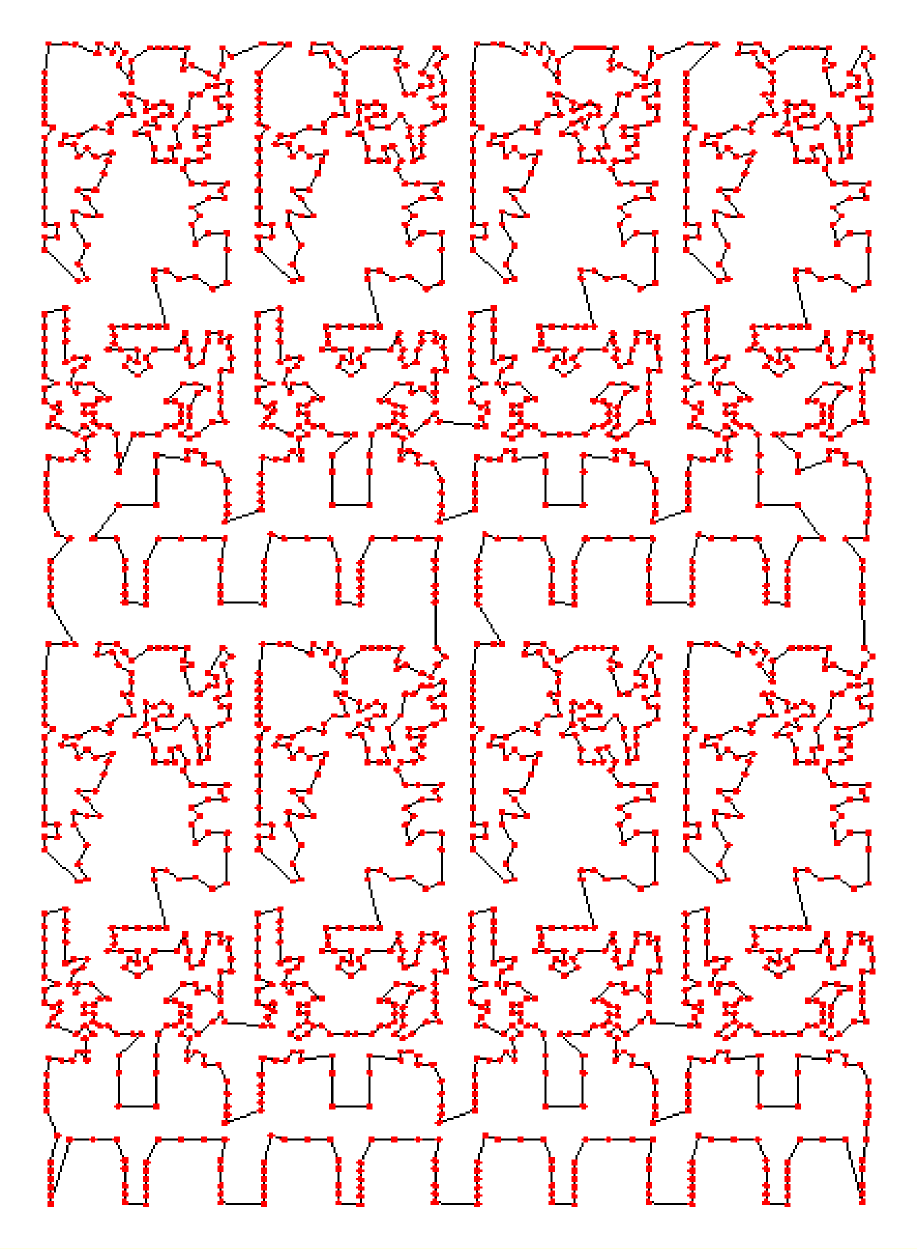


Toán rời rạc

Nguyễn Đức Nghĩa

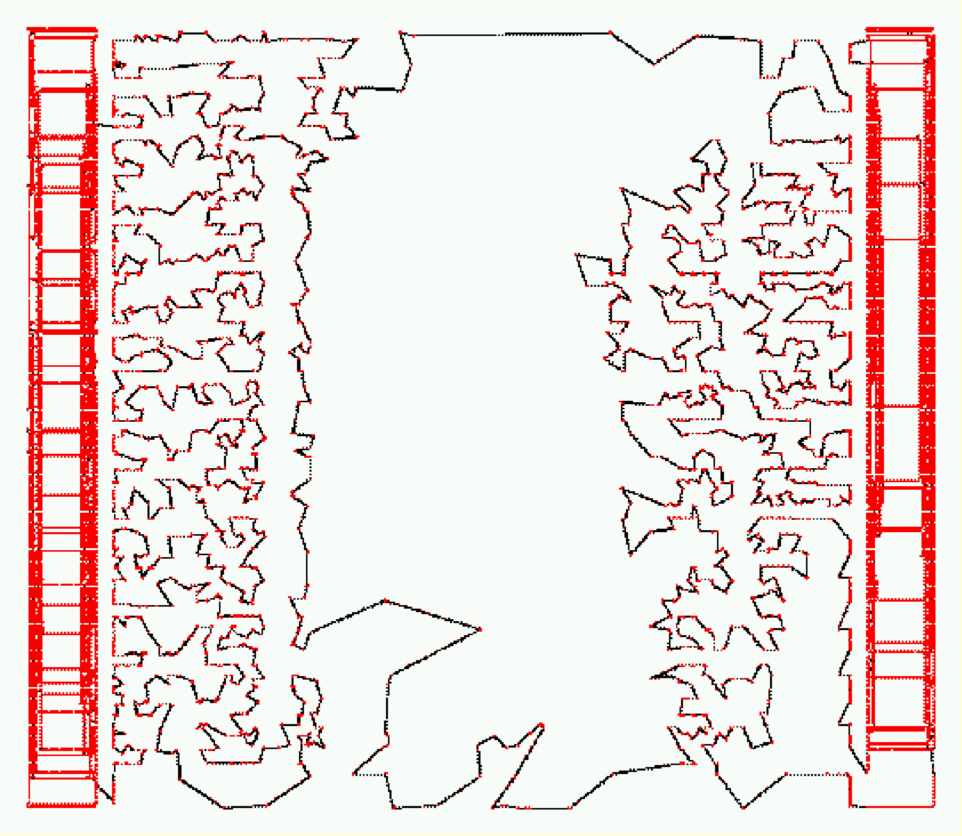
2,392 Points

- Padberg and Rinaldi (1987) found the optimal tour through a layout of 2,392 points obtained from Tektronics Incorporated.



7,397-city TSP

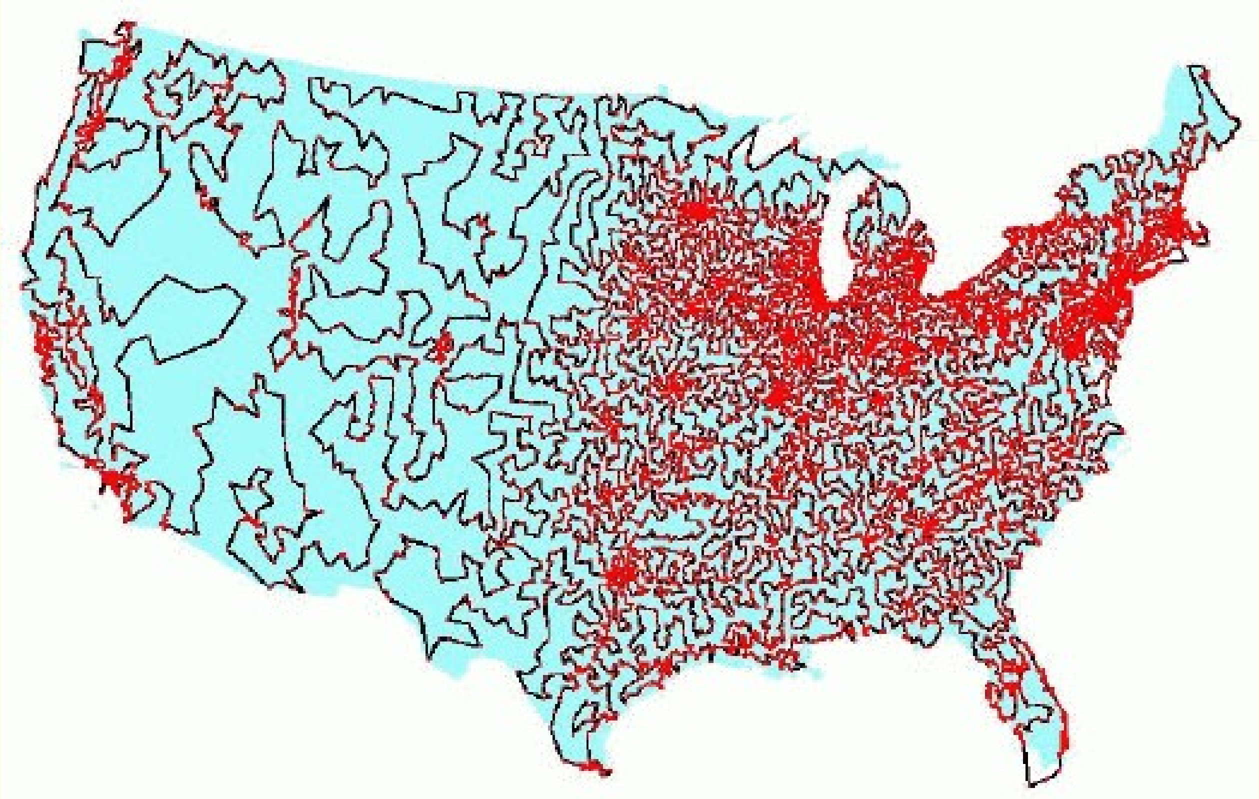
- Applegate, Bixby, Chvátal, and Cook (1994) found the optimal tour for a 7,397-city TSP that arose in a programmable logic array application at AT&T Bell Laboratories.



Nguyễn Đức Nghĩa

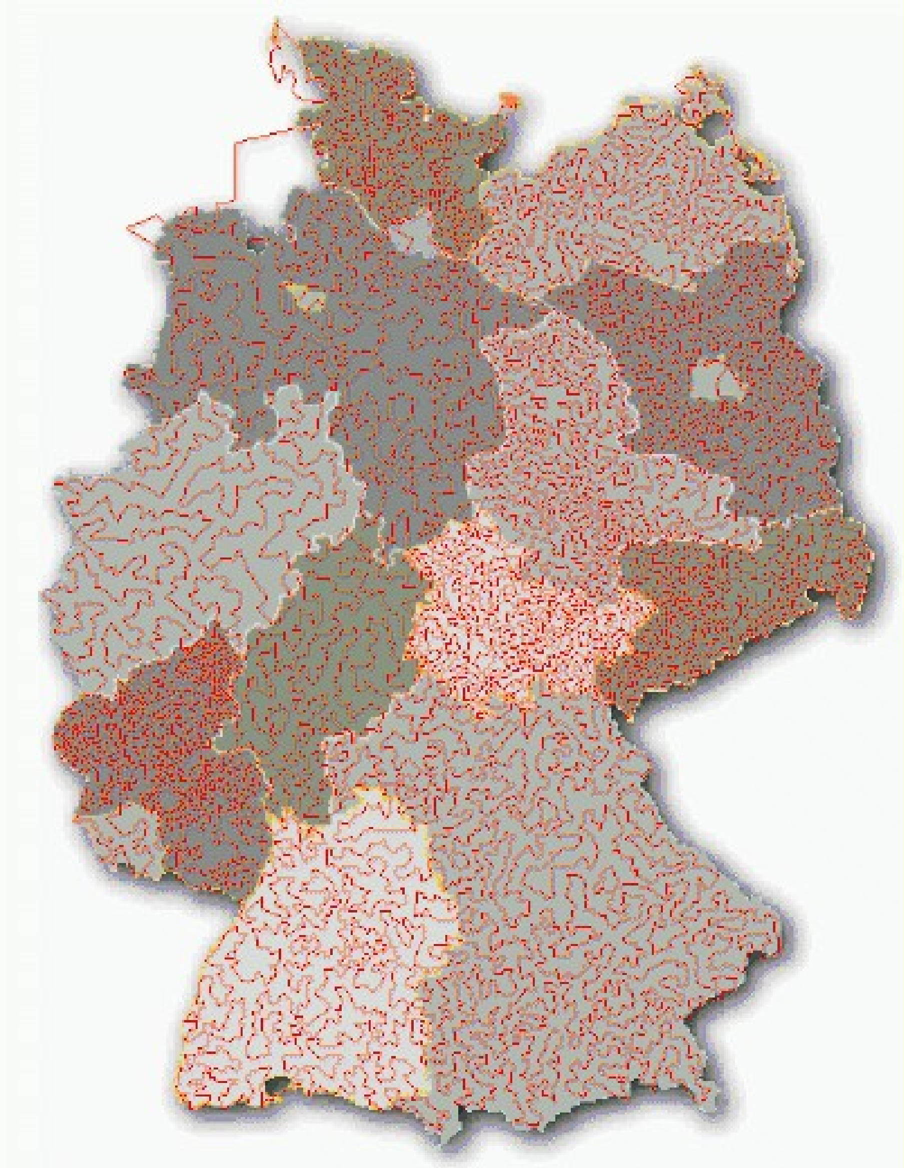
13509 Cities in the USA

- Applegate, Bixby, Chvátal, and Cook (1998) found the optimal tour of the 13,509 cities in the USA with populations greater than 500.



15112 Cities in Germany

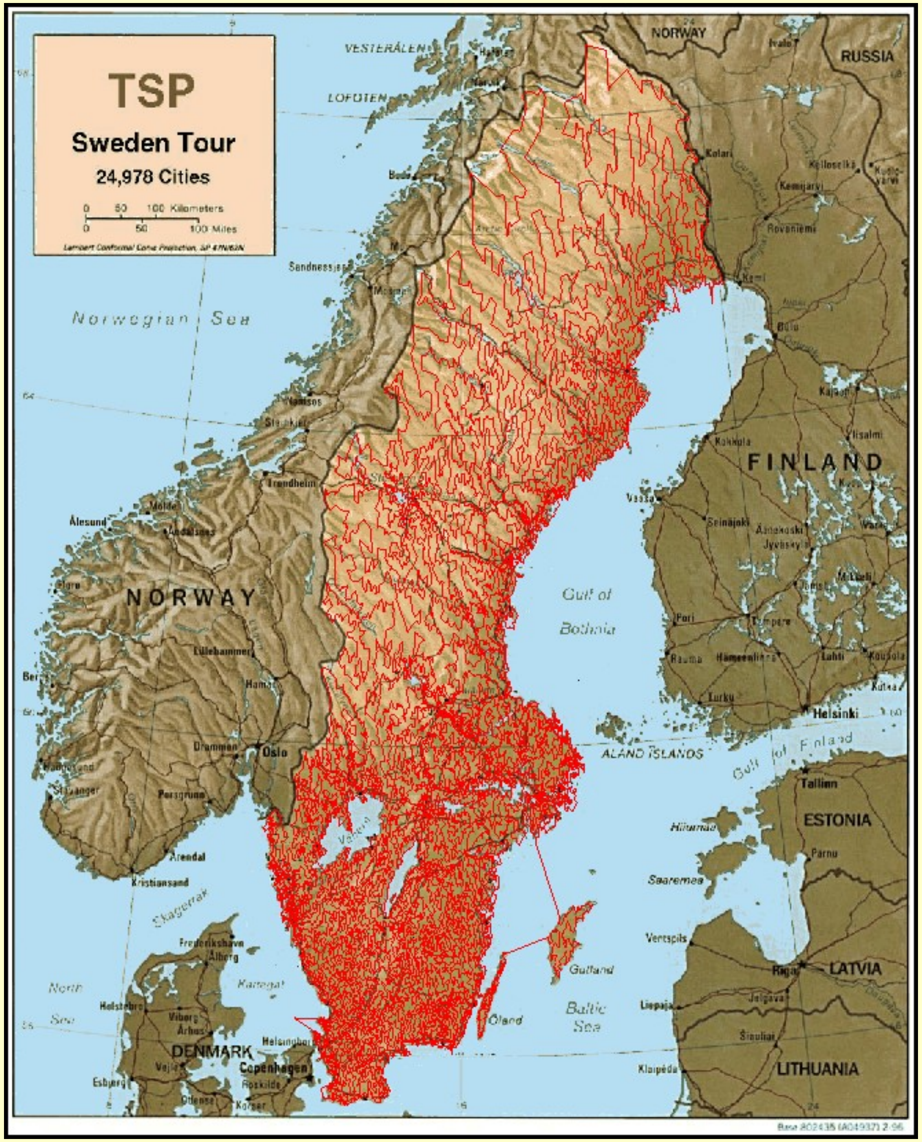
- Applegate, Bixby, Chvátal, and Cook (2001) found the optimal tour of 15,112 cities in Germany.



Toán rời rạc

24978 Swedish Cities

- Applegate, Bixby, Chvátal, Cook, and Helsgaun (2004) found the optimal tour of 24,978 cities in Sweden.



Nguyễn Đức Nghĩa

Nguyễn Đức Nghĩa

Optimal Tour of Sweden

- In May 2004, the traveling salesman problem of visiting all 24,978 cities in Sweden was solved: a tour of length 855,597 TSPLIB units (approximately 72,500 kilometers) was found and it was proven that no shorter tour exists. This is currently the largest solved TSP instance, surpassing the previous record of 15,112 cities through Germany set in April 2001.

Nguyễn Đức Nghĩa

Optimal Tour of Sweden

- Research Team
  - David Applegate, AT&T Labs - Research
  - Robert Bixby, ILOG and Rice University
  - Vašek Chvátal, Rutgers University
  - William Cook, Georgia Tech
  - Keld Helsgaun, Roskilde University

- Support for this research was provided by the following grants
  - Office of Naval Research Grant N00014-03-1-0040, "Experimental Modules for Combinatorial Optimization and Mixed-Integer Programming"
  - National Science Foundation, Grant DMI-0245609, "Local Cuts in Discrete Optimization and Mixed-Integer Programming"

Finding Sweden Tour

- The traveling salesman problem (TSP) asks for the cheapest possible tour through a given collection of cities. Solving the problem means to not only find the best tour but also to prove that no cheaper tour is possible. Early work on the TSP in the 1950s focused exclusively on the this full solution of the problem.
- Starting in the mid-1960s researchers began to study the relaxed version of the TSP where we ask only for a tour of low cost. This task is much easier, but performing it well is an important ingredient in a full (exact) solution method, as well as being an interesting problem in its own right. Indeed, tour finding is a very popular topic, having a large and growing literature devoted to its various aspects. And like the TSP itself, tour finding has led researchers to discover general purpose search techniques that have found application in many domains.
- The Sweden TSP was attacked by a number of groups with some of the top tour-finding methods that have been developed to date. Information on the improvements in the best known tour length can be found in the Sweden Computation Log; the results are summarized in the following table.

Nguyễn Đức Nghĩa

Finding Sweden Tour

855618	September 4, 2001	Tour Merging	Cook and Seymour
855612	September 20, 2001	LKH	Helsgaun
855610	September 30, 2001	LKH Merge	Helsgaun
855602	March 16, 2003	Hybrid Genetic	Hung Dinh Nguyen
855597	March 18, 2003	LKH	Helsgaun

- The final improvement in the tour length was made by Keld Helsgaun using a version of his LKH code. This 855,597 value was proved to be optimal by the Concorde TSP code.

Nguyễn Đức Nghĩa

Finding Sweden Tour

- The Concorde solver can accept as an input parameter the value of the best known tour for a TSP instance if one is available. As a full (exact) TSP solver, Concorde is designed to find optimal solutions regardless of the quality of the estimate, but knowledge of a good tour allows for better tuning of parameters that are set in the computer code.

- In the case of the Sweden TSP, the results of the tour-finding attacks guided our choices in approaching the full solution of the problem. Most importantly, the final stages that improved the lower bound from 855,595 up to the optimal value 855,597 required approximately 8 years of computation time (running in parallel on a network of Linux workstations) and without knowledge of the 855,597 tour we would not have make the decision to carry out this final computation.

Nguyễn Đức Nghĩa

New record: 85900 *cities,* 2006

- The largest solved instance of the traveling salesman problem consists of a tour through 85,900 cities in a VLSI application that arose in Bell Laboratories in the late 1980s.
- The computation with Concorde was carried out in 2005/06 and reported in the book The Traveling Salesman Problem: A Computational Study. The instance is called pla85900 in Gerd Reinelt's TSPLIB; the shortest possible tour for the problem has length 142,382,641 units.
- With the solution of pla85900, the complete TSPLIB collection of challenge problems has now been successfully solved with the Concorde code.
- http://www.tsp.gatech.edu/index.html

Nguyễn Đức Nghĩa

Picture of *pla85900* tour

15 year race for better tours

Date	Tour Length	Research Team	Method
07.06.1991	142,514,146	David S. Johnson	Iterated Lin-Kernighan
29.03.1996	142,487,006	Concorde	Tour Merging
23.09.1997	142,482,068	Concorde	Tour Merging
14.10.1998	142,416,327	Keld Helsgaun	LKH
22.10.1999	142,409,553	Concorde	Tour Merging
18.06.2001	142,406,493	Keld Helsgaun	LKH
27.06.2001	142,405,532	Keld Helsgaun	LKH
31.08.2001	142,395,130	Concorde	Tour Merging with LKH
14.12.2001	142,393,738	Keld Helsgaun	LKH
15.09.2002	142,385,237	Hisao Tamaki	Approximate Tour Merging
12.12.2002	142,383,704	Keld Helsgaun	LKH
19.03.2003	142,383,467	Nguyen Dinh Hung	Hybrid Genetic Algorithm
28.04.2003	142,383,189	Keld Helsgaun	LKH
23.12.2003	142,383,011	Keld Helsgaun	LKH
02.05.2004	142,382,641	Keld Helsgaun	LKH

Nguyễn Đức Nghĩa

Questions?

Nguyễn Đức Nghĩa

Merci à tous !


Toán rời rạc
Nguyễn Đức Nghĩa