Phương pháp tham lam

Bởi:

Đại Học Phương Đông

Đặc trưng của chiến lược tham lam

Bài toán tối ưu tổ hợp

- Là một dạng của bài toán tối ưu, nó có dạng tổng quát như sau:
- Cho hàm f(X) = xác định trên một tập hữu hạn các phần tử D. Hàm f(X) được gọi là hàm mục tiêu.
- Mỗi phần tử X Є D có dạng X = (x1, x2... xn) được gọi là một phương án.
- Cần tìm một phương án X Є D sao cho hàm f(X) đạt min (max). Phương án X như thế được gọi là phương án tối ưu.

Ta có thể tìm thấy phương án tối ưu bằng phương pháp "vét cạn" nghĩa là xét tất cả các phương án trong tập D (hữu hạn) để xác đinh phương án tốt nhất. Mặc dù tập hợp D là hữu hạn nhưng để tìm phương án tối ưu cho một bài toán kích thước n bằng phương pháp "vét cạn" ta có thể cần một thời gian mũ. Các phần tiếp theo của chương này sẽ trình bày một số kĩ thuật giải bài toán tối ưu tổ hợp mà thời gian có thể chấp nhận được.

Nội dung kĩ thuật tham ăn

Tham ăn hiểu một cách dân gian là: trong một mâm có nhiều món ăn, món nào ngon nhất ta sẽ ăn trước và ăn cho hết món đó thì chuyển sang món ngon thứ hai, lại ăn hết món ngon thứ hai này và chuyển sang món ngon thứ ba… Kĩ thuật tham ăn thường được vận dụng để giải bài toán tối ưu tổ hợp bằng cách xây dựng một phương án X. Phương án X được xây dựng bằng cách lựa chọn từng thành phần Xi của X cho đến khi hoàn chỉnh (đủ n thành phần). Với mỗi Xi, ta sẽ chọn X_i tối ưu. Với cách này thì có thể ở bước cuối cùng ta không còn gì để chọn mà phải chấp nhận một giá trị cuối cùng còn lại.

Áp dụng kĩ thuật tham ăn sẽ cho một giải thuật thời gian đa thức, tuy nhiên nói chung chúng ta chỉ đạt được một phương án tốt chứ chưa hẳn là tối ưu. Có rất nhiều bài toán mà ta có thể giải bằng kĩ thuật này.

Đặc tính lựa chọn tham lam

Toàn bộ phương pháp tối ưu có thể đạt được từ việc chọn tối ưu trong từng bước chọn.

Về khía cạnh này giải thuật tham lam khác với giải thuật quy hoạch động ở chỗ: Trong qui hoạch động chúng ta thực hiện chọn cho từng bước, nhưng việc lựa chọn này phụ thuộc vào cách giải quyết các bài toán con. Với giải thuật tham lam, tại mỗi bước chúng ta chọn bất cứ cái gì là tốt nhất vào thời điểm hiện tại, và sau đó giải quyết các vấn đề phát sinh từ việc chọn này. Vấn đề chọn thực hiện bởi giải thuật tham lam không phụ thuộc vào việc lựa chọn trong tương lai hay cách giải quyết các bài toán con. Vì vậy khác với quy hoạch động, giải quyết các bài toán con theo kiểu bottom up (từ dưới lên), giải thuật tham lam thường sử dụng giải pháp top-down (từ trên xuống). Chúng ta phải chứng minh rằng với giải thuật tham lam, toàn bộ bài toán được giải quyết một cách tối ưu nếu mỗi bước việc chọn được thực hiện tối ưu. Các bước chọn tiếp theo được thực hiện tương tự như bước đầu tiên, nhưng với bài toán nhỏ hơn. Ph- ương pháp qui nạp được ứng dụng trong giải thuật tham lam có thể được sử dụng cho tất cả các bước chọn

Cấu trúc con tối ưu

Một bài toán thực hiện optimal substructure nếu cách giải quyết tối ưu của bài toán chứa đựng cách giải quyết tối ưu những bài toán con của nó. Tính chất này được đánh giá là một thành phần có thể áp dụng được của thuật toán quy hoạch động tốt như thuật toán tham lam. Một ví dụ của optimal substructure, nếu A là đáp án tối ưu của bài toán với hành động chọn đầu tiên là 1, thì tập hợp A'= A- {1} là đáp án tối ưu cho bài toán

S'= {i Є S: $s_i \geq f_1$ }.

Sơ đồ chung của phương pháp

Đặc điểm chung của thuật toán tham lam

Mục đích xây dựng bài toán giải nhiều lớp bài toán khác nhau, đưa ra quyết định dựa ngay vào thuật toán đang có, và trong tương lai sẽ không xem xét lại quyết định trong quá khứ. Vì vậy thuật toán dễ đề xuất, thời gian tính nhanh nhưng thường không cho kết quả đúng.

- Lời giải cần tìm có thể mô tả như là bộ gồm hữu hạn các thành phần thoả mãn điều kiện nhất định, ta phải giải quyết bài toán một cách tối ưu -> hàm mục tiêu
- Để xây dựng lời giải ta có một tập các ứng cử viên
- Xuất phát từ lời giải rỗng, thực hiện việc xây dựng lời giải từng bước, mỗi bước sẽ lựa chọn trong tập ứng cử viên để bổ xung vào lời giải hiện có.
- Xây dựng một hàm nhận biết tính chấp nhận được của lời giải hiện có -> Hàm Solution(S) -> Kiểm tra thoả mãn điều kiện chưa.

Một hàm quan trọng nữa: Select(C) cho phép tại mỗi bước của thuật toán lựa chọn ứng cử viên có triển vọng nhất để bổ xung vào lời giải hiện có -> dựa trên căn cứ vào ảnh hưởng của nó vào hàm mục tiêu, thực tế là ứng cử viên đó phải giúp chúng ta phát triển tiếp tục bài toán.

Xây dựng hàm nhận biết tính chấp nhận được của ứng cử viên được lựa chọn, để có thể quyết định bổ xung ứng cử viên được lựa chọn bởi hàm Select vào lời giải -> Feasible(S x).

Sơ đồ thuật toán

Procedure Greedy;

*{*G i ả sử C l à t ập các ứng cử viên*}*

begin

S :=Ø ; { S là l ời giải xây dựng theo thuật toán *}*

While(C≠ 0)andnotSolution(S)do

Begin

x ← size 12{ leftarrow } {}Select(C);

C:=C\x;

If feasible(S x) then S:=S x

End;

If solution(S) then return S;

End;

Chứng minh tính đúng đắn

- Công việc này không phải đơn giản. Ta sẽ nêu một lập luận được sử dụng để chúng minh tính đúng đắn.
- Để chỉ ra thuật toán không cho lời giải đúng chỉ cần đưa ra một phần ví dụ
- Việc chứng minh thuật toán đúng khó hơn nhiều và ta sẽ nghiên cứu cụ thể trong phần sau:

Lập luận biến đổi (Exchange A r gument)

Giả sử cần chứng minh thuật toán A cho lời giải đúng. A(I) là lời giải tìm được bởi thuật toán A đối với bộ dữ liệu I. Còn O là lời giải tối ưu của bài toán với bộ dữ liệu này.

Ta cần tìm cách xây dựng phép biến đổi φ để biến đổi O thành O' sao cho:

1. O' cũng tốt không kém gì O (Nghĩa là O' vẫn tối ưu)
2. O' giống với A(I) nhiều hơn O.

Giả sử đã xây dựng được phép biến đổi vừa nêu. Để chứng minh tính đúng đắn dựa vào hai sơ đồ chứng minh sau

- CMbằngphảnchứng: Giả sử A không đúng đắn, hãy tìm bộ dữ liệu I sao cho A(I) khác với lời giải tối ưu của bài toán. Gọi O là lời giải tối ưu giống với A(I) nhất => A(I) khác O. Dùng phép biến đổi φ chúng ta có thể biến đổi O → O' sao cho O' vẫn tối ưu và O' giống với A(I) hơn => mâu thuẫn giả thiết O là lời giải tối ưu giống với A(I) nhất.

-CMtrựctiếp:O là lời giải tối u. Biến đổi O → O' giống với A(I) hơn là O. Nếu O' = A(I) thì A(I) chính là phương án tối u ngược lại biến đổi O' → O'' giống với A(I) hơn. Cứ thế ta thu được dãy O', O'', O'''….. ngày càng giống hơn, và chỉ có một số hữu hạn điều kiện để so sánh nên chỉ sau một số hữu hạn lần phép biến đổi

sẽ kết thúc và đó là tại A(I).

Bài toán trả tiền của máy rút tiền tự động ATM

Trong máy rút tiền tự động ATM, ngân hàng đã chuẩn bị sẵn các loại tiền có mệnh giá 100.000 đồng, 50.000 đồng, 20.000 đồng và 10.000 đồng. Giả sử mỗi loại tiền đều có số lượng không hạn chế. Khi có một khách hàng cần rút một số tiền n đồng (tính chẵn đến 10.000 đồng, tức là n chia hết cho 10000). Hãy tìm một phương án trả tiền sao cho trả đủ n đồng và số tờ giấy bạc phải trả là ít nhất.

Gọi X = (X1, X2, X3, X4) là một phương án trả tiền, trong đó X1 là số tờ giấy bạc mệnh giá 100.000 đồng, X2 là số tờ giấy bạc mệnh giá 50.000 đồng, X3 là số tờ giấy bạc mệnh giá 20.000 đồng và X4 là số tờ giấy bạc mệnh giá 10.000 đồng. Theo yêu cầu ta phải có X1 + X2 + X3 + X4 nhỏ nhất và X1 * 100.000 + X2 * 50.000 + X3 *

20.000 + X4 * 10.000 = n.

Áp dụng kĩ thuật tham ăn để giải bài toán này là: để có số tờ giấy bạc phải trả (X1 +

X2 + X3 + X4) nhỏ nhất thì các tờ giấy bạc mệnh giá lớn phải được chọn nhiều nhất. Trước hết ta chọn tối đa các tờ giấy bạc mệnh giá 100.000 đồng, nghĩa là X1 là số nguyên lớn nhất sao cho X1 * 100.000 ≤ n. Tức là X1 = n DIV 100.000.

Xác định số tiền cần rút còn lại là hiệu n – X1 * 100000 và chuyển sang chọn loại giấy

bạc 50.000 đồng…

Khách hàng cần rút 1.290.000 đồng (n = 1290000), phương án trả tiền như sau: X1 = 1290000 DIV 100000 = 12.

Số tiền cần rút còn lại là 1290000 – 12 * 100000 = 90000. X2 = 90000 DIV 50000 = 1.

Số tiền cần rút còn lại là 90000 – 1 * 50000 = 40000. X3 = 40000 DIV 20000 = 2.

Số tiền cần rút còn lại là 40000 – 2 * 20000 = 0. X4 = 0 DIV 10000 = 0.

Ta có X = (12, 1, 2, 0), tức là máy ATM sẽ trả cho khách hàng 12 tờ 100.000 đồng, 1 tờ 50.000 đồng và 2 tờ 20.000 đồng.

Bài toán về các đoạn thẳng không giao nhau

Bài toán

$C = \{(a_1, b_1), (a_2, b_2) \ldots\ldots\ldots, (a_n, b_n)\}$ = Đầu vào : Cho họ các đoạn thẳng mở

Đầu ra : Tập các đoạn thẳng không giao nhau có lực lượng lớn nhất.

Ứngdụngthựctế: Bài toán xếp thời gian biểu cho các hội thảo, bài toán phục vụ khách hành trên một máy, bài toán lựa chọn hành động (Ví dụ có *n*lời mời dự tiệc bắt đầu bởi *ai*kết thúc bởi *bi*, hãy lựa chọn sao cho đi được nhiều tiệc nhất).

Đề xuất các thuật to á n :

Greedy 1: Sắp xếp các đoạn thẳng theo thứ tự tăng dần của đầu mút trái, bắt đầu từ tập S là tập rỗng ta lần lượt xếp các đoạn thẳng trong danh sách theo thứ tự đã xếp và bổ sung đoạn thẳng đang xét vào S nếu nó không có điểm chung với bất cứ đoạn nào trong S.

Thuật toán :

Procedure Greedy1; Begin

*S:=Ø;{*Slàtậpcácđoạnthẳngcần tìm*}*

<Sắp xếp các đoạn th ẳ ng trong C theo thứ tự không giảm của nút tr á i>

WhileC≠ 0do

Begin

End;

(ac,bc)Є đoạnđầu tiêntrongC; C:=C\(ac,bc);

If <(ac, bc) không gi a o với bất cứ đoạn n ào trong s> th e n

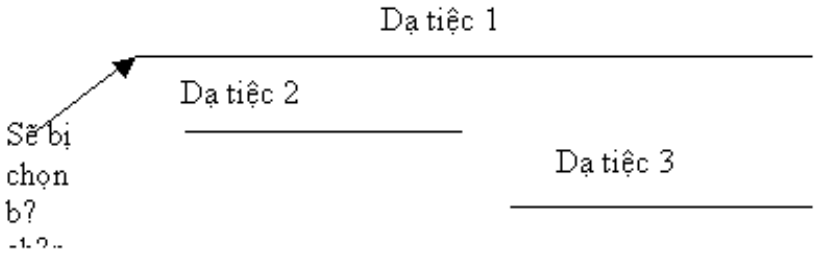
S := S (ac, bc)

End;

<S là t ập cần t ìm>

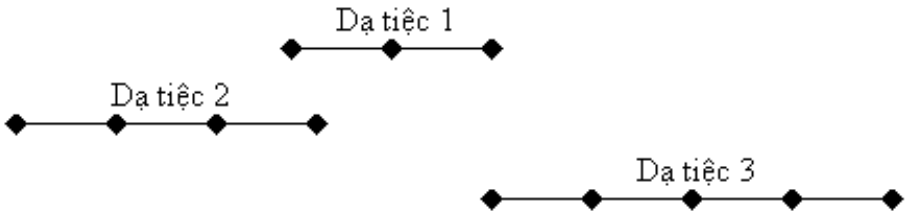
Độ phức tạp của thuật toán là $O(m \log n)$ - > nằm trong đoạn sắp xếp

Tuy nhiên Greedy1 không cho lời giải tối ưu. Ví dụ sau



Ta thấy rằng thuật toán sẽ lựa chọn dạ tiệc 1, trong khi phương án tối ưu của bài toán là (Dạ tiệc 2, Dạ tiệc 3)

Greedy2: Ta chọn đoạn có độ dài ngắn nhất bổ xung vào S. Tuy nhiên thuật toán tham lam này cũng không cho kết quả tối ưu. Sau đây là phản ví dụ



Khi đó thuật toán sẽ lựa chọn (dạ tiệc 1) trong khi lời giải tối ưu của thuật toán là (dạ tiệc 2, dạ tiệc 3).

Greedy3:Xắp xếp các đoạn thẳng theo thứ tự không giảm của mút phải. Bắt đầu từ tập S là tập rỗng ta lần lượt xét các đoạn trong danh sách theo thứ tự đã sắp xếp và bổ xung đoạn thẳng đang xét vào S nếu nó không có điểm chung với bất cứ đoạn nào trong S. (Dạ tiệc nào kết thúc sớm sẽ được xét trước).

Mệnhđề1: Thuật toán Greedy3 cho lời giải tối ưu của bài toán về các đoạn thẳng không giao nhau.

ChứngMinh: Giả sử Greedy3 không cho lời giải đúng. Phải tìm bộ dữ liệu C sao cho thuật toán không cho lời giải tối u. Giả sử G3(C) là lời giải tìm được bởi Greedy3. Gọi O là lời giải tối ưu có số đoạn thẳng chung với G3(C) là lớn nhất. Gọi X là đoạn thẳng đầu tiên có trong G3(C) nhưng không có trong O. Đoạn này là tồn tại, vì nếu trái lại thì G3(C) ≡ O (mâu thuẫn vì đã giả thiết G3(C) ≠ O) hay G3(C) Є O (Cũng mâu thuẫn vì khi đó thuật toán phải chọn đoạn thẳng X) (O cũng được sắp xếp giống G3(C)).

Gọi Y là đoạn đầu tiên kể từ bên trái của O không có mắt trong G3(C). Đoạn Y cũng phải tồn tại (Chứng minh tương tự như trên).

Khi đó mút phải của đoạn X phải ở bên trái (nhỏ hơn) mút phải của đoạn Y, vì nếu trái lại thuật toán sẽ chọn Y thay vì X.

$O' = O \setminus \{Y\} \cup \{X\}$ Xét

Rõ ràng

- O’ gồm các đoạn thẳng không giao với nhau, bởi vì X không giao với bất kì đoạn nào ở bên trái nó trong O’ (do G3(C) là chấp nhận được) cũng như không giao với bất cứ đoạn nào ở bên phải nó trong O’ (Do mút phải của X nhỏ

hơn mút phải của Y và Y không giao với bất cứ đoạn nào ở bên phải Y trong O').

- Do O' có cùng lực lượng với O nên O' cũng là tối ưu
- Tuy nhiên ta thấy rằng O' giống với G3(C) hơn là O => mâu thuẫn với giả thiết.

Bài toán cái túi

Bài toán: cho n đồ vật, trong lượng tương ứng của từng đồ vật là *wi*, và giá trị là *ci*(),

Ta chất đồ vật vào túi có trọng lượng b, sao cho tổng trọng lượng không vượt quá b và đạt giá trị lớn nhất.

Ta có thể tóm tắt bài toán như sau :

$C = \{1,2,\ldots,n\}$: tập chỉ số của đồ vật.

Tìm $I \subset C$. Sao cho $\sum_{i \in I} w_i \le b$; $\sum_{i \in I} c_i \to \max$

Đề xuất thuật toán t h am lam

Greedy1: Sắp xếp theo thứ tự không tăng của giá trị. Xét các đồ vật theo thứ tự đã xếp, lần lượt chất các đồ vật đang xét vào túi nếu dung lượng còn lại trong túi đủ chứa nó. Thuật toán tham lam này không cho lời giải tối ưu. Sau đây là phản ví dụ:

Tham số của bài toán là n = 3; b = 19.

Đồ vật 1 2 3

Giá trị 20 16 8 -> *giátrị lớnnhưngtrọnglượngcũngrấtlớn*

Trọng lượng 14 6 10

Thuật toán sẽ lựa chọn đồ vật 1 với tổng giá trị là 20, trong khi lời giải tối ưu của bài toán là lựa chọn (đồ vật 2, đồ vật 3) với tổng giá trị là 24.

Greedy2: Sắp xếp đồ vật không giảm của trọng lượng. Lần lượt chất các đồ vật vào túi theo thứ tự đã sắp xếp. Thuật toán tham lam này cũng không cho kết quả tối ưu. Sau đây là phản ví dụ

Tham số của bài toán là n = 3; b = 11

Đồ vật 1 2 3

Giá trị 10 16 28 ->*Đồvậtnhẹnhưnggiá tiềncũngrấtnhẹ*

Trọng lượng 5 6 10

Thuật toán sẽ lựa chọn (đồ vật 1, đồ vật 2) với tổng giá trị là 26, trong khi lời giải tối ưu của bài toán là (đồ vật 3) với tổng giá trị là 28.

Greedy3: Sắp xếp các đồ vật theo thứ tự không tăng của giá trị một đơn vị trọng 1lượng (*ci*/*wi*). Lần lượt xét

$$\frac{c_1}{w_1} \geq \frac{c_2}{w_2} \geq \ldots\ldots\ldots \geq \frac{c_n}{w_n}$$

Tuy nhiên Greedy3 không cho lời giải tối ưu. Sau đây là phản ví dụ của bài toán

Tham số của bài toán : $n = 2$; $b \geq 2$.

$$\frac{c_1}{w_1} = \frac{10}{1} \geq \frac{10b-1}{b} = \frac{c_2}{w_2}$$

Khi đó thuật toán chỉ lựa chọn được đồ vật 1 với tổng giá trị là 10, trong khi lời giải tối ưu của bài toán lựa chọn đồ vật 2 với tổng giá trị là 10b-1 ($\geq 10.2-1 = 19 > 10$).

Greedy4:Gọi Ij là lời giải thu được theo thuật toán Greedyj (j = 1, 2, 3). Gọi

$$I = \max\left\{\sum_{i \in I_1} c_i, \sum_{i \in I_2} c_i, \sum_{i \in I_3} c_i\right\}$$

Định lý : Lời giải I4 thoả mãn bất đẳng thức

$$\sum_{i \in I_4} c_i \geq \frac{1}{2} f^*$$

Trong đó f^* là giá trị tối u của bài toán.