فصل ۴:
برنامه‌ریزی خطی

بخش دوم:

الگوریتم افزایشی

برنامه‌ریزی خطی تصادفی

مسائل برنامه‌ریزی خطی بی‌کران

تعاریف مورد نیاز:

مجموعه n محدودیت خطی در مسئله دو متغیره را به شکل زیر نشان می‌دهیم:

$$H = \{ h_1, h_2, \dots, h_n \}$$

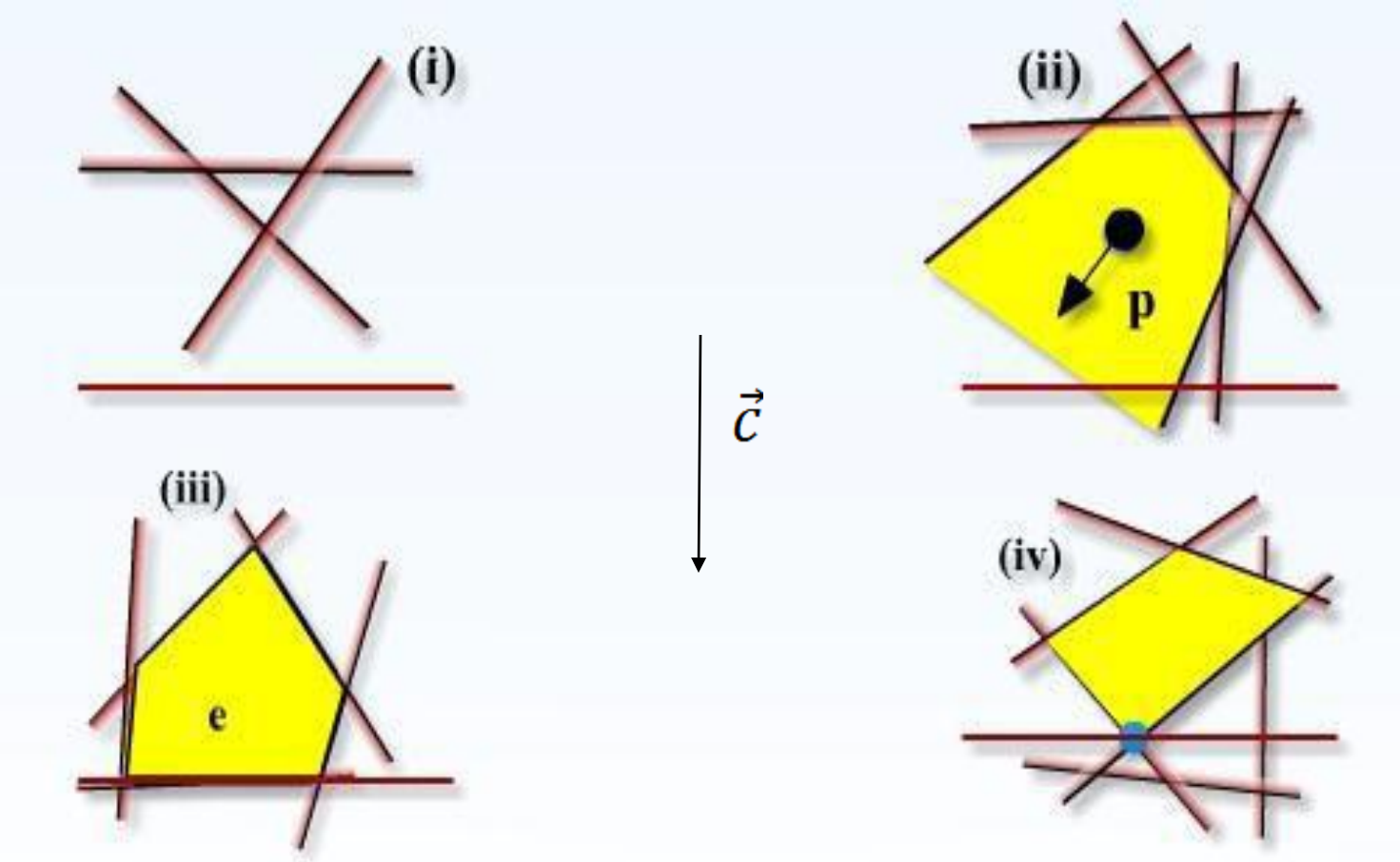
بردار $\vec{c} = (c_x, c_y)$ تابع هدف را تعریف می‌کند:

$$f_{\vec{c}}(p) = c_x p_x + c_y p_y$$

هدف: پیدا کردن نقطه‌ای مثل $p \in R^2$ است $p \in \cap H$ و $f_{\vec{c}}(p)$ ماکزیمم شود.

مسئله خطی را با $(H, \vec{c})$ و ناحیه ممکن یا feasible را با C نشان می‌دهیم.

چها حالت جواب مسئله:



چهار حالت جواب مسئله:

1) مسئله infeasible باشد که در این صورت هیچ جوابی برای مجموعه محدودیتها موجود نیست.
2) ناحیه feasible در جهت بردار $\vec{c}$ بی‌کران باشد که در اینصورت می‌توان پرتوی مانند ρ یافت که کاملا در ناحیه امکان باشد و تابع $f_{\vec{c}}$ مقادیر بزرگ دلخواه در طول آن اختیار می‌کند.
3) ناحیه امکان یک یال e داشته باشد به طوری که بردار نرمال خارجی آن به سمت بردار $\vec{c}$ باشد در اینصورت مسئله جواب یکتا ندارد و جواب آن تمامی نقاط روی e است.
4) یک جواب یکتا وجود دارد که در جهت $\vec{c}$ تابع هدف را ماکزیمم می‌کند.

الگوریتم افزایشی:

ایده اصلی:

محدودیتهای مسئله را یک به یک اضافه می‌کنیم و هر بار جواب بهینه را برای هر کدام از مسائل میانی بدست می‌آوریم در نهایت در مرحله آخر جواب کلی مسئله بدست خواهد آمد.

نیازهای الگوریتم:

الگوریتم نیاز دارد که در هر مرحله میانی جواب، خوش تعریف و یکتا باشد.

فرض می‌کنیم که در هر مرحله میانی ناحیه **feasible** همانند حالت **(iv)** دارای یک جواب بهینه یکتا است

برآوردن پیش شرطهای الگوریتم:

برای اطمینان از این که مسئله کراندار است دو محدودیت جدید به مسئله اضافه خواهیم کرد.

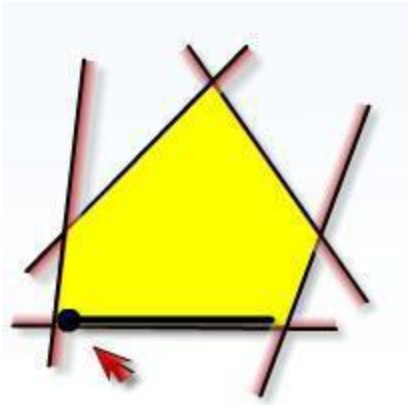
$$m_1 := \begin{cases} p_x \leqslant M & \text{if } c_x > 0 \\ -p_x \leqslant M & \text{otherwise} \end{cases}$$

$$m_2 := \begin{cases} p_y \leqslant M & \text{if } c_y > 0 \\ -p_y \leqslant M & \text{otherwise} \end{cases}$$

۱) M را باید چنان بزرگ انتخاب کنیم که در صورتی که مسئله اصلی کراندار بود تاثیری روی جواب بهینه نداشته باشد.

۲) m_1 و m_2 به عنوان توابعی از c هستند و به نیم صفحه‌های H بستگی ندارند

می توان یک ایده ساده به کار برد تا مسئله در حالت (iii) نیز یک جواب بهینه یکتا داشته باشد:



اگر چند جواب بهینه داشته باشیم کوچکترین جواب از لحاظ ترتیب لغتنامه‌ای را به عنوان جواب بهینه برمی‌گزینیم.

راس بهینه :

حال با استفاده از این دو قرارداد می توان گفت که هر مسئله برنامه‌ریزی خطی که دارای ناحیه feasible باشد دارای یک جواب یکتاست که راسی از ناحیه feasible است و ما آن را راس بهینه می‌نامیم.

چند تعریف مورد نیاز:

فرض کنید (H, c) یک مسئله برنامه‌ریزی خطی باشد. نیم صفحه‌ها را با $h_1, h_2, \dots, h_n$ نشان می‌دهیم داریم:

$$H_i = \{m_1, m_2, h_1, h_2, \dots, h_i\}$$

$$C_i = m_1 \cap m_2 \cap h_1 \cap h_2 \cap \dots \cap h_i$$

جواب یکتای مرحله i را با v_i نشان می‌دهیم و داریم:

$$C_0 \supseteq C_1 \supseteq C_2 \supseteq \cdots \supseteq C_n = C$$

در نتیجه اگر برای یک i داشته باشیم $C_i = \emptyset$ آنگاه برای هر j>=i داریم $C_j = \emptyset$

چگونگی تغییر راس بهینه در هر مرحله الگوریتم:

لم ۴.۵:

فرض کنید $1 \leq i \leq n$ و C_i و v_i مشابه بالا تعریف شده باشند، داریم:

i. اگر $v_{i-1} \in h_i$ آنگاه $v_i = v_{i-1}$

ii. اگر $v_{i-1} \notin h_i$ آنگاه یا $C_i = \emptyset$ و یا $v_i \in l_i$ که l_i خط کران h_i است.

اثبات لم ۴.۵:

i. فرض کنید $v_{i-1} \in h_i$:

$$\begin{cases} C_i = C_{i-1} \cap h_i \\ v_{i-1} \in C_{i-1} \end{cases} \Rightarrow v_{i-1} \in C_i$$

از طرف دیگر نقطه بهینه در C_i نمی‌تواند بهتر از نقطه بهینه در C_{i-1} باشد چون $C_i \subseteq C_{i-1}$.

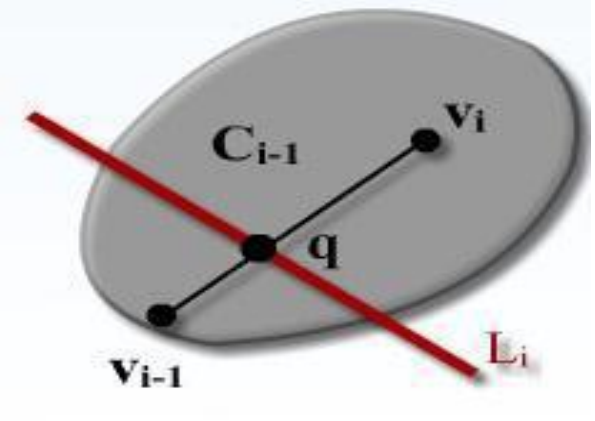
بنابراین v_{i-1} یک نقطه بهینه در C_i نیز هست.

ادامه اثبات:

ii. فرض کنید $v_{i-1} \notin h_i$

فرض خلف: $C_i \neq \emptyset$ و $v_i \notin l_i$

پاره خط $\overline{v_{i-1}v_i}$ را در نظر بگیرید می‌دانیم $v_{i-1} \in C_{i-1}$ و چون $C_i \subseteq C_{i-1}$ نتیجه می شود $v_i \in C_{i-1}$. از آنجایی که C_{i-1} محدب است پس پاره‌خط $\overline{v_{i-1}v_i}$ در ناحیه C_{i-1} قرار دارد.



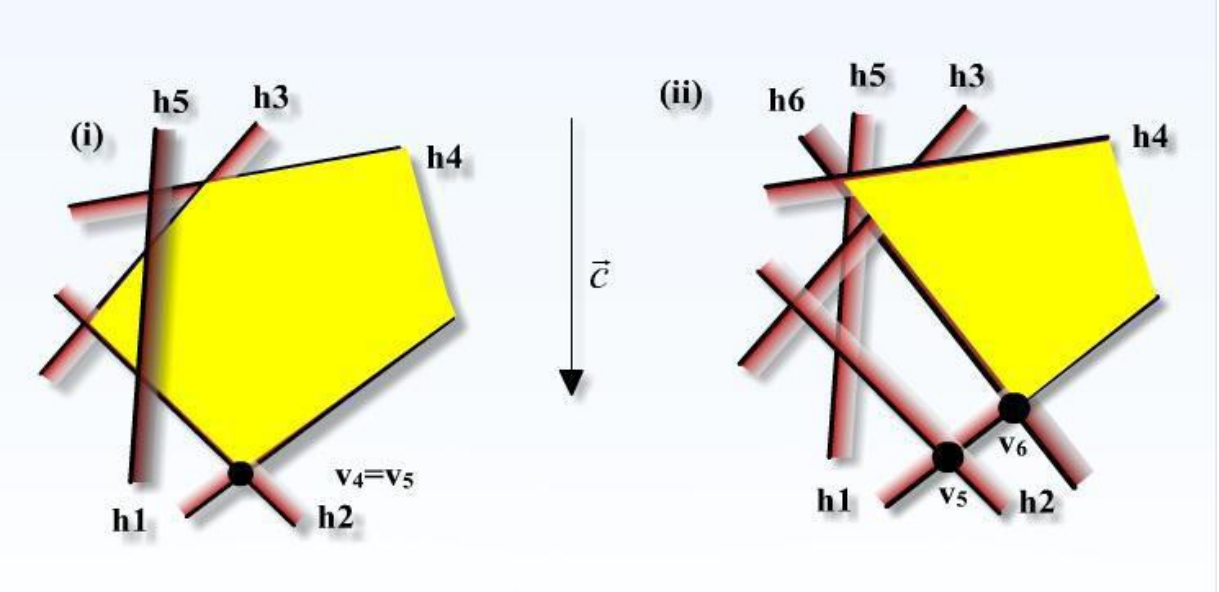
ادامه اثبات:

چون v_{i-1} راس بهینه C_{i-1} است و تابع هدف $f_{\vec{c}}$ خطی است بنابراین $f_{\vec{c}}(p)$ با حرکت p از v_i به v_{i-1} در طول پاره‌خط $\overline{v_{i-1}v_i}$ به طور یکنواخت افزایش می‌یابد.

حال نقطه q که محل برخورد $\overline{v_{i-1}v_i}$ و l_i است را در نظر می‌گیریم این نقطه وجود دارد چون $v_{i-1} \notin h_i$ و $v_i \in C_i$ از آنجایی که $\overline{v_{i-1}v_i}$ در C_{i-1} است نقطه q باید در C_i باشد.

اما مقدار تابع هدف در طول پاره‌خط $\overline{v_{i-1}v_i}$ افزایش می‌یابد که یعنی $f_{\vec{c}}(p) > f_{\vec{c}}(v_i)$ که این با تعریف نقطه v_i در تناقض است.

دو حالتی که پس از اضافه کردن نیم‌صفحه جدید رخ می‌دهند:



پیدا کردن راس بهینه در حالت دوم لم ۴.۵:

هدف:

پیدا کردن نقطه p روی خط l_i است به نحوی که $f_{\vec{c}}(p)$ ماکزیمم شود و $p \in h$ برای $h \in H_{i-1}$

پیش فرضها:

برای سادگی مسئله فرض می کنیم l_i عمودی نباشد بنابراین می‌توانیم آن را با یک مختص x نشان دهیم و تابع $\overline{f_{\vec{c}}}: R \rightarrow R$ را طوری تعریف می‌کنیم که برای نقاط $p \in l_i$ داشته باشیم:

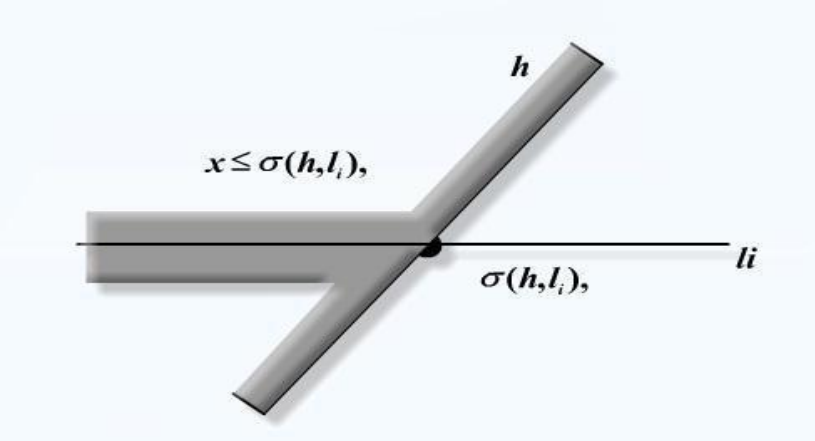
$$f_{\vec{c}}(p) = \overline{f_{\vec{c}}}(p_x)$$

پیدا کردن راس بهینه در حالت دوم لم ۴.۵:

تعریف:

فرض کنید $\sigma(h, l_i)$ مختص x محل برخورد خط l_i و نیم‌صفحه h است.

با توجه به این که $l_i \cap h$ از سمت راست یا چپ کراندار باشد به یک محدودیت روی مختص x جواب خواهیم رسید.



پیدا کردن راس بهینه در حالت دوم لم ۴.۵:

بنابراین مسئله به فرم یک متغیره زیر درمی‌آید:

$$maximize \quad \overrightarrow{f_{c}}(x)$$

$$subject\ to \quad x \geq \sigma(h, l_i) \qquad h \in H_{i-1} \qquad l_i \cap h \text{ از چپ کراندار است}$$

$$x \leq \sigma(h, l_i) \qquad h \in H_{i-1} \qquad l_i \cap h \text{ از راست کراندار است}$$

این یک مسئله برنامه‌ریزی خطی یک بعدی است. قرار می‌دهیم:

$$x_{left} = max_{h \in H_{i-1}} \{\sigma(h, l_i) : l_i \cap h \ is \ bounded \ to \ the \ left\}$$

$$x_{right} = min_{h \in H_{i-1}} \{\sigma(h, l_i) : l_i \cap h \ is \ bounded \ to \ the \ right\}$$

پیدا کردن راس بهینه در حالت دوم لم ۴.۵:

بازه $[x_{left}: x_{right}]$ ناحیه feasible مسئله یک بعدی است مسئله infeasible است اگر $x_{left} > x_{right}$ وگرنه جواب بهینه با توجه به تابع هدف یکی از نقاط x_{left} یا x_{right} است.

لم ۴.۶:

مسئله برنامه‌ریزی خطی یک بعدی را می‌توان در زمان خطی حل کرد. بنابراین اگر حالت (ii) لم ۴.۵ اتفاق بیفتد، می‌توان در زمان O(i) راس بهینه مرحله v_i را یافت یا تصمیم گرفت که مسئله infeasible است.

الگوریتم افزایشی

Algorithm 2DBOUNDEDLP($H, \vec{c}, m_1, m_2$)
Input. A linear program $(H \cup \{m_1, m_2\}, \vec{c})$, where H is a set of n half-planes, $\vec{c} \in \mathbb{R}^2$, and m_1, m_2 bound the solution.
Output. If $(H \cup \{m_1, m_2\}, \vec{c})$ is infeasible, then this fact is reported. Otherwise, the lexicographically smallest point p that maximizes $f_{\vec{c}}(p)$ is reported.

1. Let v_0 be the corner of C_0.
2. Let $h_1, \ldots, h_n$ be the half-planes of H.
3. **for** $i \leftarrow 1$ **to** n
4. **do if** $v_{i-1} \in h_i$
5. **then** $v_i \leftarrow v_{i-1}$
6. **else** $v_i \leftarrow$the point p on ℓ_i that maximizes $f_{\vec{c}}(p)$, subject to the constraints in H_{i-1}.
7. **if** p does not exist
8. **then** Report that the linear program is infeasible and quit.
9. **return** v_n

لم ۴.۷:

الگوریتم 2DBOUNDEDLP جواب یک مسئله خطی کراندار با n محدودیت و دو متغیر را در زمان $O(n^2)$ و حافظه خطی محاسبه می‌کند.

اثبات:

برای اثبات درستی باید ثابت کنیم در هر مرحله راس بهینه به درستی محاسبه می‌شود، که این مطلب مستقیما از لم ۴.۵ نتیجه می‌شود اگر مسئله برنامه‌ریزی خطی یک بعدی که حل می‌کنیم infeasible باشد آنگاه C_i تهی است و چون داریم $C = C_n \subseteq C_i$ پس در کل C نیز تهی است که یعنی مسئله برنامه‌ریزی خطی infeasible است.

ادامه اثبات لم ۴.۷:

با توجه به این که نیم صفحه‌ها یکی یکی اضافه می‌شوند و زمانی که در مرحله iام صرف می‌شود برابر است با حل مسئله خطی یک متغیره که O(i) است زمان اجرای الگوریتم برابر است با:

$$\sum_{i=1}^{n} O(i) = O(n^2)$$

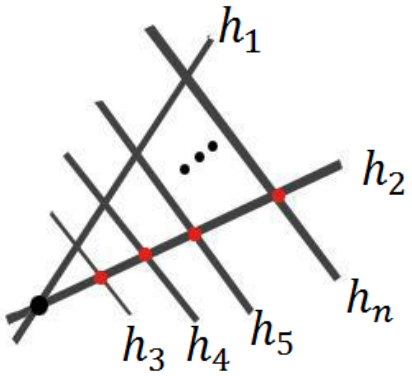
یک نکته در مورد تحلیل:

- در واقع ما هزینه هر مرحله i را توسط O(i) کراندار کردیم اما زمان مرحله iام همیشه به این سقف نمی‌رسد. وقتی $v_{i-1} \notin h_i$ مرحله iام به زمان $\theta(i)$ نیاز دارد در غیر اینصورت مرحله iام در زمان ثابت حل می‌شود.

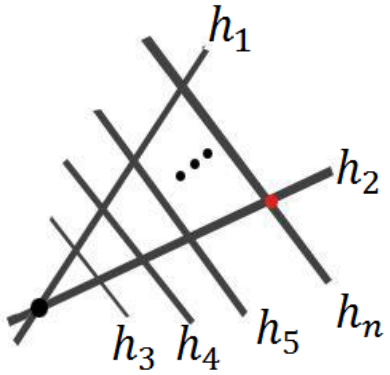
- بنابراین اگر بتوان تعداد تغییر راس در مراحل الگوریتم را محدود کرد می‌توان زمان اجرای بهتری بدست آورد.

برنامه‌ریزی خطی تصادفی

$h_1, h_2, \dots, h_n$



$h_1, h_2, h_n, \dots, h_3$



- پس برای کاهش زمان اجرای الگوریتم باید ترتیب نیم صفحه‌ها اصلاح شود.
- یافتن ترتیبی که گفتیم ساده نیست اما می‌توان از یک ترتیب تصادفی استفاده کرد.

الگوریتم تصادفی

Algorithm 2DRANDOMIZEDBOUNDEDLP(H, $\vec{c}$, m_1, m_2)
Input. A linear program $(H \cup \{m_1, m_2\}, \vec{c})$, where H is a set of n half-planes, $\vec{c} \in R_2$, and m_1, m_2 bound the solution.
Output. If $(H \cup \{m_1, m_2\}, \vec{c})$ is infeasible, then this fact is reported. Otherwise, the lexicographically smallest point p that maximizes $f_{\vec{c}}(p)$ is reported.

1. Let v_0 be the corner of C_0.
2. Compute a random permutation $h_1, \ldots, h_n$ of the half-planes by calling RANDOMPERMUTATION(H[1⋯n]).
3. **for** i←1 **to** n
4. **do if** $v_{i-1} \in h_i$
5. **then** $v_i \leftarrow v_{i-1}$
6. **else** $v_i \leftarrow$the point p on l_i that maximizes $f_{\vec{c}}(p)$, subject to the constraints in H_{i-1}.
7. **if** p does not exist
8. **then** Report that the linear program is infeasible and quit.
9. **return** v_n

ادامه الگوریتم تصادفی

- Random(k) یک تولید کننده عدد تصادفی است که عدد صحیح k را به عنوان ورودی می‌گیرد و یک عدد تصادفی بین 1 تا k تولید می‌کند.

Algorithm RANDOMPERMUTATION(A)
Input. An array A[1⋯n].
Output. The array A[1⋯n] with the same elements, but rearranged into a random permutation.

```
1.    for k←n downto 2
2.        do rndindex ←RANDOM(k)
3.            Exchange A[k] and A[rndindex].
```

زمان اجرای الگوریتم تصادفی

- زمان اجرای الگوریتم تصادفی به ترتیب انتخاب شده در خط ۲ بستگی دارد.

- از آنجایی که **n!** ترتیب مختلف وجود دارد **n!** اجرا برای الگوریتم وجود داردکه هر کدام زمان اجرای خاص خود را دارد و چون انتخاب هر حالت به روش تصادفی انجام می‌شود احتمال هر کدام از زمانهای اجرا برابر است.

- بنابراین برای تحلیل الگوریتم زمان اجرای مورد انتظار را بدست می‌آوریم که زمان اجرای میانگین روی **n!** جایگشت ممکن است.

لم ۴.۸:

مسئله برنامه‌ریزی خطی ۲-بعدی با n محدودیت می‌تواند در زمان موردانتظار O(n) و با حافظه خطی حل شود.

دو تعریف مورد نیاز:

- امید ریاضی: در نظریه احتمالات امید ریاضی، میانگین یا مقدار مورد انتظار یک متغیر تصادفی گسسته برابر است با مجموع حاصلضرب احتمال وقوع هر یک از حالات ممکن در مقدار آن حالت.

$$E(x) = \sum_{i=1}^{n} x_i \, p_X(x_i)$$

- متغیر تصادفی: متغیری است که مقدار آن از اندازه‌گیری برخی از انواع فرایندهای تصادفی بدست می‌آید به عبارت دیگر متغیر تصادفی توصیف عددی خروجی یک آزمایش است.

اثبات لم ۴.۸:

- همان طور که دیدیم حافظه مورد نیاز الگوریتم خطی است.
- زمان اجرای رویه RandomPermutation، O(n) است
- آنچه که باقی می‌ماند تحلیل زمان لازم برای اضافه کردن نیم‌صفحه‌های $h_1, h_2, \dots, h_n$ است.

۱) اگر راس بهینه تغییر نکند اضافه کردن نیم‌صفحه جدید زمان ثابت نیاز دارد.

۲) وقتی راس بهینه تغییر می‌کند لازم است که یک مسئله ۱-بعدی را حل کنیم.

- حال به تحلیل زمان مورد نیاز برای حل این مسئله می‌پردازیم.

ادامه اثبات لم ۴.۸:

متغیر تصادفی X_i را بصورت زیر تعریف می‌کنیم:

$$X_i = \begin{cases} 1 & v_{i-1} \notin h_i \\ 0 & o.w \end{cases}$$

مسئله یک بعدی با i محدودیت را می توان در زمان O(i) حل کرد.

زمان کلی لازم برای اضافه کردن n محدودیت در خط ۶ الگوریتم برابر است با:

$$\sum_{i=1}^{n} O(i) \, . X_i$$

ادامه اثبات لم ۴.۸:

حال برای بدست آوردن زمان مورد انتظار و با استفاده از مفهوم خطی بودن امید ریاضی داریم:

$$E[\sum_{i=1}^{n} O(i) \,.X_i] = \sum_{i=1}^{n} O(i).E[X_i]$$

که مقدار $E[X_i]$ برابر با احتمال این است که $v_{i-1} \notin h_i$

در واقع می‌خواهیم ببینیم با چه احتمالی در مرحله **i**ام راس بهینه تغییر می‌کند.

ادامه اثبات لم ۴.۸:

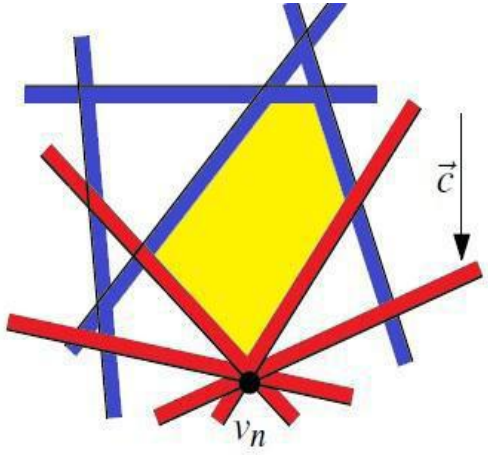
برای تحلیل $E[X_i]$ از روش Backwards analysis استفاده می‌کنیم یعنی فرض می‌کنیم که راس بهینه نهایی محاسبه شده است حال یک پله به عقب می‌رویم و به C_{n-1} می‌نگریم. C_{n-1} از C_n و با حذف نیم‌صفحه h_n بدست می‌آید در صورتی راس بهینه تغییر می‌کند که نیم صفحه h_n یکی از نیم صفحه‌هایی باشد که راس v_n را تعریف می‌کنند می‌دانیم که هر راس توسط حداقل دو نیم صفحه تعریف می‌شود ادعا می‌کنیم که احتمال تغییر راس بهینه حداکثر 2/n است.

ادامه اثبات لم ۴.۸:

چرا حداکثر 2/n ؟

۱) در صورتی که راس v_n توسط بیش از دو نیم‌صفحه تعریف شده باشد آنگاه حذف یکی از نیم‌صفحه‌ها باعث تغییر v_n نخواهد شد یعنی احتمال تغییر راس بهینه با حذف یک نیم‌صفحه صفر است.

۲) اگر راس v_n توسط دقیقا دو نیم‌صفحه تعریف شده باشد آنگاه احتمال اینکه هر کدام از آنها در ترتیب تصادفی انتخاب شده ما nامین راس باشد 1/n است.



۳) اگر v_n توسط m_1 و m_2 تعریف شده باشدچون این دو نیم‌صفحه انتخاب تصادفی نمی‌شوند احتمال برابر صفر است.

ادامه اثبات لم ۴.۸:

اگر همین روند را در مورد i نیم‌صفحه اول بکار ببریم به نتیجه مشابه می‌رسیم یعنی:

$$E(X_i) \leq 2/i$$

حال با جایگذاری $E[X_i]$ در فرمول اولیه خواهیم داشت:

$$\sum_{i=1}^{n} O(i).\frac{2}{i} = O(n)$$

مسائل برنامه‌ریزی خطی بی‌کران

در قسمت قبل برنامه‌های خطی نامحدود را با اعمال دو محدودیت اضافه کراندار کردیم اما همیشه نمی‌توان این کار را انجام داد چون حتی در مسائل کراندار نیز ممکن است Mای که به اندازه کافی بزرگ باشد نشناسیم از سوی دیگر مسائل بی‌کران در عمل رخ می‌دهند و باید بتوانیم آنها را حل کنیم.

تشخیص بی‌کران بودن یک مسئله:

گفتیم یک مسئله بی‌کران است اگر بتوانیم پرتویی مثل ρ یافت که کاملا در ناحیه feasible باشد و تابع هدف در جهت آن مقادیر بزرگ دلخواه اختیار کند.

اگر نقطه شروع پرتو را با p و جهت آن را با $\vec{d}$ نشان دهیم پرتو ρ به شکل زیر پارامتریزه می‌شود:

$$\rho = \{ p + \lambda \vec{d} : \lambda > 0 \}$$

تابع $f_{\vec{c}}$ مقادیر بزرگ دلخواه اختیار خواهد کرد اگر و تنها اگر $\vec{d} . \vec{c} > 0$ و از سوی دیگر داشته باشیم $\vec{d} . \vec{\eta}(h) \geq 0$

لم ۴.۹:

برنامه خطی $(H, \vec{c})$ بی‌کران است اگر و تنها اگر یک بردار $\vec{d}$ وجود داشته باشد به طوری که $\vec{d}.\vec{c} > 0$ و برای هر $h \in H$ داشته باشیم $\vec{d}.\vec{\eta}(h) \geq 0$ و برنامه خطی $(H', \vec{c})$، feasible باشد که:

$$H' = \{\, h \in H : \vec{\eta}(h).\vec{d} = 0 \,\}$$

اثبات لم ۴.۹:

طرف رفت: از بحثی که قبلا ارائه شد نتیجه خواهد شد فقط کافیست درستی طرف برگشت را اثبات کنیم.

طرف برگشت:

فرض: برنامه خطی $(H, \vec{c})$ و بردار $\vec{d}$ را با شرایط گفته شده در لم در نظر می‌گیریم و ثابت می‌کنیم که $(H, \vec{c})$ بی‌کران است.

می‌دانیم که $(H', \vec{c})$، feasible است نقطه‌ای مثل p_0 وجود دارد که $p_0 \in \bigcap_{h \in H'} h$ حال پرتو $\rho_0 = \{ p_0 + \lambda \vec{d} : \lambda > 0 \}$ را در نظر می‌گیریم.

چون برای هر $h \in H'$ داریم $\vec{d} . \vec{\eta}(h) = 0$ پرتو ρ_0 کاملا داخل هر کدام از نیم‌صفحه‌های $h \in H'$ قرار دارد.

ادامه اثبات لم ۴.۹:

از سوی دیگر از آنجایی که $\vec{d}.\vec{c} > 0$ تابع هدف مقادیر بزرگ دلخواه در جهت ρ_0 خواهد گرفت.

برای نیم‌صفحه‌های $h \in H \backslash H'$ داریم $\vec{d}.\vec{\eta}(h) > 0$ که یعنی یک مقدار λ_h وجود دارد به طوری که برای هر $\lambda \geq \lambda_h$ داریم:

$$p_0 + \lambda \vec{d} \in h$$

از بین این λ_h ها بزرگترین آنها را انتخاب می‌کنیم و آن را λ' می‌نامیم. با استفاده از این مقدار یک نقطه شروع برای پرتویی که درون تمام نیم‌صفحه‌های $h \in H$ می‌سازیم که این پرتو برابر است با

$$\rho = \{\, p + \lambda \vec{d} : \ \lambda > 0 \,\}$$

بنابراین مسئله $(H, \vec{c})$ بی‌کران است.

برنامه خطی غیر کراندار

برای تست بی‌کران بودن برنامه خطی سیستم مولفه را می چرخانیم، پس داریم :

$$\vec{c} = (0,1)$$
$$\vec{d}.\vec{c} > 0 \Rightarrow \vec{d} = (d_x, 1)$$
$$\vec{d}.\vec{\eta}(h) = d_x\eta_x(h) + \eta_y(h) \geq 0$$

پس برای بی‌کران بودن برنامه خطی باید n معادله خطی فوق دارای جواب باشد، که معادل است با حل برنامه خطی یک بعدی.

نکته: در برنامه خطی نیاز به تابع هدف است که در اینجا چون فقط روی وجود یا عدم وجود جواب بحث می‌شود، تابع هدف نادیده گرفته می‌شود.

مسئله جواب داشته باشد:

حالت اول: d_x^*جوابی از برنامه خطی $\hat{H}$ است. حال برای $h \in H' \subseteq H$ داریم:

$$d_x^* \eta_x + \eta_y = 0$$

پس نرمال خارجی همه صفحات داخل H'با بردار $\vec{d} = (d_x^*, 1)$ عمود هستند. یعنی خودشان با هم موازی هستند و با محور x ها برخورد دارند. که به برنامه خطی یک بعدی مشابه قبل تبدیل می شود، که در زمان خطی قابل حل است حال اگر این برنامه خطی یک بعدی که $\hat{H}'$ نامیده می شود، ممکن باشد آنگاه H' هم ممکن است آنگاه $\hat{H}$ بی‌کران است پس برنامه خطی حالت بی‌کران دارد. پس می‌توان برای آن در زمان O(n) یک پرتو مناسب یافت.

اگر $\hat{H}'$ناممکن باشد آنگاه H' و در نتیجه H هم ناممکن است.

مسئله جواب نداشته باشد:

حالت دوم: اگر $\hat{H}$ جواب ممکن نداشته باشد، طبق لم برنامه خطی کراندار است.

سؤال: آیا اطلاعات بیشتری می‌توان بدست آورد؟

پاسخ: $\hat{H}$ناممکن است اگر و تنها اگر راست‌ترین محدودیتی که از سمت چپ مسئله را کراندار می‌کند سمت راست چپترین محدودیتی باشدکه از سمت راست مسئله را محدود می‌کند. اگر اولی را $\hat{h}_1$ و دیگری را $\hat{h}_2$ بنامیم، این دو محدودیت اشتراک تهی دارند. اگر h_1 و h_2 محدودیتهای متناظر این دو در مسئله اصلی باشند آنگاه می‌توان گفت $(\{h_1, h_2\}, \vec{c})$ کراندار است. h_1و h_2 را certificate می‌نامیم.

استفاده از certificateها:

می‌توان از certificates به جای دو محدودیت m_1 و m_2 در الگوریتم 2DRandomizedLP استفاده کرد. بنابراین دیگر نیازی به محاسبه محدودیتهای غیرواقعی نیست.

با این روش کرانداری برنامه خطی $(\{h_1, h_2\}, \vec{c})$ ضمانت می شود ولی نمی توان گفت که جواب بهینه یکتا دارد. یعنی وقتی که ناممکن بودن برنامه خطی یک بعدی به این دلیل باشد که $\eta(h_1) = (0, -1)$ آنگاه بقیه نیم صفحات را بررسی می‌کنیم، اگر h_2 ای یافت شد به طوری که $\eta_x(h_2) > 0$ آنگاه h_1 و h_2 تضمین می‌کنند که مسئله یک جواب بهینه کوچک قاموسی دارد.

استفاده از certificateها:

اگر چنین h_2 ای یافت نشد آنگاه یا مسئله infeasible است یا یک جواب کوچک بهینه قاموسی ندارد که در این صورت می‌توان آن را با برنامه خطی یک بعدی که با نیم صفحات h با خاصیت $\eta_x(h) = 0$ ساخته می‌شود، حل کرد که اگر feasible باشد، می‌توان برای آن یک پرتو ρ در جهت $(-1,0)$ یافت که تمام نقاط روی ρ نقاط بهینه ممکن هستند.

الگوریتم کلی:

Algorithm 2DRANDOMIZEDLP($H, \vec{c}$)

Input. A linear program $(H, \vec{c})$, where H is a set of n half-planes and $\vec{c} \in \mathbb{R}^2$.

Output. If $(H, \vec{c})$ is unbounded, a ray is reported. If it is infeasible, then two or three certificate half-planes are reported. Otherwise, the lexicographically smallest point p that maximizes $f_{\vec{c}}(p)$ is reported.

1. Determine whether there is a direction vector $\vec{d}$ such that $\vec{d} \cdot \vec{c} > 0$ and $\vec{d} \cdot \vec{\eta}(h) \geqslant 0$ for all $h \in H$.
2. **if** $\vec{d}$ exists
3. **then** compute H' and determine whether H' is feasible.
4. **if** H' is feasible
5. **then** Report a ray proving that $(H, \vec{c})$ is unbounded and quit.
6. **else** Report that $(H, \vec{c})$ is infeasible and quit.
7. Let $h_1, h_2 \in H$ be certificates proving that $(H, \vec{c})$ is bounded and has a unique lexicographically smallest solution.
8. Let v_2 be the intersection of ℓ_1 and ℓ_2.
9. Let $h_3, h_4, \ldots, h_n$ be a random permutation of the remaining half-planes in H.
10. **for** $i \leftarrow 3$ **to** n
11. **do if** $v_{i-1} \in h_i$
12. **then** $v_i \leftarrow v_{i-1}$
13. **else** $v_i \leftarrow$ the point p on ℓ_i that maximizes $f_{\vec{c}}(p)$, subject to the constraints in H_{i-1}.
14. **if** p does not exist
15. **then** Let h_j, h_k (with $j, k < i$) be the certificates (possibly $h_j = h_k$) with $h_j \cap h_k \cap \ell_i = \emptyset$.
16. Report that the linear program is infeasible, with h_i, h_j, h_k as certificates, and quit.
17. **return** v_n

قضیه ۴.۱۰:

یک مسئله برنامه خطی ۲بعدی با n معادله در زمان مورد انتظار تصادفی O(n) و بدترین حالت ذخیره سازی خطی قابل حل است.