XÂY DỰNG THUẬT TOÁN QUY HOẠCH TUYẾN TÍNH DỰA TRÊN GIA LƯỢNG NGẪU NHIÊN

Linear Programming Algorithm based on random Increment

Nguyễn Khắc Quốc[1]

Tóm tắt

Quy hoạch tuyến tính có một vị trí quan trọng trong tối ưu hóa với hai lý do: thứ nhất, mô hình tuyến tính đơn giản, dễ áp dụng; thứ hai, nhiều bài toán quy hoạch nguyên và quy hoạch phi tuyến có thể xấp xỉ với độ chính xác cao bởi một dãy các bài toán quy hoạch tuyến tính. Trong bài báo, chúng tôi giới thiệu thuật toán gia lượng ngẫu nhiên để giải quyết bài toán này.

Từ khóa: quy hoạch tuyến tính, gia lượng ngẫu nhiên, ngẫu nhiên, quy hoạch phi tuyến.

Abstract

Linear Programming plays a very important role in optimization because of the two following reasons. First, its linear models are simple and easily applicable. Second, many mathematical problems which are original and non linear can be solved with approximately high accuracy by a series of linear programming ones. In this article, it will explore how to use random incremental algorithm to solve mathematical problems.

Keys words: linear programming, Random Increment, random, non linear programming.

1. Giới thiệu

Quy hoạch tuyến tính là một trong những lớp bài toán được nghiên cứu đầy đủ cả về mặt lý thuyết lẫn thực tiễn. Nó bắt nguồn từ những nhóm nghiên cứu của nhà toán học Nga nổi tiếng - Viện sĩ Kantorovich L.V. Ông đã nêu trong một loạt công trình về bài toán kế hoạch hóa sản xuất được công bố năm 1938. Năm 1947, nhà toán học Mỹ Dantzig đã nghiên cứu và đề xuất phương pháp đơn hình (*Simplex method*) để giải bài toán này. Năm 1952 phương pháp đơn hình đã được chạy trên máy tính điện tử ở Mỹ.

2. Nội dung

2.1. Mô tả bài toán

Bài toán quy hoạch tuyến tính là tìm cực trị hàm mục tiêu tuyến tính của các biến thực với hàm tuyến tính của nhiều biến. Trong bài báo này, chúng tôi gọi d chứa các biến số và n là số các ràng buộc. Mỗi ràng buộc n mô tả nửa - không gian trong không gian d - *chiều* với điều kiện là các điểm cực trị bị giới hạn trong nửa - không gian này.

Giao của các nửa - không gian này là một đa diện trong không gian d - *chiều* (có thể rỗng hoặc không có đường biên). Chúng ta có thể quy về "*vùng có thể thực hiện*" (*feasible region*). Tuy nhiên, chúng ta sẽ giới hạn lại số chiều của bài toán này.

Gọi $x_1, ..., x_d$ gồm d biến trong bài toán quy hoạch tuyến tính. Gọi $c_1, ..., c_d$ là các hệ số của những biến trong hàm mục tiêu, gọi A_{ij} với $1 \le i \le n$ và $1\ j \le d$ chứa hệ số x_j trong ràng buộc thứ i. Gọi $\boldsymbol{A}$ là ma trận (A_{ij}), vectơ $c(c_1, ..., c_d)$, và vectơ $x(x_1, ..., x_d)$. Bài toán được phát biểu như sau:

$$\begin{cases} C^T \text{ (nhỏ nhất)} & (2.1) \\ \boldsymbol{Ax} \le \boldsymbol{b} \quad \text{với } \boldsymbol{b} \text{ là vectơ cột.} & (2.2) \end{cases}$$

Gọi F(***A, b***) là *vùng có thể thực hiện*, được xác định bởi $\boldsymbol{A}$ và $\boldsymbol{b}$. Vectơ có hướng trong không gian d. Xét về mặt hình học, chúng ta sẽ tìm một điểm ngoài F(***A, b***) theo phương ngược lại đến c nếu như tồn tại một điểm xác định, thì:

1. Đa diện F(***A, b***) là không rỗng và có đường biên.
2. Cực tiểu hóa hàm mục tiêu x_1 hay $c = (1, 0, ..., 0)$.

Khi đó, chúng ta tìm một điểm trong F(***A, b***) với giá trị cực tiểu x.

3. Giá trị cực tiểu tìm thấy tại một điểm duy nhất là một đỉnh của F(***A, b***).
4. Với mỗi đỉnh của F(***A, b***) được xác định bằng một hằng số d.

Gọi H chứa tập các ràng buộc được xác định bởi $\boldsymbol{A}$ và $\boldsymbol{b}$. Gọi $S \subseteq H$ là tập con ràng buộc trong H. Trong bài toán quy hoạch tuyến tính đang xét, ta xác định tập con S cùng với c, khi chương trình tuyến tính đạt tới giới hạn cực tiểu. Vì vậy, mục 3 – 4 ở trên vẫn còn thỏa:

[1] *Thạc sĩ - Bộ môn Công nghệ Thông tin, Trường Đại học Trà Vinh*

(i) Cực tiểu xảy ra tại một đỉnh duy nhất.

(ii) Với mỗi đỉnh của *vùng có thể thực hiện* được xác định bởi ràng buộc *d*.

Gọi O(*S*) là giá trị của hàm mục tiêu được xác định bởi *c* và *S* (có thể O(*S*) = -∞). *B* là một tập ràng buộc *cơ sở*, O(*B*) > - ∞ và O(*B'*) > O(*B*). Cho *B'* ⊂ *B*. Cơ sở của *H* là B(*H*) khi B(*H*) được xác định là đỉnh tối ưu nhất. Có thể gọi B(*H*) hoặc O(B(*H*)) là tối ưu của bài toán quy hoạch tuyến tính.

Để giải quyết bài toán quy hoạch tuyến tính trên ta dùng thuật toán *giao của nửa - không gian* để tìm F(***A, b***) và sau đó là tìm giá trị hàm mục tiêu tại mỗi đỉnh của đa diện F(***A, b***). Như vậy, tổng số tiến trình được đánh giá là rất thấp, khi đó số các đỉnh của F(***A, b***) có thể là $\Omega(n^{[d/2]})$.

2.2. Phân tích bài toán

Để tiến hành nghiên cứu một thuật toán ngẫu nhiên cho bài toán quy hoạch tuyến tính, chúng ta hãy gọi lại các phần tử của thuật toán đơn hình. Đây là một thuật toán tất định, bắt đầu từ một đỉnh của F(***A, b***) với mỗi dãy con được gọi lại, các tiến trình đến đỉnh lân cận - nơi mà hàm mục tiêu có giá trị thấp hơn. Nếu như đỉnh đó không tồn tại thì chúng đạt được giá trị cực tiểu - đây là vấn đề chủ yếu của thuật toán đơn hình. Một vấn đề phức tạp nảy sinh khi các đỉnh lân cận có giá trị hàm mục tiêu giống nhau, như vậy rất khó xác định được cực tiểu. Chúng ta xây dựng hàm **Simplex** trong bài toán quy hoạch tuyến tính bằng cách duyệt qua các đỉnh của F(***A, b***) và lặp lại việc này cho đến khi tìm được tối ưu, nếu như tồn tại.

Gọi ràng buộc $h \in H$ đạt *cực trị* nếu O($H\setminus\{h\}$) < O(*H*). Thật vậy, nó ràng buộc trong B(*H*). Bằng trực giác cho thấy, sự vắng mặt của nó sẽ không làm thay đổi tính tối ưu. Thuật toán **SampLP** đầu tiên dùng mẫu ngẫu nhiên dẫn đến ràng buộc dư thừa rất nhanh.

Bắt đầu từ một tập rỗng, **SampLP** ta xây dựng lại một tập ràng buộc *S* chứa một chuỗi các pha. Trong mỗi pha, một tập $V \subset H \setminus S$ được thêm vào *S*. Tập *V* sẽ có hai thuộc tính quan trọng:

(i) Rất nhỏ.

(ii) Chứa tất cả ràng buộc cực trị nhỏ nhất từ B(*H*) không nằm trong *S*. Khi |B(*H*)|= *d*, kết thúc hầu hết *d* pha sau đó.

Hàm **SampLP** được mô tả bên dưới và được thực hiện chi tiết hơn trong thuật toán **InterSampLP**. Chúng ta sẽ phân tích **InterSampLP** sau.

2.3. Thuật toán SampLP

2.3.1. Mô tả thuật toán SampLP

Algorithm SampLP
Input: Một tập ràng buộc *H*
Output: B(*H*) tối ưu
1. $S \leftarrow \varnothing$;
2. **if** $n < 9d^2$
return Simplex (*H*)
else
2.1 $V \leftarrow H$; $S \leftarrow \varnothing$;
2.2 **While** ($|V| > 0$)
Chọn ngẫu nhiên $R \subset H \setminus S$, với $|R| = r = \min\{d\sqrt{n}, |H\setminus S|\}$;
$x \leftarrow$ **SampLP** ($R \cup S$);
$V \leftarrow \{h \in H \mid$ đỉnh *x* được xác định bởi vi phạm *h*$\}$;
if ($|V| \le 2\sqrt{n}$)
then $S \leftarrow S \cup V$;
2.3 **reurn** *x*;

2.3.2. Phân tích và đánh giá thuật toán

Thật vậy, với $n > 9d^2$ **SampLP** chọn ngẫu nhiên tập ràng buộc r ⊂ *R*, giá trị của *r* là $d\sqrt{n}$, trừ khi $H\setminus S$ chứa nhiều hơn $d\sqrt{n}$ ràng buộc. Giải bài toán này bằng phương pháp đệ quy, được xác định bởi $R \cup S$ và xác định một tập $V \subset H$ là vi phạm ràng buộc tính tối ưu này; chú ý rằng việc vi phạm ràng buộc xảy ra từ trong $H\setminus S$. Nếu *V* không có nhiều hơn $2\sqrt{n}$ phần tử, chúng ta thêm *V* vào *S*, khi đó *V* trở thành rỗng (nghĩa là B(*H*) được chứa trong *S*), chúng ta quay về *x*.

Thủ tục **Simplex** được gọi chỉ với $9d^2$ hoặc một số khá nhiều ràng buộc. Với mỗi bài toán quy hoạch tuyến tính được gọi là "*Small*", chúng ta đánh giá việc gọi **Simplex** như sau:

Tổng số đỉnh của đa diện cho bài toán này không nhiều hơn $\binom{9d^2}{[d/2]}$, lớn nhất là $(4.9d)^{[d/2]}$.

Đó là một hằng số *a* so với thuật toán đơn hình dùng một thời gian nhiều nhất là d^a cho mỗi đỉnh, vì thế chúng ta có hai hệ quả sau:

Hệ quả 1:

*Đánh giá tổng việc gọi **Simplex** với $9d^2$ hoặc một số khá nhiều ràng buộc là* $\mathbf{O(d^{d/2+a})}$.

Kế tiếp, chúng ta chứng tỏ rằng V - một tập của các vi phạm ràng buộc x, là nhỏ.

Hệ quả 2:

Gọi $S \subseteq H$, và $R \subseteq H\backslash S$ là một tập con ngẫu nhiên mức r. Gọi $|H\backslash S|$ chứa m. Số các vi phạm ràng buộc của H bởi $O(R \cup S)$ là không nhiều hơn $d(m - r + 1)/(r - d)$.

Như vậy, chúng ta sẽ chứng minh hai tập tối ưu được xác định bởi tập con của các ràng buộc này.

Chứng minh:

Gọi C_H chứa một tập tối ưu $\{O(T \cup S)|T \subseteq H\backslash S\}$. Thật vậy, gọi **SampLP**$(R \cup S)$ là phần tử của tập này được gọi lại. Tương tự, chúng ta xác định C_R là một tập tối ưu $\{O(T \cup S)|T \subseteq R\}$ cho một tập con riêng biệt. Bây giờ, $O(R \cup S)$ là phần tử duy nhất trong C_R thỏa mãn với mọi ràng buộc trong R. Mỗi phần tử $x \in C_H$, gọi v_x chứa số các vi phạm ràng buộc h bởi x. Gọi

$$i_x = \begin{cases} 1 \ \forall \ x \in O(R \cup S) \\ 0 \text{ trong trường hợp còn lại} \end{cases}$$

Ta có thể viết lại:

$$E[|V|] = E[\sum_{x \in C_H} v_x i_x] = \sum_{x \in C_H} v_x E[i_x] \quad (2.3)$$

Như vậy, $E[i_x]$ đơn giản chỉ là một xác suất, x là tối ưu thuộc $O(R \cup S)$. Với mỗi lần xuất hiện này, ràng buộc d phải nằm trong R, và còn lại ràng buộc $r - d$ của R phải nằm trong ràng buộc $m - v_x - d$ của $H\backslash S$ hoặc không xác định mà cũng không vi phạm bởi x.

Thật vậy,

$$E[i_x] = \frac{\binom{m - v_x - d}{r - d}}{\binom{m}{r}} \quad (2.4)$$

Từ (2.3) và (2.4) chỉ ra rằng:

$$E[|V_x|] \le \frac{m - r + 1}{r - d} \sum_{x \in C_H} v_x \frac{\binom{m - v_x - d}{r - d - 1}}{\binom{m}{r}} \quad (2.5)$$

Cuối cùng để hoàn thành việc chứng minh bằng việc chỉ ra tổng bên vế phải theo công thức 2.5 là không nhiều hơn d. Một mặt, $\binom{m - v_x - d}{r - d - 1} / \binom{m}{r}$ là xác suất, x là một phần tử của C_x và là một trong những vi phạm ràng buộc của R. Trọng số này bằng v_x và số phần tử của C_R cũng là một trong những ràng buộc của R. Tuy nhiên, số phần tử nhiều nhất là d, khi mỗi phần tử của tập $R \cup S\backslash\{h\}$ là tối ưu cho ràng buộc h để xác định $O(R \cup S)$ tối ưu, chính là xác định ràng buộc d trong tập tối ưu $O(R \cup S)$.

Như vậy, số vi phạm ràng buộc của bất đẳng thức *Makov* kéo theo nhiều mẫu ngẫu nhiên trong **SampLP**, **Pr**$[|V| > 2\sqrt{n}] \le ½$. Nó kéo theo sự tác động qua lại ở bước 2.2 giữa sự tăng S nhiều nhất là 2. Gọi $T(n)$ là thời gian chạy *maximum* của **SampLP**. Tập S được khởi tạo rỗng, với mỗi pha d thêm vào ít nhất $2\sqrt{n}$ ràng buộc. Thật vậy, $|R \cup S|$ không bao giờ vượt hơn $3d\sqrt{n}$ cho mỗi d pha, chúng ta thực hiện nhiều nhất n phép thử vi phạm ràng buộc và được đánh giá là $O(d)$ cho mỗi phép thử; tổng công việc kiểm tra ràng buộc là $O(d^2n)$. Trong sự tương tác với nhau đã làm số ràng buộc giảm xuống đến $9d^2$ hoặc nhỏ hơn. Chúng ta sắp xếp lại thời gian gọi **Simplex**, đặt chúng cùng nhau để quan sát, chúng ta có:

$$T(n) \le 2dT(3d\sqrt{n}) + O(d^2n), \text{ với } n > 9d^2 \quad (2.6)$$

2.4. Thuật toán InterSampLP

Chúng ta mô tả thuật toán **InterSampLP** bằng cách khám phá $B(H)$ dần dần, dùng kỹ thuật *interative reweighting* làm gia tăng xác suất của việc bao hàm lợi ích ràng buộc trong mẫu. Chúng ta chọn một tập con ngẫu nhiên của ràng buộc R và một tập con xác định $V \subset H$ vi phạm ràng buộc bằng sự tối ưu của bài toán quy hoạch tuyến tính xác định bởi R. Để thay cho việc thêm V vào tập S như trong **SampLP**, chúng ta đặt ràng buộc V vào sau khi tăng H, xác suất của chúng được chọn là một vòng tròn. Bằng trực giác ta thấy, ràng buộc $B(H)$ sẽ lặp lại tìm chúng trong V và kể từ đây xác suất của chúng trong R tăng nhanh hơn. Sau đó tính lặp lại dường như là tương đối, tất cả ràng buộc của $B(H)$ có lẽ đúng trong R và chúng sẽ kết thúc. Chúng ta mô tả chi tiết **InterSampLP** bên dưới, kết hợp một đại lượng tích phân dương *weight* w_h với mỗi ràng buộc $h \in H$, ràng buộc h sẽ được đặt vào R với tỷ lệ xác suất giá trị hiện tại của w_h.

Trong bước 2.2 xác suất một ràng buộc h được chọn là tỷ lệ với w_h. Chúng ta quay lại phân tích **InterSampLP**.

Việc gọi vòng lặp **while** thực hiện thành công nếu: $\sum_{h \in V} w_h \le (2\sum_{h \in H} w_h)(9d - 1)$

(thật vậy, w_h tăng gấp đôi với mỗi $h \in H$)

2.4.1. Mô tả thuật toán InterSampLP

Algorithm InterSampLP
Input: Một tập ràng buộc H
Output: B(H) tối ưu
1. $\forall h \in H$, *tập* $w_h \leftarrow 1$;
2. **if** $n < 9d^2$
   return Simplex (H) **else**
   2.1 $V \leftarrow H$;
   2.2 **While** $|V| > 0$
   Chọn ngẫu nhiên $R \subset H \setminus S$, với $|R| = r = 9d^2$ };
   $x \leftarrow$ **Samplex** (R);
   $V \leftarrow \{h \in H \mid x$ vi phạm $h\}$;

if $\sum_{h\in V} w_h \le (2\sum_{h\in H} w_h)(9d-1)$
then $\forall h \in V$,*tập* $w_h \leftarrow 2w_h$;
2.3 **reurn** x;

Hệ quả 3:

Số lần lặp của vòng lặp **While** *giữa các lần lặp thành công lớn nhất là 2.*

Ghi chú: Chúng ta không thể chỉ ra ngay kết quả của hệ quả 3 cho việc phân tích **InterSampLP**, ngay khi những ràng buộc trong tập con R xác suất ngẫu nhiên đã được chọn.

Định lý 1:

Tồn tại các ràng buộc c_1, c_2 và c_3 với kỳ vọng thời gian chạy của **InterSampLP** *là lớn nhất.* $c_1d^2n \log n + (c_2 d \log n) / d^{d/2+c_3}$

Chứng minh:

Chúng ta sẽ chứng tỏ rằng số lần thực hiện của vòng lặp **While** là O($d$log$n$). Cho rằng $\sum_{h\in B(H)} w_h$ tăng lên nhiều hơn $\sum_{h\in H} w_h$ vì sau $d$log$n$ lần lặp $V \neq \varphi$, trừ phi $\sum_{h\in B(H)} w_h > \sum_{h\in H} w_h$, nhưng điều này có lẽ là mâu thuẫn.

Sau mỗi lần thực hiện vòng lặp thành công, *wheight* w_h tăng lên gấp đôi ít nhất một ràng buộc $h \in$ B(H). Tiếp theo kd, thực hiện thành công vòng lặp, chúng ta có $\sum_{h\in B(H)} w_h = \sum_{h\in B(H)} 2^{n_h}$, với n_h là số thời gian của h thêm vào V.

Rõ ràng

$$\sum_{h\in B(H)} n_h \ge kd \Rightarrow \sum_{h\in B(H)} w_h \ge d2^k \quad (2.7)$$

Mặt khác, sau mỗi lần thực hiện thành công vòng lặp **while** tăng trong $\sum_{h\in H} w_h$ là không nhiều hơn $(2\sum_{h\in H} w_h)(9d-1)$. Giá trị ban đầu $\sum_{h\in H} w_h = n$. Tiếp theo, việc lặp kd thành công không nhiều hơn:

$$n[1 + 2/(9d-1)]^{kd} \le n \exp[2kd/(9d-1)] \quad (2.8)$$

2.4.2 Phân tích và đánh giá thuật toán

So sánh (2.7) và (2.8), chúng ta thấy sau O($d$log$n$) lần lặp chúng sẽ kết thúc vòng lặp. Câu hỏi đặt ra là: mất bao nhiêu thời gian giữa các lần lặp thành công của vòng lặp **while**? Bằng hệ quả 3, số lần lặp giữa các lần lặp thành công là 2. Ngay mỗi lần lặp, chúng ta đánh giá lời gọi **Simplex** và xác định V trong thời gian O(nd). Như vậy, các đánh giá trên đã mang lại điều phải chứng minh cho định lý 1.

2.5. Sự gia lượng trong quy hoạch tuyến tính

2.5.1. Ứng dụng gia lượng trong quy hoạch tuyến tính

Chúng ta sẽ nghiên cứu thuật toán quy hoạch tuyến tính dựa trên mẫu ngẫu nhiên. Chúng ta khảo sát thuật toán gia lượng ngẫu nhiên cho quy hoạch tuyến tính. Dùng một thuật toán gọi một cách trực tiếp chính nó: thêm n ràng buộc vào trong trật tự ngẫu nhiên, sau một thời gian. Với mỗi ràng buộc được thêm vào, xác định tối ưu việc thêm vào ràng buộc. Đây là thuật toán có lẽ cũng được xem như phương pháp "*quay lui*" (*backward*).

2.5.2. Mô tả thuật toán SeideLP

Algorithm SeideLP
Input: Một tập ràng buộc H
Output: **LP** tối ưu được xác định bởi H
1. **if** $|H| = d$, output B(H) = H;
2. Chọn ngẫu nhiên một ràng buộc $h \in H$;
3. Tìm lại B($H\setminus\{h\}$) một cách đệ qui.
   3.1 **if** B($H\setminus\{h\}$) là không vi phạm h,
   output B($H\setminus\{h\}$) được tối ưu B(H)
   3.2 **else** tất cả ràng buộc của $H\setminus\{h\}$ vào h và giải quyết bài toán mới này một cách đệ qui.

2.5.3. Phân tích và đánh giá thuật toán

Với mỗi h (chọn ngẫu nhiên những ràng buộc trong bước 2) là dư thừa (trong mỗi trường hợp chúng ta thực hiện trong bước 3.1), hoặc có thể không. Trong các trường hợp sau, chúng ta biết rằng một đỉnh được tạo bởi B(H) phải nằm trong đường biên của mặt siêu phẳng h. Trong trường hợp này, tất cả những ràng buộc của $H\setminus\{h\}$ nằm lên trên h và chúng ta giải quyết bài toán mới này với $d-1$ chiều. Khi số các ràng buộc giảm xuống d thì **SeideLP** ngừng lặp lại.

Khi d đạt nhiều nhất ràng buộc cực trị trong H, xác suất để chọn ngẫu nhiên ràng buộc h là một trong số ràng buộc cực trị, nhiều nhất là d/n.

Gọi $T(n, d)$ chứa đường biên bên trên, kỳ vọng

thời gian chạy của thuật toán cho bất kỳ bài toán với n ràng buộc trong d chiều. Chúng ta có thể viết:

$$T(n, d) \le T(n-1, d) + O(d) + \frac{d}{n}[O(dn) + T(n-1, d-1)] \quad (2.9)$$

Trong 2.9, số hạng đầu tiên bên phải chứa giá trị của vòng lặp được xác định bởi ràng buộc trong $H\backslash\{h\}$. Số hạng thứ hai là giá trị của việc kiểm tra xem h có vi phạm $B(H\backslash\{h\})$ hay không với xác suất d/n, và sự thu nạp này là biểu thức trong dấu ngoặc vuông, biểu thức này đếm giá trị số hạng đầu tiên của tất cả các ràng buộc trên h. Đếm giá trị thứ hai của việc giải quyết bài toán, nơi mà có khá nhiều ràng buộc và chiều. Định lý bên dưới chứng minh bằng phương pháp quy nạp.

Định lý 2:

Cho hai hằng số a và b như trong phép toán 2.9 thỏa mãn cách giải quyết $T(n,d) \le bnd$

Thuật toán gia lượng ở trên luôn đúng nhưng chậm trừ phi d là nhỏ. Chúng ta sẽ tự hỏi tại sao, khi giải quyết bài toán với $d-1$ chiều trong bước 3.2, chúng ta đã loại bỏ hoàn toàn bất kỳ thông tin từ cách giải quyết của bài toán tuyến tính $H\backslash\{h\}$ ở bước 1.

Bây giờ chúng ta xem lại tất cả các ràng buộc H trong bước 3.2 của **SeideLP**. Tuy nhiên, nó vẫn còn hợp lý để hy vọng rằng $B(H\backslash h)$ sẽ nằm trong mặt chứa các ràng buộc trong $B(H)$. Chúng ta có thể chỉ ra một cách sử dụng khác để $B(H)$ " *bắt đầu – rẽ nhánh*" lặp lại việc gọi trong bước 3.2 của **SeideLP**. Kết quả của ý tưởng này là thuật toán **BasisLP**. Chỉ ra hai lập luận, một tập $G \subseteq H$ của các ràng buộc và một tập cơ sở $T \subseteq G$ (tập cơ sở không đầy đủ của G). **BasisLP** quay lại tập cơ sở của G.

2.6. Thuật toán BasisLP

2.6.1. Mô tả thuật toán BasisLP

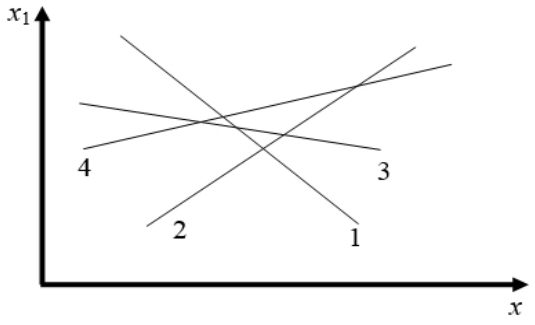
Algorithm BasisLP
Input: G, T
Output: Một tập cơ sở B cho G
1. **If** $G = T$, output T;
2. Chọn một ràng buộc ngẫu nhiên $h \in G\backslash T$;
   $T' =$ **BasisLP**$(G\backslash\{h\}, T)$;
   2.1. **If** h không vi phạm T', output T';
   2.2. **Else** output **BasisLP**(G, **Basis**$(T' \cup \{h\})$)

2.6.2. Phân tích và đánh giá thuật toán

Hàm **Basis** quay lại một tập cơ sở cho một tập của $d+1$ hoặc một số khá nhiều các ràng buộc nếu ngay khi tồn tại một tập cơ sở. Trong thuật toán này, chúng ta luôn vi phạm hàm **Basis** cho tập cơ sở T' với các ràng buộc d, cùng với một ràng buộc h mới. Bằng việc tìm giao của h với mỗi tập con d của T' có lực lượng $d-1$ và đánh giá là O tại mỗi điểm d đó. Vì vậy, chúng ta xác định **Basis**$(T' \cup \{h\})$.

Với mỗi lần gọi hàm cơ sở là đã được kiểm tra vi phạm trước (trong câu lệnh **if**). Chúng ta sẽ kiểm tra cận vi phạm, và từ suy luận này ta có cận của hàm cơ sở. Thật vậy, đây là tổng thời gian chạy. Như vậy, *kiểm tra xác suất vi phạm không đạt được cho trong quá trình thực hiện **BasisLP** là gì?* Giả sử $|G| = i$. Chúng ta gọi lại ràng buộc $h \in G\backslash T$ đã được chọn ngẫu nhiên, và hy vọng xác suất cận vi phạm h là tối ưu của $G\backslash\{h\}$. Rõ ràng, vi phạm này nhiều nhất là $d/(i-|T|)$, khi ràng buộc nhiều nhất là d của G, xác định $B(G)$ và h là gần bằng với bất kỳ ràng buộc $i-|T|$ trong $G\backslash T$. Bây giờ chúng ta sẽ làm rõ hơn xác suất ước lượng này. Bằng trực giác, xác suất này sẽ giảm hơn nữa nếu T chứa một số ràng buộc của $B(G)$.

Cho $T \subseteq G \subseteq H$, chúng ta gọi ràng buộc $h \subseteq G$ cưỡng bức trong (G, T) nếu $O(G\backslash\{h\}) < O(T)$. Điều này được thể hiện trong hình bên dưới. Trong hình này, có bốn ràng buộc, được đánh số *1*, *2*, *3* và *4*. Với mỗi ràng buộc là một đường xác định *nửa – mặt phẳng* trên chính nó như một *vùng có thể thực hiện*. Rõ ràng, ràng buộc *1* và *4* là các ràng buộc cực trị cho tập $\{1, 2, 3, 4\}$. Xét sự phân tích ngược của **BasisLP** và tình huống này các ràng buộc đã được thêm vào sau trong trật tự *1*, *2*, *3*, *4*. Quan sát ràng buộc cưỡng bức *1* trong G, T cho $G = \{1, 2, 3, 4\}$ và $T = \{1, 2\}$.



Mô hình các ràng buộc cưỡng bức.

Nếu tất cả ràng buộc cưỡng bức d của T là nằm trong (G, T). Ta có $T = B(G)$. Cho $T \subseteq G \subseteq H$, gọi $\Delta_{G,T}$ chứa d là một số âm của các ràng buộc cưỡng bức trong (G, T). Gọi $\Delta_{G,T}$ là kích thước ẩn số của (G, T). Số ràng buộc của $B(G)$ là không sẵn sàng trong T. Từ thảo luận trên, xác suất để một vi phạm xảy ra trong câu lệnh **if** có cận là $\Delta_{G,T}/(i-|T|)$. Trước tiên, chúng ta cho kích cỡ ẩn giảm

nhỏ nhất là 1 tại mỗi lần gọi lặp lại ở bước 2.2; sau đó, chúng ta sẽ hoàn thiện bằng lập luận rằng nó dường như giảm nhiều hơn.

Thật vậy, khi chúng ta tiếp tục sự lặp lại bên dưới (trong một dãy quá trình thực hiện ở bước 2.2), tử số của cận xác suất giảm nhỏ nhất là 1 tại mỗi lần thực hiện. Bây giờ, chúng ta chỉ ra rằng việc giảm kích cỡ ẩn (xác suất giảm) dường như nhanh hơn. Cho tập F và T với $T \subset F \subseteq G$, và $h \in F\backslash T$ ngẫu nhiên, cận xác suất để bổ sung h vào $F\backslash\{h\}$ lý do lặp lại việc gọi. Khi đó, hàm phân phối xác suất của kích cỡ ẩn trong lập luận này như một lời gọi.

Khi $h=g$, với $g=\{g_1,g_2,\ldots g_l\}$ sẽ là cưỡng bức trong (F, **Basis** ($B(F\backslash\{h\}) \cup \{h\}$)). Khi đó, lặp luận của việc gọi lặp lại sẽ có kích cỡ ẩn là $\Delta_{G,T} - l$. Quan sát chủ yếu là ngay khi bất kỳ g_i là gần bằng h, l là hàm phân phối đồng dạng trên các số nguyên trong $[1, s]$. Thật vậy, kích cỡ ẩn của việc gọi lặp lại là hàm phân phối đồng dạng trên các số nguyên trong $[0, s-1]$.

Cho lời gọi **BasisLP** với lặp luận (G, T), - $|G| = m$ và $\Delta_{G,T} - k$ gọi $T(m, k)$ là kỳ vọng lớn nhất của kiểm tra vi phạm (việc thực hiện câu lệnh **if**). Như vậy, thời gian chạy của **BasisLP** trong bài toán với n ràng buộc trong d chiều là $\mathbf{O(d^4\, 2^d n)}$.

3. Kết luận

Quy hoạch tuyến tính và phương pháp đơn hình có tầm quan trọng căn bản vì một lớp rất rộng các bài toán giải được bằng phương pháp này. Với một số bài toán đặc biệt, chúng ta còn có các thuật toán tốt hơn, tuy nhiên chỉ một số ít kỹ thuật có thể áp dụng rộng rãi như kỹ thuật quy hoạch tuyến tính. Những nghiên cứu về quy hoạch tuyến tính rất mạnh mẽ, vì vậy muốn có được sự hiểu biết đầy đủ tất cả các vấn đề đòi hỏi nhiều kiến thức rất phức tạp. Trong khuôn khổ bài báo này, chúng tôi dừng lại ở việc phân tích, đánh giá thuật toán quy hoạch tuyến tính dựa trên gia lượng ngẫu nhiên. Bài báo nhằm chỉ ra được những ưu điểm của thuật toán gia lượng ngẫu nhiên so với những thuật toán tất định. Từ đó, chúng ta có thể lựa chọn thuật toán để áp dụng cho phù hợp.

Tài liệu tham khảo

Aho, A.V., JE, Hopcroft và Ullman, J.D. 1990. *The Design and Analysis of Computer Algorithms*. Addison-Wesley. Reading.

Aldous, D.J. *1994. Reversible Markov chains and random walks on graphs*. Unpublished Monograph. Bekeley.

Aleliunas, R. 1982. *Randomized parallel communication.* In ACM-SIGOPS Symposium on Principles of Distributed System.

Bollobas, B. 1987. *Random Graphs*. Acedemic Press: New York.

Cormen,T. & Leiserson, C.E. 1990. *Introduction to Algorithms*: New York.

Dantzig. G.B. 2003. *Leanear Programing and Extensions.* Princeton University Press.

Diestel, R. 2000. *Graph Theory*. Springer-Verlag New York.

Đinh, Mạnh Tường. 2002. *Cấu trúc dữ liệu và thuật toán*. NXB Đại học Quốc Gia Hà Nội.

Motwani, R. 2000. *Randomized Algorithms.*Stanford University.

Nguyễn, Anh Tuấn. 2000. *Quy hoạch gần lồi - gần lõm ứng dụng vào quy hoạch tuyến tính*. NXB Khoa học Kỹ thuật.

Nguyễn, Hải Thanh. 2006. *Tối ưu hóa*. NXB Bách Khoa Hà Nội.

Nguyễn, Hữu Điền. 2005. *Một số vấn đề về thuật toán*. NXB Giáo dục.

Phan, Quốc Khánh & Trần, Huệ Nương. 2000. *Quy hoạch tuyến tính*. NXB Giáo dục.

Sedgewick, R. 2004. *Cẩm nang thuật toán.* NXB Khoa học và Kỹ thuật.