بهینه‌سازی ترکیبیاتی و برنامه‌ریزی خطی

محمد مهینی

دانشکده‌ی مهندسی کامپیوتر

دانشگاه صنعتی شریف

فهرست

۱ مسائل بهینه‌سازی و برنامه‌ریزی ریاضی ۵

۱-۱ مسائل بهینه‌سازی ۵

۱-۲ برنامه‌ریزی ریاضی ۱۰

۱-۳ برنامه‌ریزی محدب ۱۱

۱-۴ برنامه‌ریزی خطی ۱۷

۲ برنامه‌ریزی خطی ۲۲

۲-۱ فرم‌های برنامه‌ریزی خطی ۲۲

۲-۲ جواب ممکن مقدماتی ۲۵

۳ جلوه‌های هندسی برنامه‌ریزی خطی ۳۶

۳-۱ قضیه‌ی *Caratheodory*، کاربردی از برنامه‌ریزی خطی ۳۷

۳-۲ جلوه‌های هندسی برنامه‌ریزی خطی ۳۹

۳-۳ خلاصه و یک مثال ویژه ۴۳
۳-۳-۱ نمای بیرونی یک برنامه خطی ۴۳
۳-۳-۲ معنای هندسی ۴۵

۴ الگوریتم *Simplex* ۴۶

۴-۱ ورودی به الگوریتم *Simplex* ۴۶
۴-۱-۱ فاز شروع ۴۷
۴-۱-۲ فاز حرکت ۴۸

۴-۲ چگونه جواب ممکن مقدماتی جدید را باید انتخاب کرد ۵۲

۴-۳ الگوریتم *Simplex* ۵۵
۴-۳-۱ سوالاتی درباره الگوریتم *Simplex* ۵۶
۴-۳-۲ قوانین محور اصلی ۵۷
۴-۳-۳ *Simplex* یک الگوریتم از زمان چندجمله‌ای نمی‌باشد ۵۸

۴-۴ تحلیل *Smooth* ۶۳
۴-۴-۱ نتایج *Kalai* ۶۴

۵ الگوریتم‌های ترکیبیاتی احتمالی ۶۶

۵-۱ الگوریتم‌های ترکیبیاتی احتمالی ۶۶
۵-۱-۱الگوریتم *Seidel* ۶۶
۵-۱-۲تحلیل الگوریتم *Seidel* ۶۹
۵-۱-۳پیشرفت *Matousek, Sharir, Welzl* ۷۱

۶ Duality ۷۲

۶-۱ قضیه *Duality* ۷۲

۶-۲ کمی درباره‌ی *Duality* ۷۶
۶-۲-۱تبدیل فرم‌های مختلف *primal* به *dual* ۷۶
۶-۲-۲موارد استفاده : کوتاهترین مسیر ۷۸

۶-۳ قضیه ضعیف و قوی *Duality* ۸۱

۶-۴ رده‌بندی‌های ممکن *primal* و *dual* ۸۳

۶-۵ *Complementary slackness* ۸۴

۶-۶ امکان جواب در یک سیستم خطی و لم *Farkas* ۸۵
۶-۶-۱لم *Farkas* ۸۶

۷ **قضیه کم‌بیشینه‌ی *Von Neumann*، اصل کم‌بیشینه *Yao*** ۸۹

۷-۱ اصل کمبیشینه *von Neumann* ۸۹

۷-۲ اصل کمبیشینه Yao ۹۳

فصل ۱

مسائل بهینه‌سازی و برنامه‌ریزی ریاضی

در این فصل قصد داریم، سوالاتی را که از نوع بهینه‌سازی می‌باشند معرفی کرده، و کاربرد و گستردگی آنها را در مسائل مختلف ببینیم.
در ادامه به دنبال مدل‌هایی ریاضی برای حل چنین مسائلی می‌گردیم. در ابتدا با کلی‌ترین مدل موجود که به آن مدل ریاضی گفته می‌شود آشنا شده، در ادامه با بررسی نیازهایمان سعی در ایجاد مدلی با کارایی بالا می‌کنیم، که بتواند مسائل ما را حل کرده و یا دید مناسبی نسبت به مسئله به ما بدهد. در فصول بعد خواهیم دید که این روش که آن را برنامه‌ریزی خطی می‌نامیم، می‌تواند ما را به حلی الگوریتمی که کاملاً دور از هرگونه فرمول ریاضی باشد، راهنمایی می‌کند.

۱-۱ مسائل بهینه‌سازی

امروزه به مسائلی بر می‌خوریم که در آنها هدف بیشینه کردن یا کمینه کردن چیزی است. مثلاً در بازار بورس یا هر بازار دیگری افراد سعی در بیشینه کردن سود خود دارند؛ و یا کشوری که می‌خواهد با کمترین هزینه شبکه ریلی خود را ایجاد کند.

در کل همه روزه افراد زیادی در همه علوم و جوامع سعی در حل مسائلی می‌کنند که حل آنها نیاز به بیشینه یا کمینه کردن مقداری دارد، این مقدار سود یا ضرر یک شرکت،

بیشینه کردن کارایی، بیشینه کردن ضریب اطمینان، کمینه کردن مصرف، و ... می‌تواند باشد.

مثال ۱-۱ . بیشترین مساحتی که توسط یک خم بسته که طول آن ۱ است، در فضای دو بعدی محدود می‌شود، چقدر است؟

این مسئله یک مسئله بهینه سازی است که می‌خواهیم مساحتی را بیشینه کنیم.

مثال ۱-۲ . کارخانه‌ای توانایی تولید اجناسی را دارا می‌باشد، که آنها را با ۱ تا n شماره‌گذاری می‌کنیم. برای تولید این اجناس نیاز به مواد خامی داریم که آنها را با ۱ تا m شماره‌گذاری می‌کنیم. هر واحد از ماده خام i ام ρ_i تومان قیمت دارد. همچنین هر واحد از جنس i ام که ما می‌فروشیم c_i تومان قیمت دارد. برای تولید جنس i ام a_{ij} واحد از ماده خام j ام نیاز است.

تا به اینجا با واقعیتی سر و کار داشتیم که افراد زیادی را درگیر خود می‌کند.

مدیر تولید به عنوان یک بیشینه‌گر: یکی از افرادی که می‌خواهد سود شرکت را بیشینه کند. برای این کار دوست دارد اختلاف مقدار فروش و هزینه خرید مواد خام را بیشینه کند (یعنی همان سود شرکت). فرض کنید در این حالت مقدار هر ماده خام هم محدود و برابر σ_i می‌باشد. او می‌تواند سود خود را با مشخص کردن اینکه چه مقدار از هر جنس را تولید کند، تغییر دهد و به طبع آن بیشینه کند.

تعریف ۱-۱ . یک سوال **بهینه‌سازی** دارای یک مجموعه دامنه‌ی D و یک تابع مقدار به فرم $f : D \to \mathbb{R}$ می‌باشد.

$f(x) \in \mathbb{R}$ مشخص کننده مقدار «سود» یا «هزینه»ای است که توسط $x \in D$ مشخص می‌شود.

مسائل بهینه‌سازی را می‌توان به دو دسته‌ی مسائل کمینه‌سازی و مسائل بیشینه‌سازی تقسیم کرد. در یک مسئله بیشینه‌سازی، هدف پیدا کردن $x \in D$ است به طوری که برای

همه $y \in D$ داشته باشیم $f(x) \geq f(y)$. به معنای دیگر، ما عنصری در دامنه را می‌خواهیم که بیشترین سود را به ما بدهد. بعکس در یک مسئله کمینه‌سازی، هدف پیدا کردن $x \in D$ است به طوری که برای همه $y \in D$ داشته باشیم $f(x) \leq f(y)$. به معنای دیگر، ما عنصری در دامنه را می‌خواهیم که کمترین هزینه را داشته باشد.

مثال ۱-۳ . به ازای چه مقدار x تابع $\frac{x}{\sin x}$ به بیشترین مقدار خود می‌رسد.

از آنجایی که می‌دانیم $\lim_{x\to\infty} \frac{x}{\sin x} = \infty$، در نتیجه چنین xی وجود ندارد.

در نتیجه، ممکن است تابع f جواب بهینه در دامنه خود نداشته باشد؛ یعنی ممکن است بتوانیم هر مقدار که می‌خواهیم f را بزرگ کنیم، ولی اگر دامنه مسئله یعنی D، متناهی باشد، در این صورت f هم دارای مقدار کمینه و هم دارای مقدار بیشینه در D می‌باشد.

در اینجا به بررسی چند مثال می‌پردازیم:

مثال ۱-۴ . فرض کیند p یک چند جمله‌ای تک متغیری می‌باشد. چه زمان p بیشترین مقدار را دارد اگر دامنه آن را به بازه $[۰, ۱] \subset \mathbb{R}$ محدود کنیم.

این مسئله یک مسئله بیشینه‌سازی است که دامنه ما $D = [۰, ۱]$ و تابع مقدار ما p است.

مثال ۱-۵ . در «مثال ۱» دامنه D شامل تمام خم‌های بسته به طول ۱ می‌باشد، و تابع مقدار ما یعنی f مساحت داخل خم در فضای دوبعدی است.

مثال ۱-۶ . مسئله کلاسیک درخت فراگیر مینیمم (MST) را در نظر بگیرید. فرض کنید ما یک گراف $G = (V, E)$ و یک تابع وزن $w : E \to \mathbb{R}$ روی یال‌های گراف داریم. وزن یک زیرگراف G' از G با مجموعه یالهای $E' \subseteq E$ برابر مجموع وزن یال‌های آن می‌باشد، $W(G') \stackrel{def}{=} \sum_{e\in E'} w(e)$. یک مسیر در G دنباله‌ای از رئوس $v_۱, \ldots, v_n \in V$ می باشد به

طوری که هر (v_i, v_{i+1}) یالی از گراف G باشد. یک دور یک مسیر بسته است، یعنی $v_1 = v_n$. G همبند است اگر بین هر دو رأس u, v در گراف یک مسیر وجود داشته باشد و بدون دور است اگر هیچ دوری نداشته باشد. یک درخت فراگیر T از G یک زیرگراف همبند بدون دور از G با مجموعه رئوس V می‌باشد. درخت فراگیر مینیمم در G را بدست آورید.

می‌توان دید، این مسئله یک مسئله کمینه سازی است. دامنه مسئله $\{T$ درخت فراگیر G باشد: $\}$، $\tau(G) \stackrel{def}{=} \{T$ و تابع مقدار مجموع وزن یال‌ها می‌باشد. در نظر داشته باشید از آنجایی که $|\tau(G)| \leq 2^{|E|}$، در این مسئله دامنه متناهی است. بهترین الگوریتم‌هایی که برای این مسئله شناخته شده است الگوریتم‌های پریم[1] و کروسکال[2] و چازل[3] می‌باشند که از زمان چندجمله‌ای هستند.

مثال ۱–۷ . یک مسئله بهینه سازی کلاسیک دیگر، مسئله فروشنده دوره‌گرد[4] می‌باشد. فرض کنید $v_1, \ldots, v_n$ نماینده شهرهای یک کشور می‌باشند. تابع فاصله $d(u, v)$ فاصله بین شهرهای u و v را نشان می‌دهند. یک فروشنده دوره‌گرد قصد دارد اجناس خود را در تمام شهرهای این کشور به فروش برساند. هدف فروشنده پیدا کردن مسیری است که، در کوتاهترین زمان همه شهرهای کشور را بگردد. این مسئله نیز یک مسئله کمینه سازی است. دامنه این مسئله $\{H$ یک مسیر می‌باشد که از همه شهرها می‌گذرد $: H\}$، $H(G) \stackrel{def}{=} \{H$ و تابع مقدار آن طول مسیر می‌باشد. همانند مسئله MST دامنه این مسئله نیز محدود می‌باشد. ولی برخلاف مسئله MST، برای این مسئله تا به حال راه حلی چندجمله‌ای پیدا نشده و یک مسئله‌ی $NP - Hard$ می‌باشد.

---

[1] Prim

[2] Kruskal

[3] Chazelle

[4] TSP

مثال ۱–۸ . یک شبکه شار یک گراف جهت‌دار $G = (V, E)$ است که روی یال‌ها ظرفیت $c : E \to \mathbb{R}^{\geq 0}$ تعریف می‌شود، و رئوس مبدأ و مقصد آن به ترتیب s و t می‌باشد. برای هر رأس $v \in V$، مجموعه یال‌هایی که به این راس وارد می‌شوند را با $IN(v)$ و مجموعه یال‌هایی که از این راس خارج می‌شود را با $OUT(v)$ نمایش می‌دهیم. یک شار در G به یک تابع $f : E \to \mathbb{R}$ گفته می‌شود که سه شرط زیر را دارا باشد:

برای هر یال e، $f(e) \geq 0$.

برای هر یال e، $f(e) \leq c(e)$.

برای هر رأس v بجز s, t داشته باشیم: $\sum_{e \in IN(v)} f(e) = \sum_{e \in OUT(v)} f(e)$.

اندازه یک شار برابر مقدار شاری است که از s خارج و به t وارد می‌شود، در واقع اندازه f برابر است با $||f|| \stackrel{def}{=} \sum_{e \in OUT(s)} f(e) - \sum_{e \in IN(s)} f(e)$. بیشترین مقدار شار در گراف G چقدر است؟

این سوال یک مسئله بیشینه سازی است. دامنه این مسئله $\{f$ یک شار در G باشد $: F(G) \stackrel{def}{=} \{f$ و تابع مقدار آن اندازه شار است. همچنین با اینکه دامنه $F(G)$ نامتناهی است، ولی برای حل این مسئله، الگوریتم‌هایی از زمان چندجمله‌ای وجود دارد. یکی از بهترین الگوریتم‌ها برای حل این سوال، الگوریتم ادموندز[۵]–کارپ[۶] است. سریع‌ترین الگوریتمی که تا به حال برای حل این سوال پیدا شده است، الگوریتم $push - relabel$ می‌باشد.

یکی از وجوه جالب شبکه‌های شار، قضیه «شار ماکسیمم–برش مینیمم[۷]» است. یک برش در G، به تقسیم رئوس گراف به دو مجموعه S و T گفته می‌شود بطوریکه $S \bigcup T = V$ و همچنین $s \in S$ و $t \in T$ باشد. وزن برش (S, T) برابر $C(S, T) \stackrel{def}{=} \sum_{e \in E(S,T)} c(e)$ می‌باشد بطوریکه $E(S, T) \stackrel{def}{=} \{(u, v) \in E : u \in S, v \in T\}$.

---

۵ Edmonds

۶ Karp

۷ maximum flow-minimum cut

مجموعه تمام برش‌های موجود در گراف G را با $C(G)$ نمایش می‌دهیم.

قضیه ۱-۱ . اندازه شار ماکسیمم در گراف G، برابر اندازه برش مینیمم در G می‌باشد.

این قضیه، یک حالت خاص از قضیه‌ی *duality* در برنامه‌ریزی خطی[۸] می‌باشد که در بخش‌های آینده درباره آن صحبت خواهیم کرد.

۱-۲ برنامه‌ریزی ریاضی

در بخش قبل، تعاریفی که گفته شد، کمی دور از واقعیت و مجازی می‌باشند تا اینکه الگوریتمی و جذاب باشند. در این بخش قصد داریم به برنامه‌ریزی ریاضی بپردازیم که کمی به واقعیت نزدیک‌تر است.

تعریف ۱-۲ . یک **برنامه‌ریزی ریاضی** شامل تابع مقدار $f : \mathbb{R}^n \to \mathbb{R}$ که قصد داریم آن را کمینه کنیم، مجموعه‌ای از m تابع قید $g_i : \mathbb{R}^n \to \mathbb{R}$ که دامنه جستجوی ما را مشخص می‌کنند و یک بردار ثابت $\vec{b} = (b_1, \ldots, b_m) \in \mathbb{R}^m$ می‌باشد. دامنه مسئله $D \subseteq \mathbb{R}^n$، توسط نامساوی‌های $g_i(\vec{x}) \leq b_i$، به صورت $D = \bigcap_{i=1}^{m} \{\vec{x} \in \mathbb{R}^n : g_i(\vec{x}) \leq b_i\} = \{\vec{x} \in \mathbb{R}^n : g_i(\vec{x}) \leq b_i, 1 \leq i \leq m\}$ تعریف می‌شود. هدف پیدا کردن بردار $\vec{x} \in D$ می‌باشد، به طوری که $f(\vec{x})$ کمینه شود.

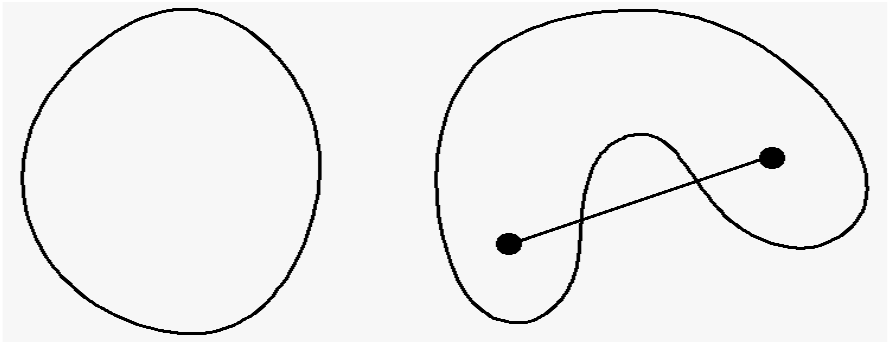
معمولاً برنامه‌ریزی ریاضی را به صورت زیر نشان می‌دهیم:

$$\min f(\vec{x})$$

$$g_i(\vec{x}) \leq b_i \quad , \quad i = 1 \ldots m$$

اگر دامنه ما به صورت زیرمجموعه‌ای از یک فضای اقلیدسی n-بعدی باشد، قید قوی‌تری است و مسائل در با این قید راحت‌تر حل می‌شوند. بعضی از مسائل، مانند مثال ۲ بالا را

[۸] Linear Programming



شکل ۱: یک مجموعه محدب در چپ و یک مجموعه غیر محدب در راست

نمی‌توان به صورت یک برنامه‌ریزی ریاضی نوشت. برنامه‌ریزی ریاضی هنوز یک حالت کلی برای هدف ما می‌باشد؛ از آنجایی که می‌توان بسیاری از مسائل پیچیده را به برنامه‌ریزی ریاضی تبدیل کرد، این روش کمک چندانی در حل مسائل به ما نمی‌کند. در واقع گستردگی و کلی بودن این روش دست ما را برای استفاده از ایده‌های خاص جهت حل مسئله می‌بندد. در نتیجه ما ترجیح می‌دهیم در حالات خاص‌تری کار کنیم، بطوریکه با توابع f و g زیباتری سروکار داشته باشیم.

۱-۳ برنامه‌ریزی محدب

تعریف ۱-۳ . مجموعه $S \subseteq \mathbb{R}^n$ **محدب** است اگر هر خط راستی که دو نقطه در S را به هم وصل می‌کند، خود در S باشد. به بیان دیگر یک مجموعه $S \subseteq \mathbb{R}^n$ محدب است اگر برای هر $\vec{x}, \vec{y} \in S$ و $\lambda \in [0, 1]$ داشته باشیم $\lambda \vec{x} + (1-\lambda) \vec{y} \in S$.

ما می‌توانیم به نقطه $\lambda \vec{x} + (1-\lambda) \vec{y}$ به صورت میانگین وزنی $\vec{x}$ و $\vec{y}$ که وزن‌ها برابر λ و $(1-\lambda)$ می‌باشند ($0 \leq \lambda \leq 1$)، نگاه کنیم. این مدل میانگین‌ها، به ترکیب‌های محدب معروف‌اند. اگر S محدب باشد، در این صورت هر ترکیب محدب از نقاط داخل S نیز درون S قرار می‌گیرد، در نتیجه S نسبت به ترکیب‌های محدب بسته است.

مثال ۱-۹ .(مجموعه‌های محدب) در زیر مثال‌هایی از مجموعه‌های محدب آمده است:

یک توپ n–بعدی : $\{\vec{x} \in \mathbb{R}^n : \| \vec{x} \|_۲ \leq ۱\}$

هر زیرفضای برداری از $\mathbb{R}^n$

در شکل ۱، **مجموعه سمت چپ**

مثال ۱–۱۰ .(مجموعه‌های غیرمحدب) در زیر مثال‌هایی از مجموعه‌های غیرمحدب آمده است:

$\mathbb{R}^n - \{\vec{۰}\}$

در شکل ۱، **مجموعه سمت راست**

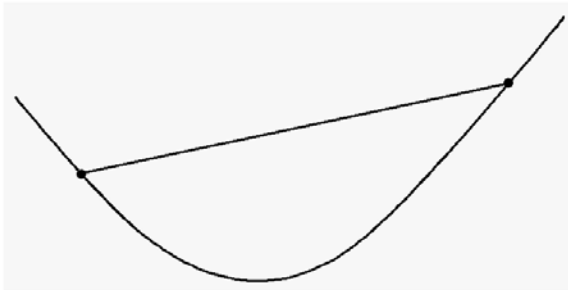
تعریف ۱–۴ . فرض کنید $S \subseteq \mathbb{R}^n$ یک مجموعه محدب باشد. یک تابع $f : S \to \mathbb{R}$ محدب است، اگر برای هر $\vec{x}, \vec{y} \in S$ و $\lambda \in [۰, ۱]$ داشته باشیم $f(\lambda \vec{x} + (۱ - \lambda) \vec{y}) \leq \lambda f(\vec{x}) + (۱ - \lambda) f(\vec{y})$. دقت کنید بخاطر محدب بودن S، $\vec{z} = \lambda \vec{x} + (۱ - \lambda) \vec{y}$ عضو S می‌باشد، و در نتیجه می‌توان مقدار f را توسط $\vec{z}$ محاسبه کرد. به بیان دیگر، f یک تابع محدب است اگر تصویر هر ترکیب محدب از نقاط در S تحت f توسط ترکیب محدب تصاویر آنها از بالا محدود شده باشد.

از دیدگاه هندسی، f یک تابع محدب است اگر اپیگراف[۹] تابع f که بصورت $epi(f) \stackrel{def}{=} \{(\vec{x}, y) \in \mathbb{R}^{n+۱} : \vec{x} \in S, y \geq f(\vec{x})\}$ تعریف می‌شود، یک مجموعه محدب باشد. برای توابع چند بعدی، این بدان معناست که هر خط راستی که دو نقطه را روی نمودار f به هم وصل می‌کند در داخل یا بالای نمودار تابع قرار گیرد.

مثال ۱–۱۱ .(توابع محدب) توابع خطی، نرم‌ها، x^k برای $x \in \mathbb{R}^{\geq ۰}$ و $k \in N$ و تابعی که در شکل ۲ نشان داده شده است، **توابعی محدب می‌باشند.**

---

[۹] epigraph



شکل ۲: تابع محدب

مثال ۱-۱۲ .(توابع غیرمحدب) توابع رادیکالی $\sqrt{x}$ که $x \in \mathbb{R}^{\geq 0}$.

برای اینکه ببینیم تابع $\sqrt{x}$ محدب نیست، داریم:

$$\sqrt{\frac{1}{2}0 + \frac{1}{2}100} = \sqrt{50} \approx 7.07 > \frac{1}{2}\sqrt{0} + \frac{1}{2}\sqrt{100} = 5$$

نرم یک تابع، تابع $\| \vec{x} \| : \mathbb{R}^n \to \mathbb{R}$ می‌باشد که در شرایط زیر صدق می‌کند:

- $\| \vec{x} \| > 0$ برای هر $\vec{x} \in \mathbb{R}^n - \{\vec{0}\}$ و $f(\vec{0}) = 0$.
- $\|c\,\vec{x}\| = |c|.\| \vec{x} \|$ برای هر $\vec{x} \in \mathbb{R}^n$ و $c \in \mathbb{R}$.
- $\| \vec{x} + \vec{y} \| \leq \| \vec{x} \| + \| \vec{y} \|$ برای هر $\vec{x}, \vec{y} \in \mathbb{R}^n$ (نامساوی مثلثی).

لم ۱-۱ . توابع نرم، توابعی محدب هستند.

اثبات. فرض کنید $\| \vec{x} \| : \mathbb{R}^n \to \mathbb{R}$، یک تابع نرم باشد و $\vec{x}, \vec{y} \in \mathbb{R}^n$ و $\lambda \in [0, 1] \subseteq \mathbb{R}$. داریم:

$$\|\lambda\,\vec{x} + (1-\lambda)\,\vec{y}\| \leq \|\lambda\,\vec{x}\| + \|(1-\lambda)\,\vec{y}\|$$

$$= |\lambda|.|| \vec{x} || + |(۱ - \lambda)|.|| \vec{y} ||$$

$$= \lambda|| \vec{x} || + (۱ - \lambda)|| \vec{y} ||$$

□

حال ما آماده‌ایم به تعریف برنامه‌ریزی محدب بپردازیم.

تعریف ۱-۵ . یک برنامه‌ریزی ریاضی

$$\min f(\vec{x})$$

$$g_i(\vec{x}) \leq b_i \quad , \quad i = ۱ \dots m$$

محدب است اگر تابع مقدار $f : \mathbb{R}^n \to \mathbb{R}$ و توابع قید $g_i : \mathbb{R}^n \to \mathbb{R}$، محدب باشند. در نتیجه طبق تعریف در این حالت، دامنه یک مجموعه محدب می‌باشد.

در اینجا به دو لم زیر نیازمندیم.

لم ۱-۲ . محل برخورد m مجموعه محدب، خود محدب است. در واقع اگر S_i ها مجموعه‌هایی محدب باشند، $S = \bigcap_{i=۱}^{m} S_i$ نیز محدب است.

اثبات. فرض کنید $\vec{x}, \vec{y} \in S = \bigcap_{i=۱}^{m} S_i$ و $\lambda \in [۰, ۱]$. از آنجایی که برای هر $۱ \leq i \leq m$، بخاطر محدب بودن S_i داریم $\vec{z} = \lambda \vec{x} + (۱ - \lambda) \vec{y} \in S_i$، در نتیجه $\vec{z} \in \bigcap_{i=۱}^{m} S_i = S$. □

لم ۱-۳ . فرض کنید $g : \mathbb{R}^n \to \mathbb{R}$ یک تابع محدب و $b \in \mathbb{R}$ یک عدد ثابت باشد. در این صورت مجموعه $S = \{\vec{x} \in \mathbb{R}^n : g(\vec{x}) \leq b\} \subseteq \mathbb{R}^n$ یک مجموعه محدب است.

اثبات. فرض کنید $\vec{x}, \vec{y} \in S$ و $\lambda \in [۰, ۱]$ باشد. می‌خواهیم نشان دهیم $\lambda \vec{x} + (۱ - \lambda) \vec{y}$ عضو S می‌باشد. برای این کار کافیست ثابت کنیم $g(\lambda \vec{x} + (۱ - \lambda) \vec{y}) \leq b$. در نتیجه، داریم:

$$g(\lambda \vec{x} + (۱ - \lambda) \vec{y}) \leq \lambda g(\vec{x}) + (۱ - \lambda) g(\vec{y})$$

$$\leq \max\{g(\vec{x}), g(\vec{y})\} \leq b$$

□

قضیه زیر نتیجه‌ای مستقیم از لم‌های بالا می‌باشد.

قضیه ۱-۲ . برنامه‌ریزی محدب زیر را در نظر بگیرید:

$$\min f(\vec{x})$$

$$g_i(\vec{x}) \leq b_i \quad , \quad i = ۱ \dots m$$

دامنه D_i را فضای محصور شده توسط تابع g_i در نظر می‌گیریم، در واقع $D_i = \{\vec{x} \in \mathbb{R}^n : g_i(\vec{x}) \leq b_i\}$. D دامنه‌ی این مسئله که $D = \bigcap_{i=۱}^{m} D_i$ می‌باشد، یک مجموعه محدب است.

فرض کنید D یک زیرمجموعه غیرتهی از $\mathbb{R}^n$ و $f : \mathbb{R}^n \to \mathbb{R}$ باشد. می‌دانیم $\vec{x} \in D$ یک مینیمم موضعی برای f در D است اگر $f(\vec{x})$ کمترین مقدار خود را در یک کره n-بعدی به مرکز $\vec{x}$ که در داخل D قرار دارد، داشته باشد. به طور دقیق‌تر، اگر $\epsilon \in \mathbb{R}^{>۰}$ وجود داشته باشد که برای هر $\vec{y} \in D$ که $\| \vec{x} - \vec{y} \|_۲ \leq \epsilon$، داشته باشیم $f(\vec{x}) \leq f(\vec{y})$، $\vec{x}$ یک مینیمم موضعی برای تابع f در دامنه D می‌باشد. همچنین اگر برای هر بردار $\vec{y}$ در دامنه D، $f(\vec{x}) \leq f(\vec{y})$ باشد، در این صورت $\vec{x}$ یک مینیمم اکید برای f در D می‌باشد. ماکسیمم موضعی و اکید نیز به همین ترتیب تعریف می‌شوند.

یکی از دلایلی که مسائل بهینه‌سازی مسائل سختی می‌باشند، این است که تعداد بسیار زیادی عضو بهینه موضعی وجود دارد، که از عضو بهینه اکید بسیار دور هستند. به همین دلیل یک عضو بهینه موضعی به طور حتم نمی‌تواند ما را به یک عضو بهینه اکید راهنمایی کند.

با تعاریفی که از برنامه‌ریزی محدب ارائه دادیم، خواهیم دید که این پیامد در برنامه‌ریزی محدب وجود ندارد. در واقع خواهیم دید، هر مینیمم موضعی برای تابع f در دامنه D در برنامه‌ریزی محدب، یک مینیمم اکید برای f در D می‌باشد.

قضیه ۱-۳ . فرض کنید $f : \mathbb{R}^n \to \mathbb{R}$ یک تابع محدب باشد. در این صورت هر بردار مینیمم موضعی $\vec{x}$ برای تابع f در مجموعه محدب $D \subseteq \mathbb{R}^n$، یک مینیمم اکید برای f در D می‌باشد.

اثبات. فرض کنید $\vec{y} \in D$ یک نقطه در دامنه و $\vec{x} \in D$ یک مینیمم موضعی برای f در D باشد. هر ترکیب محدب $\vec{z} = \lambda\, \vec{x} + (1-\lambda)\, \vec{y}$ که $\lambda \in [0, 1]$ را در نظر بگیرید که به $\vec{x}$ نزدیک باشد، یعنی $\| \vec{x} - \vec{z} \| \leq \epsilon$. در نتیجه به‌خاطر محدب بودن D، $\vec{z} \in D$ می‌باشد. همچنین از آنجایی که $\vec{x}$ مینیمم موضعی است، $f(\vec{x}) \leq f(\vec{z})$ می‌باشد. داریم:

$$\begin{aligned} f(\vec{x}) \leq f(\vec{z}) &= f(\lambda\, \vec{x} + (1-\lambda)\, \vec{y}) \\ &\leq \lambda f(\vec{x}) + (1-\lambda) f(\vec{y}) \end{aligned}$$

در نتیجه $(1-\lambda) f(\vec{x}) \leq (1-\lambda) f(\vec{y})$. از آنجایی که $1 - \lambda \geq 0$، می‌توان نتیجه گرفت $f(\vec{x}) \leq f(\vec{y})$ و قضیه ثابت می‌شود. □

مثال ۱-۱۳ . به یاد آورید l_2 یا همان نرم اقلیدسی $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$، به صورت $\| \vec{x} \|_2 \stackrel{def}{=} \sqrt{\sum_{i=1}^n x_i^2}$ تعریف می‌شود، و l_1 یا نرم منهتن[10] به صورت $\| \vec{x} \|_1 \stackrel{def}{=} \sum_{i=1}^n x_i^2$ تعریف می‌شود. کدام بردار $\vec{x} = (x_1, x_2) \in \mathbb{R}^2$ در ربع اول دارای کوچکترین نرم اقلیدسی می‌باشد با این شرط که نرم منهتن آن ۱ باشد؟ از آنجایی که $\vec{x}$ در ربع اول قرار دارد، در نتیجه $\sum_{i=1}^2 |x_i| = \sum_{i=1}^2 x_i$. برنامه‌ی زیر را برای حل این مسئله می‌نویسیم:

$$\begin{aligned} &\min \sqrt{x_1^2 + x_2^2} \\ &x_1 \geq 0 \\ &x_2 \geq 0 \\ &x_1 + x_2 = 1 \end{aligned}$$

[10] Manhattan

البته، این برنامه در فرم درست خود قرار ندارد، یعنی رابطه تساوی و بزرگتری وجود دارد، پس این برنامه را به صورت زیر بازنویسی می‌کنیم:

$$\min \sqrt{x_1^2 + x_2^2}$$
$$-x_1 \leq 0$$
$$-x_2 \leq 0$$
$$x_1 + x_2 \leq 1$$
$$-x_1 - x_2 \leq 1$$

تابع مقدار از آنجایی که نرم است، محدب است، همچنین توابع قید چون خطی هستند، محدب می‌باشند.

۱-۴ برنامه‌ریزی خطی

برنامه‌ریزی خطی LP، یک حالت مهم از برنامه‌ریزی محدب است که در ۶۰ سال اخیر بسیار مورد توجه قرار گرفته است.

تعریف ۱-۶ . تابع $f : \mathbb{R}^n \to \mathbb{R}$ خطی است اگر:

$$f(\vec{x} + \vec{y}) = f(\vec{x}) + f(\vec{y}) \text{ و}$$

$$f(\lambda\, \vec{x}) = \lambda f(\vec{x})$$

برای همه $\vec{x}, \vec{y} \in \mathbb{R}^n$ و $\lambda \in \mathbb{R}$.

مشاهده . برای هر تابع خطی $f : \mathbb{R}^n \to \mathbb{R}$، می‌توان آن را به صورت زیر نوشت:

$$f(\vec{x}) = \langle \vec{c}, \vec{x} \rangle = \sum_{i=1}^{n} c_i x_i$$

که $c_i = f(\vec{e_i})$ و $\vec{e_i} = (0, \ldots, 0, 1, 0, \ldots, 0)$ برابر i امین بردار استاندارد واحد است (در مکان i ام آن ۱ و در بقیه مکان‌ها ۰ است).

اثبات.

$$f(\overrightarrow{x}) = f(\sum_{i=1}^{n} x_i \overrightarrow{e_i}) = \sum_{i=1}^{n} f(x_i \overrightarrow{e_i}) = \sum_{i=1}^{n} x_i f(\overrightarrow{e_i}) = \sum_{i=1}^{n} c_i x_i$$

□

لم ۱-۴ . توابع خطی، محدب می‌باشند.

اثبات. فرض کنید $f : \mathbb{R}^n \to \mathbb{R}$ یک تابع خطی باشد و $\overrightarrow{x}, \overrightarrow{y} \in \mathbb{R}^n$ و $\lambda \in [0, 1]$ باشد. در نتیجه:

$$\begin{aligned} f(\lambda \overrightarrow{x} + (1-\lambda) \overrightarrow{y}) &= f(\lambda \overrightarrow{x}) + f((1-\lambda) \overrightarrow{y}) \\ &= \lambda f(\overrightarrow{x}) + (1-\lambda) f(\overrightarrow{y}) \\ &\leq \lambda f(\overrightarrow{x}) + (1-\lambda) f(\overrightarrow{y}) \end{aligned}$$

□

در واقع، تصویر یک ترکیب خطی از دو نقطه در دامنه آن دقیقاً برابر ترکیب خطی تصاویر آن‌ها می‌باشد.

تعریف ۱-۷ . یک برنامه محدب

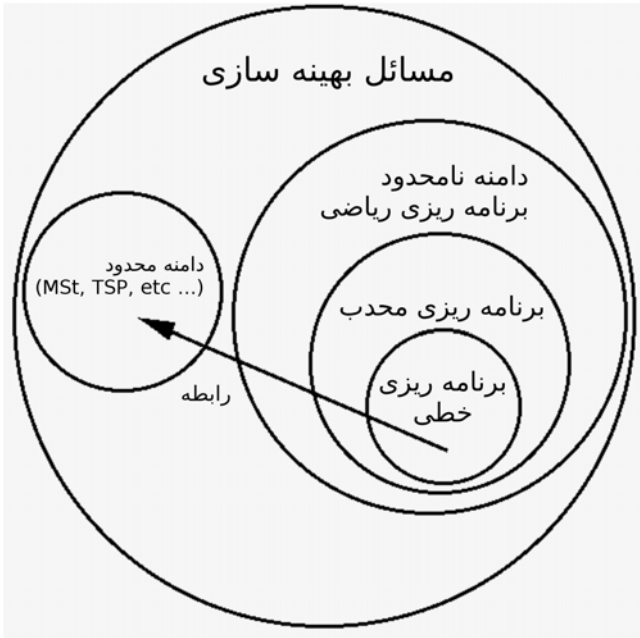
$$\min f(\overrightarrow{x})$$

$$g_i(\overrightarrow{x}) \leq b_i \quad , \quad i = 1 \dots m$$

خطی است اگر تابع مقدار $f : \mathbb{R}^n \to \mathbb{R}$ و توابع قید $g_i : \mathbb{R}^n \to \mathbb{R}$ خطی باشند و در نتیجه بتوانند به صورت $f(\overrightarrow{x}) = \sum_{j=1}^{n} c_j x_j$ و $g_i(\overrightarrow{x}) = \sum_{j=1}^{n} a_{ij} x_j$ نوشته شوند. در این صورت برنامه‌ریزی خطی به صورت زیر در می‌آید:

$$\min \sum_{j=1}^{n} c_j x_j$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad , \quad i = 1 \dots m$$



شکل ۳: فضای مسائل بهینه‌سازی

مثال ۱-۱۴ . (مسئله رژیم غذایی) هر فردی می‌خواهد برای خود یک برنامه غذایی داشته باشد، به طوری که در هر روز از هر ویتامین به مقدار کافی بخورد و همچنین مجموع هزینه غذاهایی که می‌خورد کمینه شود. n غذای متفاوت وجود دارد؛ هر گرم از غذای j ام قیمتی برابر $c_j \in \mathbb{R}$ دارد. انسان برای زنده ماندن نیاز به b_i میلیگرم از ویتامین i ام در هر روز دارد. افراد برای اینکه مقدار هزینه غذایی خود را کمینه کنند و همچنین زنده بمانند چه باید بکنند؟

x_j را برابر مقداری از غذای jام که هر آدم می‌خورد، قرار دهید. برنامه زیر را برای این مسئله می‌نویسیم:

$$\min \sum_{j=1}^{n} c_j x_j$$

$$x_j \geq 0 \quad , \quad i = 1 \ldots n$$

$$\sum_{j=1}^{n} a_{ij}x_j \geq b_i \quad , \quad i = 1 \dots m$$

البته این برنامه در فرم درست خود قرار ندارد، یعنی رابطه تساوی و بزرگتری وجود دارد، پس این برنامه را به صورت زیر بازنویسی می‌کنیم:

$$\min \sum_{j=1}^{n} c_j x_j$$

$$-x_j \leq 0 \quad , \quad i = 1 \dots n$$

$$\sum_{j=1}^{n} -a_{ij}x_j \leq b_i \quad , \quad i = 1 \dots m$$

مثال ۱-۱۵ . (شبکه شار) مسئله شار در شبکه را به یاد آورید. همان‌طور که گفته شد الگوریتم‌هایی چندجمله‌ای برای حل این مسئله وجود داشت. با این وجود، ما می‌توانیم از طریق راه حل‌های برنامه‌ریزی خطی نیز سعی در حل آن کنیم.

فرض کنید x_e برای هر یال $e \in E$ مقدار شاری باشد که از یال e رد می‌شود. در نتیجه برنامه زیر را داریم:

$$\max \sum_{e \in OUT(s)} x_e - \sum_{e \in IN(s)} x_e$$

$$x_e \geq 0 \quad , \quad \forall\, e \in E$$

$$x_e \leq c(e) \quad , \quad \forall\, e \in E$$

$$\sum_{e \in OUT(s)} x_e = \sum_{\in IN(s)} x_e \quad , \quad \forall\, v \in V - \{s, t\}$$

با توجه به آنچه گفته شد، مجبور به بازنویسی این برنامه هستیم، بطوری که برنامه‌ریزی خطی آن به صورت زیر درمی‌آید:

$$\min \sum_{e \in IN(s)} x_e - \sum_{e \in OUT(s)} x_e$$

$$-x_e \leq \circ \quad , \ \forall\, e \in E$$

$$x_e \leq c(e) \quad , \ \forall\, e \in E$$

$$\sum_{e \in OUT(s)} x_e - \sum_{\in IN(s)} x_e \quad \leq \circ, \ \forall\, v \in V - \{s,t\}$$

$$\sum_{e \in IN(s)} x_e - \sum_{\in OUT(s)} x_e \quad \leq \circ, \ \forall\, v \in V - \{s,t\}$$

فصل ۲

برنامه‌ریزی خطی

در بخش قبل به بررسی مسائل بهینه‌سازی پرداختیم و با نحوه نوشتن آن‌ها در برنامه‌ریزی ریاضی و برنامه‌ریزی خطی آشنا شدیم. در این فصل قصد داریم فرم‌های متعارف برای نوشتن برنامه‌ریزی خطی ، شامل فرم *standard* و *canonical* را بررسی کنیم.

۲-۱ فرم‌های برنامه‌ریزی خطی

از بخش قبل به یاد آورید که یک مسئله برنامه‌ریزی خطی به صورت زیر می‌نوشتیم:

$$\min \sum_{j=1}^{n} c_j x_j$$
$$x_j \geq 0 \quad , \quad i = 1 \dots m$$
$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i \quad , \quad i = 1 \dots m$$

برای سادگی می‌توان یک برنامه‌ریزی خطی را به صورت زیر نیز بنویسیم:

$$\min \langle c, x \rangle$$

$$\langle a_i, x \rangle \geq b_i \quad , \quad i = 1 \dots m$$

$$a_i \in \mathbb{R}^n \quad , \quad i = 1 \dots m$$

$$x_j \geq 0 \quad , \quad i = 1 \dots m$$

که $\langle , \rangle$ علامت ضرب داخلی می‌باشد. می‌توان به بیانی ساده‌تر نوشت:

$$\min\langle c, x\rangle$$
$$Ax \geq b$$
$$x \geq 0$$

تعریف ۲-۱ . یک برنامه‌ریزی خطی در فرم *standard* به صورت زیر نوشته می‌شود:

$$\min\langle c, x\rangle$$
$$Ax = b$$
$$x \geq 0$$

همچنین یک برنامه‌ریزی خطی در فرم *canonical* به صورت زیر نوشته می‌شود:

$$\min\langle c, x\rangle$$
$$Ax \geq b$$
$$x \geq 0$$

فرم‌های متفاوت دیگری نیز وجود دارد، ولی این دو فرم از مهم‌ترین و جالب‌ترین فرم‌ها برای بیان برنامه‌ریزی خطی می‌باشند. یکی از کلی‌ترین فرم‌های ممکن برای بیان برنامه‌ریزی خطی را می‌توان به این صورت تصور کرد، که دارای یک سری نابرابری، برابری، تعدادی متغیر نامنفی و تعدادی متغیر آزاد می‌باشد:

$$\langle a_i, x\rangle = b_i \quad , \quad i \in E$$
$$\langle a_i, x\rangle \geq b_i \quad , \quad i \in I^+$$
$$\langle a_i, x\rangle \leq b_i \quad , \quad i \in I^-$$
$$x_j \geq 0 \quad , \quad j \in N$$

نکته جالب توجه این است که تمام این فرم‌ها از لحاظ قدرت بیان در یک سطح قرار دارند. در واقع می‌توان هر کدام از این فرم‌ها را به دیگری تبدیل کرد. مثلاً چگونه می‌توان یک LP در حالت کلی را به یک LP در حالت $standard$ تبدیل کرد؛

مشاهده .(تبدیل LP از حالت کلی به حالت $standard$) در ابتدا ما برای هر نابرابری $\langle a_i, x\rangle \leq b_i$ یک متغیر کمکی y_i تعریف می‌کنیم و می‌نویسیم:

$$\langle a_i, x\rangle + y_i = b_i \quad , \quad y_i \geq ۰$$

همچنین از آنجایی که $\langle a_i, x\rangle \geq b_i$ برابر با $\langle -a_i, x\rangle \leq -b_i$ می‌باشد، به این ترتیب این نابرابری ها را نیز می‌توان به همین روش به برابری تبدیل کرد. همچنین برای رسیدن به فرم $standard$ لازم است متغیرهای آزاد را حذف کنیم. از آنجایی که هر عدد حقیقی را می‌توان به صورت اختلاف دو عدد نامنفی نوشت، پس به جای هر متغیر آزاد x_j عبارت $x_j^+ - x_j^-$ را قرار می‌دهیم، که $x_j^+, x_j^- \geq ۰$ می‌باشند. به این ترتیب یک LP در حالت کلی به حالت $standard$ تبدیل می‌شود.

مثال ۲-۱ . LP زیر را در نظر بگیرید:

$$\max(x_۱ + ۳x_۲)$$

$$۲x_۱ - x_۲ \geq ۱۰$$

$$x_۲ \geq ۰$$

اگر متغیر کمکی $y_۱$ را به معادله اول اضافه کنیم، و عبارت $x_۱^+ - x_۱^-$ که دو متغیر نامنفی هستند، را به جای متغیر اول در نظر بگیریم، همچنین برای اینکه این سوال را از حالت بیشینه کردن به حالت کمینه کردن بیاوریم، منفی تابع مقدار را کمینه کنیم، در نتیجه فرم $standard$ این LP به صورت زیر در می‌آید:

$$\min(-x_۱^+ + x_۱^- - ۳x_۲)$$

$$۲x_۱^+ - ۲x_۱^- - x_۲ - y۱ = ۱۰$$
$$x_۱^+, x_۱^-, x_۲, y_۱ \geq ۰$$

که $\max\langle c, x\rangle = -\min\langle -c, x\rangle$ می‌باشد.

۲-۲ جواب ممکن مقدماتی

در حال حاضر می‌خواهیم با سیستم LP در حالت $standard$ کار کنیم. معادلات $Ax = b$ را که A یک ماتریس با m سطر و n ستون است در نظر گرفته‌ایم.

رتبه یک ماتریس برابر بعد فضای سطری و همچنین بعد فضای ستونی آن می‌باشد؛ در جبرخطی ثابت می‌شود این دو مقدار با هم برابرند. به عبارت دیگر رتبه ماتریس تعداد سطر(ستون)های مستقل خطی یک ماتریس می‌باشد.

در اینجا نشان می‌دهیم که می‌توانیم فرض کنیم ماتریس A دارای رتبه پر سطری است، که به این ترتیب $m \leq n$. همچنین می‌توانیم بگوییم که $m = \text{rank}(A) \leq n$.

مثال ۲-۲ . سیستم برابری‌های زیر را در نظر بگیرید:

$$x_۱ + x_۲ = ۵$$
$$۲x_۲ + x_۳ = ۸$$
$$۳x_۱ + ۵x_۲ + x_۳ = ?$$

اگر مقدار گذاشته نشده برابر ۲۳ باشد، برابری سوم به درد نمی‌خورد، زیرا از ترکیب دو برابری اول بدست می‌آید. اگر این مقدار برابر ۲۳ نباشد سیستم جواب ندارد.

در اینجا A^i را به معنی سطر i ام ماتریس A و A_j را به معنی ستون j ام A تعریف می‌کنیم.

حال فرضی را که در ابتدای این بخش درباره A کریم، ثابت می‌کنیم. فرض کنید A^i نسبت به سطرهای دیگر وابسته‌ی خطی باشد. در نتیجه می‌توان نوشت $A^i = \sum_{j\neq i} \lambda_j A^j$، که در اینصورت برای هر جواب x برای دستگاه معادلات $Ax = b$، داریم:

$$\begin{aligned} b_i = \langle A^i, x\rangle &= \langle \sum_{j\neq i} \lambda_j A^j, x\rangle \\ &= \sum \lambda_j \langle A^j, x\rangle = \sum \lambda_j b_j \end{aligned}$$

با توجه به گفته‌های بالا اگر $b_i = \sum \lambda_j b_j$ در اینصورت معادله i ام از معادلات دیگر بدست می‌آید و بی‌فایده است. در غیر اینصورت، هیچ x ای در سیستم معادلات صدق نمی‌کند. هر دو این حالات به راحتی در روش حذف گاوسی قابل مشاهده است. در نتیجه در حالت اول این سطر می‌تواند از ماتریس حذف شود، و در حالت دوم نیز سیستم معادلات جواب ندارد. در نتیجه این گفته‌ها، می‌توانیم فرض کنیم سطرهای ماتریس A مستقل خطی می‌باشند.

با این فرض، بیایید با حالت خاصی که در آن $m = n$ است کار را شروع کنیم. در این حالت چون $\text{rand}(A) = m = n$، A یک ماتریس وارون‌پذیر می‌باشد و در نتیجه سیستم معادلات $Ax = b$ دارای جواب یکتای $x = A^{-1}b$ می‌باشد. اگر $x \geq 0$ در نتیجه این مقدار، تنها جواب برنامه‌ریزی خطی مورد نظر می‌باشد.

ما این مشاهده ساده را استفاده کردیم که مشخصه‌ی یک مجموعه از جواب‌های ممکن که یک نقش مهم را در تصمیم‌گیری‌های آینده ایفا می‌کند، ببینیم. قبل از آنکه جلوتر برویم، به تعریف زیر نیازمندیم:

تعریف ۲-۲ . مجموعه $P = \{x | x \geq 0 \ , \ Ax = b\}$ را به عنوان مجموعه جواب‌های ممکن در نظر می گیریم.

یک جواب برای سیستم معادلات $Ax = b$، برابری $\sum_j x_j A_j = b$ را برقرار می‌کند. در واقع، هر ترکیب خطی از ستون‌های A که برابر بردار b شود، یک جواب برای سیستم معادلات می‌باشد. مجموعه‌های m عضوی $I \subset \{1, \ldots, n\}$ از ستون‌های A را که مستقل

خطی باشند، در نظر می‌گیریم. از آنجایی که $\text{rand}(A) = m$، حداقل یک چنین مجموعه‌ای وجود دارد. برای هر I بردار یکتای x وجود دارد که برای اندیس‌هایی که عضو I نیستند، ۰ باشد و در سیستم معادلات $Ax = b$ صدق کند. در واقع، اگر A_I را برابر ماتریسی متشکل از ستون‌های مجموعه I از A، و x_I را برداری متشکل از اندیس‌های مجموعه I از x در نظر بگیریم، در این صورت بردار x را به این صورت تعریف می‌کنیم که $x_I = A_I^{-۱}b$ و $x_{\overline{I}} = ۰$ (در نظر داشته باشید که طبق تعریف، A_I یک ماتریس مربعی وارون‌پذیر می‌باشد).

تعریف ۲-۳ . یک بردار x که به صورت بالا تعریف شده باشد یک جواب مقدماتی منتسب به مجموعه I می‌باشد. اگر $x \geq ۰$، در این صورت $x \in P$ و به آن یک جواب ممکن مقدماتی می‌گوییم.

در اینجا به مثال یک تابع که جواب ممکن مقدماتی می‌سازد، می‌پردازیم.

مثال ۲-۳ . LP زیر را در نظر بگیرید:

$$\min\langle x, (-۱, ۱, ۰)^T\rangle$$
$$Ax = b$$

$$A = \begin{pmatrix} ۱ & ۲ & ۰ \\ ۱ & ۲ & ۱ \end{pmatrix} \quad , \quad b = \begin{pmatrix} ۴ \\ ۷ \end{pmatrix}$$

در ابتدا قرار می‌دهیم $I_۱ = \{۱, ۳\}$، در نتیجه:

$$A_{I_۱} = \begin{pmatrix} ۱ & ۰ \\ ۱ & ۱ \end{pmatrix}$$

پس داریم:

$$\begin{pmatrix} ۱ & ۰ \\ ۱ & ۱ \end{pmatrix} \begin{pmatrix} x_۱ \\ x_۳ \end{pmatrix} = \begin{pmatrix} ۴ \\ ۷ \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} x_۱ \\ x_۳ \end{pmatrix} = \begin{pmatrix} ۴ \\ ۳ \end{pmatrix} \Rightarrow x = \begin{pmatrix} ۴ \\ ۰ \\ ۳ \end{pmatrix}$$

سپس قرار می‌دهیم $I_۲ = \{۲, ۳\}$، در نتیجه:

$$A_{I_۲} = \begin{pmatrix} ۲ & ۰ \\ ۲ & ۱ \end{pmatrix}$$

پس داریم:

$$\begin{pmatrix} ۲ & ۰ \\ ۲ & ۱ \end{pmatrix} \begin{pmatrix} x_۲ \\ x_۳ \end{pmatrix} = \begin{pmatrix} ۴ \\ ۷ \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} x_۲ \\ x_۳ \end{pmatrix} = \begin{pmatrix} ۲ \\ ۳ \end{pmatrix} \Rightarrow x = \begin{pmatrix} ۰ \\ ۲ \\ ۳ \end{pmatrix}$$

دقت کنید مجموعه $I_۳ = \{۱, ۲\}$ را نمی‌توان انتخاب کرد، زیر $A_۱$ و $A_۲$ مستقل خطی نمی‌باشند. در نتیجه برای این سوال دو جواب مقدماتی وجود داشت که هر دو جواب ممکن مقدماتی بودند.

برای روشن‌تر شدن مفهوم bfs (جواب ممکن مقدماتی)، ما به شرح لم زیر می‌پردازیم. این لم در ادامه توضیحات اثبات خواهد شد.

لم ۲-۱ . اگر یک LP در فرم $standard$ دارای یک جواب بهینه باشد، این LP دارای جواب بهینه‌ای است که جواب ممکن مقدماتی می‌باشد.

لم بالا چه چیز به ما می‌گوید؟ با وجود دامنه نامتناهی که سوال می‌تواند داشته باشد، ما می‌توانیم توجه خود را به یک مجموعه متناهی و خاص، یعنی bfs ها محدود کنیم. در این صورت، یک الگوریتم brute force وجود دارد که در صورت وجود جواب بهینه، آن را برای LP پیدا می‌کند.

الگوریتم $(Brute\ force)$

ورودی: A, b, c.
مقداردهی اولیه : $Z = \infty$.
برای هر زیرمجموعه m عضوی $I \subset \{1, \ldots, n\}$ به صورت زیر عمل کن:
اگر A_I معکوس‌پذیر بود و $A_I^{-1}b \geq 0$ در این صورت، فرض کنید x، bfs مربوط به I باشد، یعنی $x_I = A_I^{-1}b$ و $x_{\bar{I}} = 0$.
اگر $\langle x, c\rangle < Z$ در این صورت قرار دهید $Z = \langle x, c\rangle$ و $x_{best} = x$.

خروجی : x_{best} به عنوان بهترین جواب مسئله.

وقتی زمان اجرای این الگوریتم را محاسبه می‌کنیم، می‌بینیم که تعداد زیادی تکرار داریم، که مثلاً برای $m = \frac{n}{2}$ یک مقدار نمایی بر حسب n می‌شود.

تعریف ۲-۴ . به نقطه $x \in P$، یک نقطه حد[1] برای P گفته می‌شود، اگر میانگین وزنی دو نقطه $y, z \in P$ نباشد. نقطه x میانگین وزنی دو نقطه y, z است اگر یک ترکیب محدب به صورت $\lambda y + (1 - \lambda)z$ از این دو نقطه باشد.

برای مثال نقاط حد در بازه $[0, 1]$ برابر $\{0, 1\}$ می‌باشند. همچنین $\vec{0}$ نقطه حد مجموعه $\{x \in \mathbb{R} | x \geq 0\}$ می‌باشد.

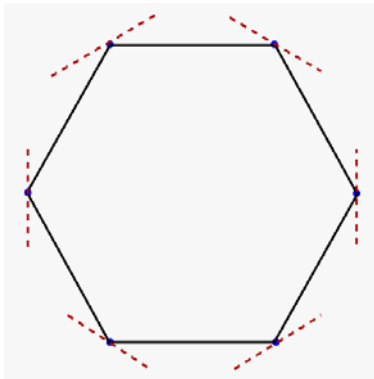
مثال ۲-۴ . مجموعه‌ی $\{x \in \mathbb{R} |\ ||x||_2 \leq 1\}$ را در نظر بگیرید. نقاط حد این مجموعه، مجموعه $\{||x||_2 = 1\}$ می‌باشد.

مثال ۲-۵ . یک چندضلعی طبق شکل ۱ در صفحه ۲–بعدی در نظر بگیرید. نقاط حدی این چندضلعی، رئوس آن می‌باشند.

لم ۲-۲ . x یک bfs است اگر و فقط اگر x یک نقطه حدی باشد.

---

[1] extreme



شکل ۱: رئوس چندضلعی به عنوان رئوس حد؛ همچنین راستاهایی را که در آنها هر کدام از نقاط حد، در آن راستا نسبت به بقیه نقاط کمینه هستند نشان داده شده است.

اثبات. $bfs \Rightarrow extreme$

فرض کنید x یک bfs باشد و میانگین وزنی دو نقطه y و z نیز باشد، یعنی $x = \lambda y + (1-\lambda)z$ که $0 < \lambda < 1$. در نتیجه به ازای هر j داریم $x_j = \lambda y_j + (1-\lambda)z_j$. برای $j \notin I$ داریم:

$$0 = x_j = \lambda y_j + (1-\lambda)z_j$$

ولی $y_j, z_j \geq 0$. در نتیجه برای $j \notin I$، $y_j = z_j = 0$. از آنجایی که y و z در P هستند، داریم:

$$y_I = z_I = A_I^{-1} b = x_I$$

و در نتیجه $x = y = z$. این بدان معناست که x یک نقطه حدی می‌باشد.

$extreme \Rightarrow bfs$

در ابتدا به این نتیجه می‌رسیم که x یک bfs است اگر و فقط اگر $J = \{j | x_j > 0\}$ به یک مجموعه از ستون‌های مستقل خطی از A اشاره کند. اگر این مجموعه وابسته باشد که طبق تعریف به این نتیجه می‌رسیم که x نمی‌تواند bfs باشد و بالعکس اگر این مجموعه از ستون‌ها مستقل باشند، به وضوح می‌توان مجموعه J را به یک مجموعه مستقل از ستون‌های ماتریس A مثل I که $|I| = m$ گسترش داد. در نتیجه x، bfs مربوط به مجموعه

I می‌باشد.

فرض کنید نقطه حد x، bfs نباشد. قرار می‌دهیم $J = \{j|x_j > 0\}$. در این صورت ستون‌های A_J وابسته خطی می‌باشند و در نتیجه یک بردار ناصفر v وجود دارد که در خارج از J، 0 باشد و $Av = 0$. در نتیجه داریم:

$$A(x \overset{+}{-} \lambda v) = Ax \overset{+}{-} \lambda Av = Ax = b$$

به بیان دیگر $x \overset{+}{-} \lambda v$ یک جواب برای سیستم برابر $Ax = b$ می‌باشد. اگر $\lambda > 0$ را به اندازه کافی کوچک در نظر بگیریم، دو بردار $x + \lambda v$ و $x - \lambda v$ نامنفی می‌شوند(زیرا v در خارج از مجموعه J 0 است). پس $x \overset{+}{-} \lambda v \in P$ و در نتیجه داریم:

$$x = \frac{1}{2}\underbrace{(x + \lambda v)}_{=y} + \underbrace{(x - \lambda v)}_{=z}$$

که با فرض نقطه حد بودن x به تناقض می‌رسیم. □

لم ۲-۳ . اگر x یک bfs باشد، در این صورت یک انتخاب برای بردار c وجود دارد به طوری که به ازای هر $y \in P$ داشته باشیم $\langle x, c\rangle < \langle y, c\rangle$.

اثبات. از آنجایی که x یک bfs است، در نتیجه برای $j \notin I$ داریم $x_j = 0$. ما c را به این ترتیب انتخاب می‌کنیم که برای هر $j \in I$ قرار می‌دهیم $c_j = 0$ و برای هر $j \notin I$، قرار می‌دهیم $c_j = 1$. بوضوح $\langle x, c\rangle = 0$ است. اگر $y \neq x$ ، در این صورت $y_{\overline{I}} \neq 0$ می‌باشد (در غیر این‌صورت $y_I = x_I$ و در نتیجه $y = x$) و همچنین از آنجایی که $y_{\overline{I}} \geq 0$ ، در نتیجه $\langle y, c\rangle = \sum_{j \neq I} y_j > 0$ و به این ترتیب $\langle y, c\rangle > \langle x, c\rangle$ است، و لم اثبات می‌شود.

دقت کنید طرف دیگر این لم واضح است. یعنی اگر x یک bfs نباشد، میانگین دو نقطه y و z می‌باشد و در نتیجه برای هیچ برداری، نمی‌تواند از هر دوی آنها کمتر باشد. □

حال ما آماده‌ایم، لم ۵ را ثابت کنیم.

اثبات. فرض کنید جواب بهینه x^* وجود دارد. اگر x^* یک bfs باشد، مسئله حل است. در غیر این صورت ما می‌توانیم یک bfs توسط حلقه زیر پیدا کنیم:

- با یک جواب ابتدایی x° شروع کن.
- قرار بده $J = \{j | x_j^{\circ} > \circ\}$. اگر A_j که $j \in J$، مستقل خطی بود که توقف کن، چون x یک bfs است. در غیر اینصورت، بردار v را طبق لم ۶ به این ترتیب انتخاب کن که $Av = \circ$ و برای هر $x_j = \circ$، $v_j = \circ$.
- عدد k را برابر اندیسی قرار بده که

$$\left|\frac{x_k}{v_k}\right| = \min_{j:v_j \neq \circ} \left|\frac{x_j}{v_j}\right|$$

و $\lambda = -\frac{x_k}{v_k}$ قرار بده.

- $x^{۱} = x^{\circ} + \lambda v$ قرار بده. به راحتی می‌توانیم ببینیم که

$$x^{۱} \geq \circ \ , \ x_k^{۱} = \circ \ , \ x_j^{۱} = \circ (j \notin J)$$

و به این کار ادامه می‌دهیم.

این حلقه وقتی به پایان می‌رسد که، J به مجموعه‌ای از ستون‌های مستقل در A اشاره کند (به یاد داشته باشید که ما می‌توانیم با $J = \{\}$ که در این صورت $x = \circ \in P$ یک bfs می‌باشد، کار را پایان دهیم). اثبات را با نشان دادن اینکه bfs بدست آمده به اندازه خود جواب بهینه، یعنی x^* خوب است، پایان می‌دهیم. در هر تکرار حلقه در تابع بالا، تابع مقدار به اندازه $\langle c, \lambda v \rangle$ تغییر می‌کند، که ثابت می‌کنیم این مقدار، برابر $\circ$ است.

فرض کنید x^* یک جواب بهینه باشد. برای $\lambda > \circ$ به مقدار لازم کوچک، $x \overset{+}{-} \lambda v \in P$ می‌باشد و

$$\langle c, x \overset{+}{-} \lambda v \rangle = \langle c, x \rangle \overset{+}{-} \lambda \langle c, v \rangle$$

اگر قرار دهیم $p = \lambda \langle c, v \rangle$، دو جواب $\langle c, x \rangle \overset{+}{-} p$ وجود دارد؛ در نتیجه اگر p صفر نباشد، یکی از این دو مقدار جواب بهتری تولید می‌کند که با بهینه بودن x^* در تناقض است. □

در اینجا ما با چند مثال جریان بدست آوردن یک bfs را نشان می‌دهیم.

مثال ۲-۶ . سیستم معادلات $Ax = b$ را که در آن

$$A = \begin{pmatrix} ۱ & ۰ & ۱ & ۲ & ۳ \\ ۱ & ۱ & ۲ & ۳ & ۵ \\ ۰ & ۱ & ۱ & ۱ & ۸ \end{pmatrix} \ , \ b = \begin{pmatrix} ۹ \\ ۱۸ \\ ۹ \end{pmatrix} \ , \ x^{۰} = \begin{pmatrix} ۱ \\ ۳ \\ ۴ \\ ۲ \\ ۰ \end{pmatrix}$$

است، در نظر بگیرید. bfs را پیدا کنید.

در مرحله اول $J = \{۱, ۲, ۳, ۴\}$ است. A_J مستقل خطی نمی‌باشد و در نتیجه بردار $v = (۱, ۱, -۱, ۰, ۰)^T$ را که $Av = ۰$ و $v_۵ = ۰$ در نظر می‌گیریم. در ادامه عدد λ را با مقایسه مقدار $\left|\frac{x_j}{v_j}\right|$ که در آن $v_j \neq ۰$ انتخاب می‌کنیم:

$$\left|\frac{x_۱}{v_۱}\right| = \left|\frac{۱}{۱}\right| \quad \left|\frac{x_۲}{v_۲}\right| = \left|\frac{۳}{۱}\right| \quad \left|\frac{x_۳}{v_۳}\right| = \left|\frac{۴}{-۱}\right|$$

در نتیجه مقدار مینیمم برای $j = ۱$ است، پس $\lambda = -۱$ می‌باشد. قرار می‌دهیم

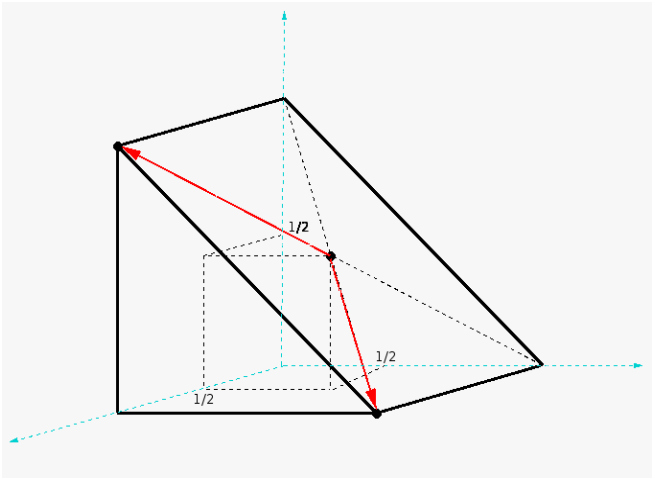
$$x^{۱} = x^{۰} + \lambda v = \begin{pmatrix} ۱ \\ ۳ \\ ۴ \\ ۲ \\ ۰ \end{pmatrix} + (-۱) \begin{pmatrix} ۱ \\ ۱ \\ -۱ \\ ۰ \\ ۰ \end{pmatrix} = \begin{pmatrix} ۰ \\ ۲ \\ ۵ \\ ۲ \\ ۰ \end{pmatrix}$$

در مرحله دوم $J = \{۲, ۳, ۴\}$ است. A_J مستقل خطی نمی‌باشد و در نتیجه بردار $v = (۰, -۱, ۲, -۱, ۰)^T$ را که $Av = ۰$ و $v_۱, v_۵ = ۰$ در نظر می‌گیریم. در ادامه عدد λ را با مقایسه مقدار $\left|\frac{x_j}{v_j}\right|$ که $v_j \neq ۰$ انتخاب می‌کنیم:

$$\left|\frac{x_۲}{v_۲}\right| = \left|\frac{۲}{-۱}\right| \quad \left|\frac{x_۳}{v_۳}\right| = \left|\frac{۵}{۲}\right| \quad \left|\frac{x_۴}{v_۴}\right| = \left|\frac{۲}{-۱}\right|$$

در نتیجه مقدار مینیمم برای $j = ۲, ۴$ است، پس $\lambda = ۲$ می‌باشد. قرار می‌دهیم

$$x^{۲} = x^{۱} + \lambda v = \begin{pmatrix} ۰ \\ ۲ \\ ۵ \\ ۲ \\ ۰ \end{pmatrix} + ۲ \begin{pmatrix} ۰ \\ -۱ \\ ۲ \\ -۱ \\ ۰ \end{pmatrix} = \begin{pmatrix} ۰ \\ ۰ \\ ۹ \\ ۰ \\ ۰ \end{pmatrix}$$



شکل ۲: پیدا شدن bfs بهینه از جواب بهینه

بنابراین با یک ستون A، یعنی ستون سوم تنها ماندیم که در آن $x_۳ \neq ۰$ می‌باشد. این بردار تنها بوضوح مستقل خطی است و در نتیجه جواب ما $x^۲ = (۰, ۰, ۹, ۰, ۰)^T$ است که یک bfs می‌باشد.

مثال ۲-۷ . برای بدست آمدن یک شهود هندسی نسبت به پیدا شدن bfs به این مثال می‌پردازیم. برنامه‌ریزی خطی زیر را در نظر بگیرید:

$$\max(x_۱ + x_۲)$$

$$x_۳ \leq ۱$$

$$x_۱ + x_۲ \leq ۱$$

$$x_۱, x_۲, x_۳ \geq ۰$$

یک جواب بهینه برای این برنامه‌ریزی خطی بردار $x^* = (\frac{۱}{۲}, \frac{۱}{۲}, \frac{۱}{۲})$ می‌باشد که در شکل ۲ نشان داده شده است. فرم $standard$ این برنامه‌ریزی خطی به صورت زیر می‌باشد:

$$\max(x_۱ + x_۲)$$

$$x_۳ + y_۱ = ۱$$
$$x_۱ + x_۲ + y_۲ = ۱$$
$$x_۱, x_۲, x_۳, y_۱, y_۲ \geq ۰$$

در این صورت داریم:

$$A = \begin{pmatrix} ۰ & ۰ & ۱ & ۱ & ۰ \\ ۱ & ۱ & ۰ & ۰ & ۱ \end{pmatrix} \ , \ b = \begin{pmatrix} ۱ \\ ۱ \end{pmatrix} \ , \ x^۰ = \begin{pmatrix} \frac{۱}{۲} \\ \frac{۱}{۲} \\ \frac{۱}{۲} \\ ۰ \\ ۰ \end{pmatrix}$$

در مرحله اول $J = \{۱, ۲, ۳\}$ است. A_J مستقل خطی نمی‌باشد و در نتیجه بردار $v = (۱, -۱, ۰, ۰, ۰)^T$ را که $Av = ۰$ و $v_۴, v_۵ = ۰$ در نظر می‌گیریم. در ادامه عدد λ را با مقایسه مقدار $\left|\frac{x_j}{v_j}\right|$ که در آن $v_j \neq ۰$ انتخاب می‌کنیم:

$$\left|\frac{x_۱}{v_۱}\right| = \left|\frac{\frac{۱}{۲}}{۱}\right| \quad \left|\frac{x_۲}{v_۲}\right| = \left|\frac{\frac{۱}{۲}}{-۱}\right|$$

در نتیجه مقدار مینیمم برای $j = ۱$ با $\lambda = -\frac{۱}{۲}$ و یا $j = ۲$ با $\lambda = \frac{۱}{۲}$ می‌باشد. قرار می‌دهیم

$$x_۰^۱ = x^۰ + \lambda v = \begin{pmatrix} \frac{۱}{۲} \\ \frac{۱}{۲} \\ \frac{۱}{۲} \\ ۰ \\ ۰ \end{pmatrix} + (-\frac{۱}{۲}) \begin{pmatrix} ۱ \\ -۱ \\ ۰ \\ ۰ \\ ۰ \end{pmatrix} = \begin{pmatrix} ۰ \\ ۱ \\ \frac{۱}{۲} \\ ۰ \\ ۰ \end{pmatrix}$$

$$x_۱^۱ = x^۰ + \lambda v = \begin{pmatrix} \frac{۱}{۲} \\ \frac{۱}{۲} \\ \frac{۱}{۲} \\ ۰ \\ ۰ \end{pmatrix} + (\frac{۱}{۲}) \begin{pmatrix} ۱ \\ -۱ \\ ۰ \\ ۰ \\ ۰ \end{pmatrix} = \begin{pmatrix} ۱ \\ ۰ \\ \frac{۱}{۲} \\ ۰ \\ ۰ \end{pmatrix}$$

که هر دو آنها bfs می‌باشند. در شکل ۲ حرکت از نقطه $x_۰$ به این دو نقطه نشان داده شده است.

فصل ۳

جلوه‌های هندسی برنامه‌ریزی خطی

در این بخش قصد داریم کمی درباره‌ی وجوه هندسی برنامه‌ریزی خطی صحبت کنیم تا شهودی بهتر نسبت به برنامه‌ریزی خطی بدست آوریم.

یادآوری

در بخش‌های قبلی مسیر مهمی را برای تعریف یک الگوریتم مفید برای حل برنامه‌ریزی خطی طی کردیم.

- تعریفی برای جواب ممکن مقدماتی (bfs) دادیم.
- ثابت کردیم که نقاط bfs و حدی برابر هستند.
- نشان دادیم که اگر جواب بهینه‌ای برای LP وجود داشته باشد، در اینصورت یک bfs وجود دارد که مقدار بهینه را داشته باشد.

-- این نکته به ما یک الگوریتم نمایی که هر مجموعه m تایی مستقل از ستون‌های A را چک می‌کرد نشان داد.

-- و یک نتیجه واضح این است که اگر یک جواب وجود داشته باشد، یعنی دامنه‌ی برنامه‌ریزی خطی ناتهی باشد ($P \neq \emptyset$)، در این صورت یک جواب bfs وجود دارد.

- در آخر نیز ثابت کردیم که اگر $\vec{x}$ یک bfs باشد، در اینصورت بردار $\vec{c}$ وجود دارد به طوری که:

$$\langle \vec{x}, \vec{c} \rangle < \langle \vec{y}, \vec{c} \rangle \quad \forall\, \vec{y} \in P\,, \quad \vec{y} \neq \vec{x}$$

این بدان معنی است که می‌توان یک تابع مقداری را انتخاب کرد که دقیقاً یک bfs به طور یکتا، جواب بهینه شود.

۳-۱ قضیه‌ی $Caratheodory$، کاربردی از برنامه‌ریزی خطی

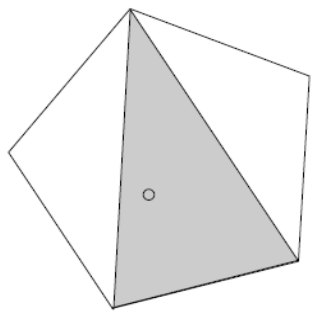
این بخش را با بیان و اثبات قضیه $Caratheodory$ شروع می‌کنیم. در این بخش می‌بینیم که چگونه می‌توان از برنامه‌ریزی خطی برای حل مسائل تئوری کمک جست.

قضیه ۳-۱. $Caratheodory$ فرض کنید $\vec{b}, \vec{v_1}, \ldots, \vec{v_t}$ بردارهایی در $\mathbb{R}^n$ باشند و b ترکیب محدب $\vec{v_1}, \ldots, \vec{v_t}$ باشد.

$$\vec{b} = \sum_{i=1}^{t} x_i\, \vec{v_i}\,, \quad x_i \geq 0\,, \quad \sum_{i=1}^{t} x_i = 1$$

اگر $\vec{b}$ یک ترکیب محدب از $\vec{v_i}$ ها باشد، در این صورت این بردار را می‌توان به صورت ترکیب محدبی از حداکثر $n+1$ بردار $\vec{v_i}$ نشان داد. به بیان دیگر، مجموعه‌ی حداکثر $n+1$ عضوی I وجود دارد بطوری که

$$\vec{b} = \sum_{i \in I} \alpha_i\, \vec{v_i}\,, \quad \alpha_i \geq 0\,, \quad \sum_{i \in I} \alpha_i = 1$$



شکل ۱: مثالی از قضیه‌ی $Caratheodory$

اثبات. برای اثبات این قضیه یک برنامه خطی برای پیدا کردن α_i به صورت زیر می‌نویسیم:

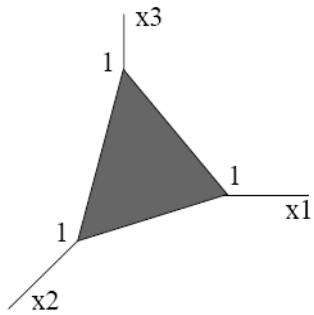
$$\begin{pmatrix} \vec{v_1} & \dots & \vec{v_t} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_t \end{pmatrix} = \vec{b} \ , \ \ x_i \geq 0 \ , \ \ \sum_{i=1}^{t} x_i = 1$$

سپس برابری $\sum_i x_i = 1$ را با اضافه کردن یک سطر ۱ مثل زیر به سیستم برابری‌ها اضافه می‌کنیم:

$$\begin{pmatrix} \vec{v_1} & \dots & \vec{v_t} \\ 1 & \dots & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_t \end{pmatrix} = \begin{pmatrix} \vec{b} \\ 1 \end{pmatrix} \ , \ \ x_i \geq 0$$

طبق فرض قضیه، یک جواب برای این برنامه‌ریزی خطی وجود دارد. طبق گفته‌های فصل قبل از آنجایی که دامنه برنامه‌ریزی خطی ناتهی است، یک bfs برای این سیستم وجود دارد. از آنجایی که $n+1$ معادله داریم، یک جواب $(x_1, \dots, x_t)^T \in P$ که حداکثر $n+1$ عنصر ناصفر دارد، وجود دارد. ما اعضای این جواب را به عنوان α_i ها در نظر می‌گیریم و قضیه ثابت می‌شود. □

مثال ۳-۱ .(مثال هندسی از قضیه $Caratheodory$) در شکل ٦، **یک مثال هندسی از قضیه $Caratheodory$ نشان داده شده است. در این شکل یک ۵ ضلعی منتظم است که پوش محدب رئوس خود می‌باشد. در هر حال، نقطه‌ای که در داخل این ۵ ضلعی وجود دارد می‌تواند به صورت ترکیب محدب فقط ۳ تا از رئوس این ۵ ضلعی نشان داده شود. در مثال**



شکل ۲: جواب‌های ممکن برای برنامه‌ریزی خطی

نشان داده شده نقطه داخلی با یک دایره کوچک مشخص شده است که در مثلث سایه زده شده که توسط ۳ تا از رئوس این ۵ ضلعی ساخته شده است قرار دارد.

۳-۲ جلوه‌های هندسی برنامه‌ریزی خطی

برنامه‌ریزی خطی زیر را در نظر بگیرید:

$$x_۱ + x_۲ + x_۳ = ۱$$

$$x_۱, x_۲, x_۳ \geq ۰$$

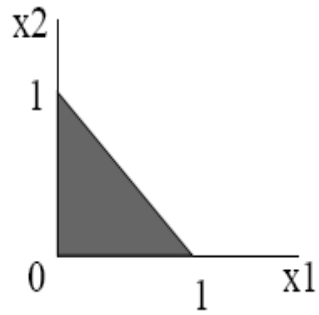
مجموعه جواب‌ها برای برنامه‌ریزی خطی بالا در شکل ۲ به صورت یک مثلث نشان داده شده. ما می‌توانیم بعد این مسئله را با حذف کردن متغیر اضافی $x_۳$ به صورت زیر کم کنیم:

$$x_۱ + x_۲ \leq ۱$$

$$x_۱, x_۲ \geq ۰$$

در این صورت برای این برنامه‌ریزی خطی ما شکل ۳ را داریم که در ۲ بعد دامنه این برنامه‌ریزی خطی را کشیده است.

تعریف ۳-۱ . یک ابرصفحه H یک مجموعه $\{\vec{x} \mid \langle \vec{x}, \vec{a} \rangle = \alpha\}$ می‌باشد که در آن $\vec{a}$، برداری ناصفر و عضو $\mathbb{R}^n$ می‌باشد و $\alpha \in R$ است.



شکل ۳: جواب‌های ممکن برای برنامه‌ریزی خطی

تعریف ۳-۲ . یک نیم صفحه H^+ یک مجموعه $\{\vec{x} \mid \langle \vec{x}, \vec{a} \rangle \geq \alpha\}$ می‌باشد که در آن $\vec{a}$، برداری ناصفر و عضو $\mathbb{R}^n$ می‌باشد و $\alpha \in R$ است. به همین ترتیب نیم صفحه H^- یک مجموعه $\{\vec{x} \mid \langle \vec{x}, \vec{a} \rangle \leq \alpha\}$ می‌باشد. در نظر داشته باشید که برخورد دو نیم صفحه H^+ و H^- یک ابرصفحه H است ($H^+ \cap H^- = H$).

مشاهده . اگر ما یک برنامه‌ریزی خطی را در فرم *standard* داشته باشیم، آنگاه مجموعه جواب‌های ممکن این برنامه‌ریزی خطی برابر می‌شود با:

$$P = \bigcap_{j=1}^{n} \{x_j \geq 0\} \bigcap \left(\bigcap_{i=1}^{m} \{\langle \vec{a_i}, \vec{x} \rangle = b_i\} \right)$$

در نتیجه مجموعه جواب‌های ممکن این برنامه‌ریزی خطی به صورت برخورد یک سری نیم صفحه و یک سری ابرصفحه می‌باشد. از آنجایی که ما می‌توانیم ابرصفحه‌ها را به صورت برخورد نیم صفحه‌ها بنویسیم، در نتیجه این برنامه‌ریزی خطی می‌تواند به صورت برخورد یک سری نیم صفحه به صورت زیر نوشته شود:

$$P = \bigcap_{j=1}^{n} \{x_j \geq 0\} \bigcap \left(\bigcap_{i=1}^{m} \{\langle \vec{a_i}, \vec{x} \rangle \geq b_i\} \right) \bigcap \left(\bigcap_{i=1}^{m} \{\langle \vec{a_i}, \vec{x} \rangle \leq b_i\} \right)$$

از آنجایی که نیم صفحات محدب هستند، و برخورد یک سری مجموعه محدب، محدب می‌شود، پس فضای جواب‌های ممکن ، محدب است. برعکس، هر فضای محدب نیز می‌تواند به صورت برخورد یک سری (ولی نه لزوماً متناهی)، نیم صفحه بوجود آید.

تعریف ۳-۳ . یک *polyhedron* یک مجموعه متناهی از برخورد نیم صفحات می‌باشد. همچنین یک *polyhedron* که محدود شده (در یک کره n بعدی جا شود) را یک *polytope* می گویند.

از آنجایی که جواب‌های ممکن یک مجموعه متناهی از برخورد نیم صفحات می‌باشند، پس *polyhedron* می‌باشند.

تعریف ۳-۴ . H یک ابرصفحه‌ی محافظ برای P می‌باشد اگر $H \cap P \neq \emptyset$ و P در یک سمت H قرار بگیرد، یعنی $P \subset H^-$ و یا $P \subset H^+$.

تعریف ۳-۵ . محل برخورد یک ابرصفحه‌ی محافظ H با P را یک وجه می گویند. همچنین $facet$ به یک وجه ماکسیمال از P گفته می‌شود.

تعریف ۳-۶ . بعد یک مجموعه برابر بعد کوچکترین فضایی که شامل آن مجموعه است، می‌باشد.

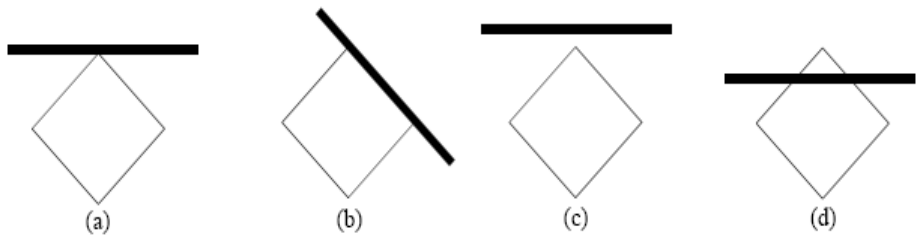
برای مثال مجموعه جواب‌های ممکن در شکل ۲ و ۳ دارای بعد ۲ می‌باشند.

مثال ۳-۲ . (مثال‌هایی از وجه‌های *polyhedron* و بعد آن‌ها)

یک راس از *polyhedron* یک وجه با بعد ۰ است.

یک یال از *polyhedron* یک وجه با بعد ۱ است.

یک $facet$ از *polyhedron* یک وجه با بعد $d-1$ می‌باشد که d بعد *polyhedron* می‌باشد.



شکل ۴: مثال تصویری نیم‌صفحه و ابرصفحه‌ی محافظ

مثال ۳-۳ . (**مثال گرافیکی یک** ابرصفحه‌ی محافظ) در شکل ۴ **ما چهار نیم‌صفحه کشیده‌ایم، ۲ نیم‌صفحه‌ی قسمت** (a) **و** (b) ابرصفحه‌ی محافظ **می‌باشند ولی نیم‌صفحه‌های** (c) **و** (d) ابرصفحه‌ی محافظ **نمی‌باشند. همچنین محل برخورد نیم‌صفحه در** (a) **یک راس می‌باشد که ۰ بعدی است و در** (b) **یک یال می‌باشد که ۱ بعدی است.**

لم ۳-۱ . vertex فرض کنید $P = \{x \geq 0 | Ax = b\}$ باشد. x یک راس از P است اگر و تنها اگر x یک bfs باشد. همچنین x یک راس از P است اگر و تنها اگر x یک نقطه حدی باشد.

قبل از اینکه به اثبات این قضیه بپردازیم، در نظر داشته باشید که چون راس‌ها برابر با bfs ها و نقاط حدی می‌باشند، کافی است برای پیدا کردن جواب بهینه به رئوس نگاه کنیم.

اثبات. در اینجا ثابت می‌کنیم که یک bfs، یک راس است. به خاطر آورید که اگر $\vec{x}$ یک bfs باشد، در این صورت بردار ناصفر $\vec{c}$ وجود دارد، به طوری که برای هر $\vec{y} \in P$ داش‌ته باشیم $\langle \vec{x}, \vec{c} \rangle < \langle \vec{y}, \vec{c} \rangle$. ما نیم‌صفحه H را به صورت زیر تعریف می‌کنیم:

$$H = \{\vec{v} \,|\langle \vec{v}, \vec{c} \rangle = \alpha\} \ , \ \ \alpha = \langle \vec{x}, \vec{c} \rangle$$

از آنجایی که برای هر $\vec{z} \in P$، $\langle \vec{y}, \vec{c} \rangle > \alpha$ می‌باشد، در نتیجه $H \cap P = \{x\}$ می‌باشد. همچنین از آنجایی که برای هر $\vec{y} \in P$، $\langle \vec{y}, \vec{c} \rangle > \alpha$ است، نتیجه می‌گیریم که $\vec{z} \in H^+$ و

چون $P \subset H^+$ است، در نتیجه H یک ابرصفحه‌ی محافظ برای P می‌باشد و محل برخورد آن با P راس x است. در نتیجه x یک راس از P می‌باشد. □

متذکر می‌شویم که قبلاً ثابت کرده بودیم bfs ها و نقاط حدی هم‌ارز می‌باشند. اثبات هم‌ارزی بین راس‌ها و نقاط حدی اثبات بالا را کامل می‌کند و به عنوان یک تمرین به خواننده واگذار می‌شود.

تعریف ۳-۷ . یک bfs به مجموعه‌ی m عضوی I از اندیس‌ها منتسب می‌شود که به m ستون مستقل خطی از A اشاره می‌کند. به دو bfs متفاوت همسایه گفته می‌شود، اگر I و I' مجموعه مشخصه‌ی این دو bfs باشد و $I' = I - \{i_۱\} \cup \{i_۲\}$، که $i_۱$ و $i_۲$ دو اندیس متفاوت در $\{۱, \ldots, n\}$ می‌باشند.

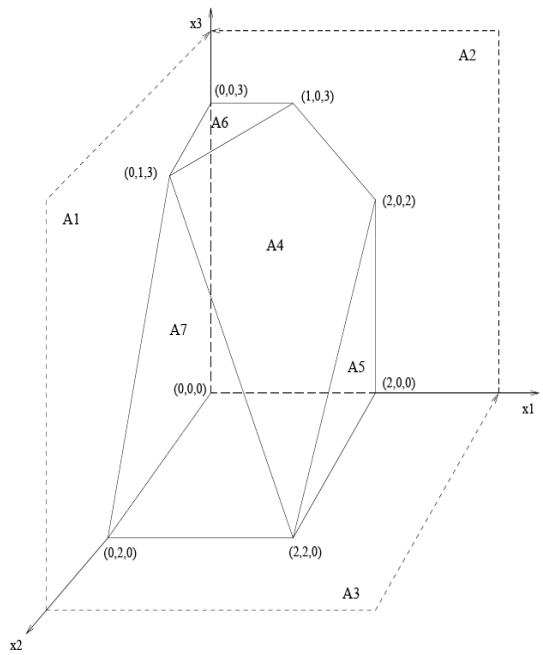
در لم قبل ما رابطه بین bfs ها و رئوس را مطالعه کردیم. با تعریف ارائه شده برای همسایه یک bfs، ما اتصال هندسی این همسایه‌ها را می‌توانیم مشاهده کنیم. در اینجا، تعویض یک اندیس معادل است با حرکت دادن یک راس به راس دیگر بر روی یک یال از یک *polytope* . در واقع اگر از یک bfs به یکی از bfs های همسایه‌ی آن برویم، روی یکی از یال‌های *polytope* حرکت می‌کنیم. اثبات برای این حرف‌ها را می‌توانید در *Papadimitriou* و *Steiglitz* مشاهده کنید.

۳-۳ خلاصه و یک مثال ویژه

۳-۳-۱ نمای بیرونی یک برنامه خطی

ما خلاصه را با در نظر گرفتن یک برنامه‌ریزی خطی که مقدار تابع مقدار $f(x) = ۲x_۲ + x_۴ + ۵x_۷$ را مینیمم می‌کند، شروع می‌کنیم:

$$\begin{array}{lllllllll}
x_۱ & +x_۲ & +x_۳ & +x_۴ & & & & & = ۴ \\
x_۱ & & & & +x_۵ & & & & = ۲ \\
 & & +x_۳ & & & +x_۶ & & & = ۳ \\
 & +۳x_۲ & +x_۳ & & & & +x_۷ & & = ٦
\end{array}$$



شکل ۵: *polytope*

و $x_i \geq ۰$ می‌باشد. اولین کار ما تبدیل این برنامه‌ریزی خطی به فرم *standard* می‌باشد. در نظر داشته باشید که این یک مسئله است که $x \in R^n$ و با m شرط برابری است. این برابری‌ها می‌توانند به صورت $A\ \vec{x}=\vec{b}$ در یک ماتریس بنشینند. اگر متغیرهای $x_۴$ تا $x_۷$ را حذف کنیم این LP به فرم زیر در می‌آید که می‌توان از A_i به معادله i ام نام برد :

$$\begin{array}{c|cccc}
A_۴ & x_۱ & +x_۲ & +x_۳ & \leq ۴ \\
A_۵ & x_۱ & & & \leq ۲ \\
A_۶ & & & +x_۳ & \leq ۳ \\
A_۷ & & +۳x_۲ & +x_۳ & \leq ٦ \\
A_۱ & x_۱ & & & \geq ۰ \\
A_۲ & & +x_۲ & & \geq ۰ \\
A_۳ & & & +x_۳ & \geq ۰
\end{array}$$

دقت کنید که این سوال در فرم *canonical* بسیار ساده‌تر و نشان دادن دید هندسی آن راحت‌تر می‌باشد که این کار توسط حذف متغیرهای کمکی آن محقق شده است. همچنین وقتی یک برنامه‌ریزی خطی را در فرم *standard* می‌نویسیم با یک فضای برداری n بعدی

سروکار داریم و هنگامی برنامه‌ریزی خطی را در فرم *canonical* می‌نویسیم، با یک فضای برداری $n-m$ بعدی سروکار داریم که m تعداد معادلات در فرم *standard* می‌باشد.

۳-۳-۲ معنای هندسی

همانطور که دیدیم هر نابرابر یک نیم فضا را نشان می‌دهد. همچنین برخورد n نیم فضا، فضای جواب‌های ممکن P را ایجاد می‌کند. در نهایت چون نیم‌فضاها محدب هستند، برخورد آنها نیز محدب است و یک *polytope* را در R^{n-m} ایجاد می‌کنند. در شکل ۵ دیده می‌شود که ما ۷ وجه علامت‌گذاری شده با $A_1, \ldots, A_7$ داریم. به یاد آورید که یک bfs یک بردار $\vec{x}$ است به طوری که دارای حداقل $n-m$ مقدار ۰ باشد (در فرم *standard*). از آنجایی که ۰ در مکان i ام در فرم *standard* به برابری A_i (طبق نامگذاری مثال فوق) در فرم *canonical* اشاره می‌کند، یک bfs به محل برخورد $n-m$ ابرصفحه (بخاطر $n-m$ مقدار ۰ در $\vec{x}$) اشاره می‌کند. چون ابرصفحه‌ای که ما در فرم *canonical* ساخته‌ایم در R^{n-m} می‌باشد، و چون این ابرصفحه‌ها وجه‌های این *polytope* هستند، محل برخورد این $n-m$ ابرصفحه یک راس را برای *polytope* مشخص می‌کند.

یک bfs با پایه مشخصه $I = \{A_1, A_2, A_3, A_6\}$ در مثال فوق در نظر بگیرید. این بدین معنی است که x_4, x_5, x_7 در فرم *standard* صفر می‌باشند. در شکل ۵ می‌توانیم ببینیم که برخورد صفحات A_4, A_5, A_7، نقطه $(2, 2, 0)$ می‌باشد. با قرار دادن این مقادیر در فرم *standard* ما یک bfs، $\vec{x} = (2, 2, 0, 0, 0, 3, 0)$ را بدست می آوریم.

فصل ۴

الگوریتم $Simplex$

در این بخش قصد داریم به بررسی و توضیح الگوریتم $Simplex$ که یکی از قوی‌ترین الگوریتم‌ها برای حل مستقیم برنامه‌ریزی خطی می‌باشد، بپردازیم.

۴-۱ ورودی به الگوریتم $Simplex$

الگوریتم $Simplex$ یکی از قدیمی‌ترین الگوریتم‌ها برای حل مسائل برنامه‌سازی خطی می‌باشد. در حالی که این الگوریتم در بدترین حالت ممکن در زمان، دارای پیچیدگی نمایی می‌باشد، و این باور وجود ندارد که بتوان آن را چندجمله‌ای کرد، ولی به طور معمول در زمان بسیار خوبی جواب می‌دهد. در بخش بعدی پیچیدگی این الگوریتم را به طور دقیق‌تری مورد مطالعه قرار می‌دهیم.

در حال حاضر یک دید کلی درباره این الگوریتم را بررسی می‌کنیم. در این الگوریتم بجای آنکه تمام حالات bfs ها را بررسی کنیم، با حرکت هوشمندانه روی bfsها می‌توانیم سرعت این الگوریتم را بسیار پیشرفت دهیم. این دید کلی از الگوریتم $Simplex$ دارای ۳ فاز است:

الگوریتم (دید کلی از الگوریتم $Simplex$)

S : با یک bfs ابتدایی شروع کن.

M : به یک همسایه bfs برو اگر مقدار تابع مقدار را کاهش می‌دهد.

T : اگر چنین bfs ای وجود نداشت، متوقف شو.

مشاهده .

فاز S بسیار سخت می‌باشد!

ما نیاز داریم به اینکه شرط توقف قابل قبول باشد. یعنی شرط بهینه بودن موضعی نتیجه‌ای بر بهینه اکید بودن یک نقطه bfs باشد.

اگر فرم‌های دیگری در اختیار داریم، ابتدا آنها را به فرم $standard$ تبدیل می‌کنیم. زیرا این الگوریتم در این فرم بهتر کار می‌کند.

۴-۱-۱ فاز شروع

دیدیم پیدا کردن یک جواب ممکن مقدماتی که با آن الگوریتم $Simplex$ بتواند کار را شروع کند بسیار سخت است. در اینجا روشی ارائه می‌دهیم که توسط آن می‌توان یک چنین bfs ای را پیدا کرد.

برنامه‌ریزی خطی در فرم $standard$ زیر را در نظر بگیرید:

$$\min_{\vec{x}} \langle \vec{x}, \vec{c} \rangle \ , \ A\,\vec{x} = \vec{b} \ , \ \vec{x} \geq 0$$

فرض کنید بدون کاسته شدن از کلیت مسئله، $\vec{b}$ غیرمنفی می‌باشد، در غیر اینصورت ما می‌توانیم معادله‌ای که برای آن b_i منفی است را در ۱− ضرب کنیم، بدون اینکه فضای جواب تغییر کند.

برنامه‌ریزی خطی جدیدی به صورت زیر تعریف می‌کنیم:

$$\min_{\vec{y}} \sum_i y_i \ , \ A\,\vec{x} + \vec{y} = \vec{b} \ , \ \vec{x}, \vec{y} \geq 0$$

این مسئله می‌تواند به صورت زیر نیز نوشته شود:

$$[\ A \quad I\ ] \begin{bmatrix} \vec{x} \\ \vec{y} \end{bmatrix} = \vec{b}$$

ما برای حل این برنامه‌ریزی خطی می‌توانیم از الگوریتم $Simplex$ استفاده کنیم. دقت کنید که در این سیستم جدید فاز S آسان است. براحتی $\vec{x} = 0$ و $\vec{y} = \vec{b}$ را به عنوان یک bfs قرار دهید. اگر ما این مسئله را با $\sum_i y_i = 0$ حل کنیم، در این صورت ما یک جواب برای $A\,\vec{x} = \vec{b}$ پیدا کرده‌ایم و در نتیجه از bfs مورد نظر می‌توانیم شروع به حل مسئله اصلی بکنیم.

بطور دقیق‌تر، در پایان فاز اول، الگوریتم $Simplex$ یک جواب بهینه $[\ \vec{x^*} \quad \vec{y^*}\ ]$ به برنامه خطی ما می‌دهد. در این مرحله ۳ حالت زیر پیش می‌آید:

۱) اگر $\vec{y^*} \neq 0$ باشد، در اینصورت هیچ bfs ای برای برنامه اصلی وجود ندارد، به معنای دیگر برنامه‌ریزی خطی اصلی ما بدون جواب است.

۲) اگر $\vec{y^*} = 0$ باشد و $I = \{i |\ \vec{x^*} > 0\}$ دارای اندازه m باشد، $\vec{x^*}$ یک جواب ممکن برای برنامه‌ریزی خطی اصلی می‌باشد و I مجموعه ستون‌های مستقل خطی می‌باشد، یعنی $\vec{x^*}$ بردار bfs مربوط به I می‌باشد.

۳) اگر $\vec{y^*} = 0$ و $|I| < m$ باشد، در اینصورت می‌توان I را به یک مجموعه مستقل خطی J به اندازه m گسترش داد و این مجموعه را مجموعه ابتدایی قرار می‌دهیم.

۴-۱-۲ فاز حرکت

در حال حاضر ما روشی برای پیدا کردن یک bfs برای شروع کار داریم. حال ما آماده‌ایم که فاز حرکت را در این الگوریتم شرح دهیم. با کمی جبرخطی شروع می‌کنیم. مجموعه

$I = \{e_1, \ldots, e_m\}$ را مجموعه‌ای می‌گیریم که bfs اول به آن منتسب است. در نتیجه اگر bfs اول $\vec{x_0} = (x_1, \ldots, x_n)^T$ باشد، آنگاه:

$$\sum_{i=1}^{m} x_{0c_i} A_{e_i} = \vec{b}$$

همچنین هر ستون A که در I نمی‌باشد، می‌تواند به صورت ترکیب خطی از ستون‌های دیگر نوشته شود:

$$A_j = \sum_{i=1}^{m} r_{ij} A_{e_i}$$

در اینجا، ما r_{ij} را ضرایبی در نظر گرفتیم که توسط آنها ستون j ام توسط ستون‌های داخل I ساخته می‌شد. از نظر هندسی یک ترکیب خطی از ستون‌های I و یک ضریبی از ستون j ام یعنی αA_j $(0 \leq \alpha \leq \alpha_0)$، حرکتی روی یک یال را شبیه‌سازی می‌کند. حد بالای α_0 به زودی توضیح داده می‌شود. ما می‌توانیم ضریبی از ستون j ام را به صورت زیر بنویسیم:

$$\alpha A_j = \sum_{i=1}^{m} \alpha r_{ij} A_{e_i}$$

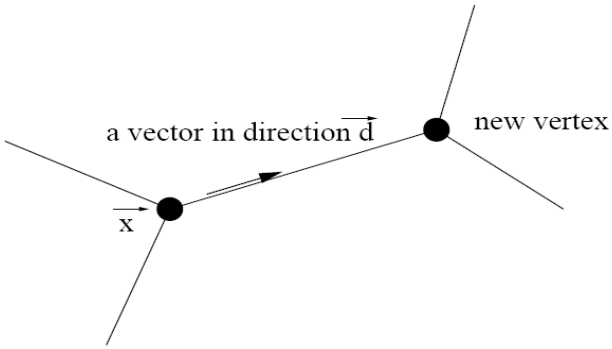
با کم و زیاد کردن مقدار αA_j از معادله اول بدست می‌آید:

$$\sum_{i=1}^{m} x_{0e_i} A_{e_i} - \alpha A_j + \alpha A_j = \vec{b}$$
$$\sum_{i=1}^{m} (x_{0e_i} - \alpha r_{ij}) A_{e_i} + \alpha A_j = \vec{b}$$

به بیان دیگر، ما مقدار تمام متغیرهای درون I را کم کرده و مقدار متغیر j ام را برابر α می‌کنیم و بقیه متغیرها صفر باقی می‌مانند.

در حالی که α زیاد می‌شود، مقدار جدید x_{c_i} برابر $(x_{e_i} - \alpha r_i)$ می‌شود. اگر $\alpha \geq 0$ و $x_{e_i} - \alpha r_i \geq 0$ باشد، ما هنوز یک جواب ممکن داریم.

اگر در حالتی اندیس i ای وجود داشته باشد که برای آن $x_{0e_i} - \alpha r_{ij} = 0$ شود، در این صورت ما به یک bfs جدید رسیده‌ایم که در برای آن i از مجموعه I خارج شده و j به آن اضافه می‌شود. در شکل ۴ حرکت از یک bfs به bfs دیگر نشان داده شده است. در اینجا



شکل ۱: حرکت روی جواب‌های ممکن مقدماتی

حرکت از $\vec{x}$ و در راستای بردار $\vec{d}=(\circ,\ldots,\circ,\overset{e_1}{-r_1},\circ,\ldots,\circ,\overset{j}{1},\circ,\ldots,\circ,\overset{e_m}{-r_m},\circ,\ldots)^T$ انجام می‌شود.

مثال ۴-۱ . در اینجا ۳ مثال برای پیدا کردن حد $\alpha_\circ$ می‌زنیم:

- $$x = ۴ \quad ۱ \quad ٦$$
$$r = ۸ \quad -۲ \quad -۹$$
که در اینجا $\alpha_\circ = \frac{۱}{۲}$ است.

- $$x = \circ \quad ۲ \quad ۵$$
$$r = ۱ \quad -۱ \quad -٦$$
در اینجا $\alpha_\circ = \circ$ است. در نتیجه هیچ حرکتی قابل انجام نیست. این حالتی است که یک متغیر $\circ$ و متناظر آن عددی مثبت باشد.

- $$x = \circ \quad ۱ \quad ٦$$

$$r = ۰ \quad -۱ \quad -۲$$

در اینجا $\alpha_۰ = \infty$ است. این حالت وقتی پیش می‌آید که $r_i \leq ۰$ باشد و در نتیجه اضافه کردن α هیچ عنصری را به محور ۰ نزدیک نمی‌کند. ما به طور معمول $\alpha_۰$ را به صورت زیر تعریف می‌کنیم:

$$\alpha_۰ \stackrel{def}{=} min_{\{i|r_{ij}>۰\}} \frac{x_{e_i}}{r_{ij}} = \frac{x_{e_l}}{r_{lj}}$$

دلیل مثبت بودن r_i در تعریف این است که، عمل کاهش فقط زمان‌هایی انجام می‌شود که برای i مقدار $r_i > ۰$ باشد.

در ۳ مثال بالا دیدیم که مقدار $\alpha_۰$ می‌تواند حالالت زیر را داشته باشد:

- $\alpha_۰ > ۰$. در این حالت ما می‌توانیم α را زیاد کنیم تا به مقدار لازم برسیم و به یک bfs جدید دست پیدا کنیم.
- $\alpha_۰ = ۰$. این حالت وقتی اتفاق می‌افتد که مقدار ۰ در بین x_{e_i} ها و مقدار مثبت r_i وجود داشته باشد. در این حالت ما نمی‌توانیم مقدار α را اضافه کنیم. ولی می‌توانیم j را به عنوان یک اندیس جدید به I اضافه کنیم و اندیس i را از I حذف کنیم.
- $\alpha = \infty$. این حالت وقتی اتفاق می‌افتد که هیچ راهی وجود ندارد و همه r_i ها منفی هستند.

لم ۴-۱ . اگر مقدار x'_i را به صورت زیر تعریف کنیم:

$$x'_i = \begin{cases} x_{e_j} - \alpha_۰ r_{kj} & i \in I, i = e_k \\ \alpha_۰ & i = j \end{cases}$$

و l اندیسی از مجموعه I باشد که متغیر مربوط به آن (x'_l) صفر شده باشد، در این صورت x' یک bfs جدید با مجموعه مشخصه $I' = I + \{j\} - \{l\}$ می‌باشد.

اثبات. ما اثبات خود را با نشان دادن اینکه x' جواب ممکن است شروع می‌کنیم. چون هر دو مقدار $x_{e_j} - \alpha_\circ r_{kj}$ و $\alpha_\circ$ نامنفی هستند، پس x' جواب ممکن است. حال نشان می‌دهیم I' هنوز مجموعه مشخصه باقی‌مانده است. برای اینکار نشان می‌دهیم ستون‌های $A_{e'_1}, \ldots, A_{e'_m}$ مستقل خطی هستند. فرض کنید j ستونی است که به مجموعه مشخصه وارد شده است. در نتیجه A_j می‌تواند به صورت ترکیب خطی از ستون‌های I به صورت زیر نوشته شود:

$$A_j = \sum_{i=1}^{m} r_{ij} A_{e_i}$$

بخاطر انتخاب l (اندیسی که متغیر متناظر با آن، یعنی x'_l صفر شده است) ما می‌دانیم که $r_l > \circ$ است. در نتیجه ما نامساوی زیر را داریم:

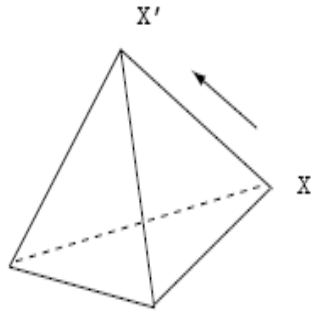
$$A_l = \frac{A_j - \sum_{i=1, i\neq l}^{m} r_{ij} A_{e_i}}{r_{lj}}$$

این نشان می‌دهد A_l، می‌تواند به صورت ترکیب خطی از ستون جدید A_j و ستون‌های قبلی نوشته شود. در نتیجه چون فضای ساخته شده توسط $A_{e_1}, \ldots, A_{e_m}$ و فضای ساخته شده توسط $A_{e'_1}, \ldots, A_{e'_m}$ برابر می‌شود، در نتیجه ستون‌های $A_{e'_1}, \ldots, A_{e'_m}$ مستقل خطی می‌باشند. □

۴-۲ چگونه جواب ممکن مقدماتی جدید را باید انتخاب کرد

وقتی ما از یک bfs به یک bfs دیگر حرکت می‌کنیم، ستون j به مجموعه مشخصه اضافه می‌شود. فرض کنید $\vec{v}$ بردار واحد در جهت حرکت ما باشد. داریم:

$$v_i = \begin{cases} -r_{ij} & i \in I \\ 1 & i = j \\ \circ & otherwise \end{cases}$$



شکل ۲: فاز حرکت در الگوریتم $Simplex$

در نتیجه bfs جدید برابر $\vec{x} - \alpha_\circ \vec{v}$ می‌شود. همچنین تابع مقدار جدید برابر است با:

$$z' = \langle \vec{c}, \vec{x'} \rangle$$

$$= \langle \vec{c}, \vec{x} + \alpha_\circ \vec{v} \rangle$$

$$= \langle \vec{c}, \vec{x} \rangle + \alpha_\circ \langle \vec{c}, \vec{v} \rangle$$

$$= z + \alpha_\circ (c_j - \sum_{i=1}^{m} r_{ij} c_{e_i})$$

اگر $\alpha_\circ > 0$ و $c_j - \sum_{i=1}^{m} r_{ij} c_{e_i} < 0$ باشد، ما می‌توانیم تابع مقدار را با اضافه کردن j به مجموعه بهبود بخشیم. در نتیجه وقتی می‌خواهیم ستون j ام را به مجموعه مشخصه اضافه کنیم، مقدار $p_j = c_j - \sum_{i=1}^{m} r_{ij} c_{e_i}$ را محاسبه می‌کنیم. اگر $p_j < 0$ و $\alpha_\circ > 0$ باشد، ما می‌دانیم که این حرکت باعث کمتر شدن تابع مقدار می‌شود. لم بعد نشان می‌دهد که اگر یک همچنین j ای وجود نداشته باشد، ما به هدف خود رسیده‌ایم.

لم ۴-۲ . فرض کنید $x_\circ$ یک bfs باشد بطوریکه برای هر $j \notin I$ مقدار $p_j \geq 0$ باشد، در این صورت $x_\circ$ جواب بهینه می‌باشد.

اثبات. فرض کنید $I = \{e_1, \ldots, e_m\}$ مجموعه اندیس‌های مشخصه bfs و N مجموعه بقیه اندیس‌ها باشد. اگر معادلات را بصورت ماتریسی بنویسیم $Ax = A_I x_I + A_N x_N = b$ برقرار است. درنتیجه داریم:

$$x_I = A_I^{-1}(b - A_N x_N)$$

اگر مقدار تابع مقدار را بر حسب x_N بنویسیم به هدف خود می‌رسیم. داریم:

$$z = \langle c, x \rangle$$

$$= c_I x_I + c_N x_N$$

$$= c_I A_B^{-1}(b - A_N x_N) + c_N x_N$$

$$= c_I A_B^{-1} b + (c_N - c_I A_I^{-1} A_N) x_N$$

برای bfs حال حاضر، یعنی x_0 مقدار $x_N = 0$ می‌باشد. در نتیجه z_0 مقدار حال حاضر تابع مقدار برابر $c_I A_I^{-1} b$ می‌باشد. از اطلاعات قبل می‌دانیم که $A_I^{-1} A_N = (r_{ij})$، یعنی ستون j ام آن ضرایب ترکیب خطی عنصر j ام مجموعه N بر حسب اعضای مجموعه I می‌باشد. در نتیجه داریم:

$$(c_N - c_I A_I^{-1} A_N) x_N = \sum_{j \notin I} p_j x_j$$

چون برای هر جواب ممکن x، $x_j \geq 0$ می‌باشد و چون برای $j \notin I$ ، $p_j \geq 0$ ، به این نتیجه می‌رسیم که برای هر جواب ممکن :

$$\langle c, x \rangle \geq c_I A_I^{-1} b = z_0$$

و به بیان دیگر z_0 مقدار مینیمم و x_0 جواب بهینه است. □

به این ترتیب، ما به این نتیجه رسیدیم که اگر برای هر j، r_i های مربوط به آن ستون نامثبت باشند، نتیجه می‌دهد که P هیچ حدی برای جواب بهینه ندارد، یعنی می‌توان تا

جای ممکن جواب را بهتر کرد. در اینجا می‌توانیم بگوییم که تابع مقدار بدون حد است اگر برای هر j مقدار r_i های مربوط به آن ستون نامثبت و $p_j < 0$ باشد. همچنین اگر $p_j \geq 0$ باشد و $\alpha_0 = \infty$، مجموعه جواب‌های ممکن p بدون حد می‌باشد ولی ممکن است که تابع مقدار محدود باشد. به هر حال الگوریتم $Simplex$ به این حالت وارد نمی‌شود، چون وقتی $p_j \geq 0$ است، j به مجموعه مشخصه وارد نمی‌شود.

۴-۳ الگوریتم $Simplex$

الگوریتم $(Simplex)$

با یک bfs ابتدایی x_0 با مجموعه مشخصه $I = \{e_1, \ldots, e_m\}$ که توسط فاز اول این الگوریتم بدست می‌آید، شروع کن.

مراحل زیر را تکرار کن:

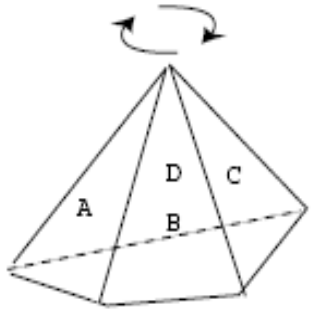
برای ستون j که در مجموعه مشخصه حال حاضر نمی‌باشد، مقدار r_{ij} ها و p_j را محاسبه کن.

اگر برای هر $j \notin I$، $p_j \geq 0$ باشد، متوقف شو و این bfs را به عنوان bfs بهینه گزارش کن.

ستون j را که برای آن $p_j < 0$ است را انتخاب کن:

اگر برای هر i، $r_{ij} \leq 0$ باشد، متوقف شو و تابع مقدار را به عنوان یک تابع بدون حد گزارش کن.

قرار بده $\alpha_0 = min_{\{i|r_i>0\}} \frac{x_{e_i}}{r_{ij}}$. فرض کنید l اندیسی باشد که برای آن $\alpha_0 = \frac{x_{e_l}}{r_{lj}}$ باشد. به bfs ای که جای دو ستون c_l و j را جابجا کرده برو.



شکل ۳: قوانین محور اصلی

۴-۳-۱ سوالاتی درباره الگوریتم $Simplex$

۱) آیا الگوریتم $Simplex$ یک الگوریتم چندجمله‌ای است و آیا اصلاً این الگوریتم پایان‌پذیر است؟

۲) آیا می‌توانیم محاسبه مقدار r_{ij} و p_j را به راحتی انجام دهیم؟

توضیح سوالات بالا:

۱) تعداد محدودی نقطه است که الگوریتم می‌تواند به آنها برود. در نتیجه اگر یک نقطه دو بار دیده نشود، این الگوریتم پایان می‌پذیرد. در غیر اینصورت دو راسی که بین آنها تکرار رخ داده حلقه‌ای ایجاد می‌کنند که تا ابد ادامه دارد. در ادامه خواهیم گفت که چون $\alpha_\circ > \circ$ است مقدار تابع مقدار کاهش می‌یابد و در نتیجه تکرار میسر نخواهد بود. $\alpha_\circ = \circ$ تنها در صورتی رخ می‌دهد که bfs مذکور دارای بیشتر از $n-m$ صفر باشد، که چون از یک مجموعه مشخصه به مجموعه دیگر می‌رویم، bfs تغییر نمی‌کند. به بیان دیگر اگر طبق شکل ۳ **از مجموعه مشخصه‌های** $ABC \to ABD \to BCD \to ABC$ **عبور کنیم، در یک دور می‌افتیم.**

۲) برای ستون k داریم:

$$A_k = \sum_{i=1}^{m} r_{ik} A_{e_i}$$

فرض کنید j ستونی باشد که به مجموعه مشخصه وارد شده و e_l ستونی باشد که از آن خارج شده است. از بخش قبل داریم:

$$A_{e_l} = \frac{A_j - \sum_{i=1, i\neq l}^{m} r_{ij} A_{e_i}}{r_{lj}}$$

در نتیجه در تکرار بعدی داریم:

$$\begin{aligned} A_k &= \sum_{i=1}^{m} r_{ik} A_{e_i} \\ &= \sum_{i\neq l} r_{ik} A_{e_i} + r_{lk} A_{e_l} \\ &= \sum_{i\neq l} r_{ik} A_{e_i} + r_{lk} \frac{A_j - \sum_{i\neq l} r_{ij} A_{e_i}}{r_{lj}} \\ &= \sum_{i\neq l} (r_{ik} - \frac{r_{lk} r_{ij}}{r_{lj}}) A_{e_i} + \frac{r_{lk}}{r_{lj}} A_j \end{aligned}$$

در نتیجه در تکرار جدید داریم:

$$r'_{ik} = \begin{cases} r_{ik} - \frac{r_{lk} r_{ij}}{r_{lj}} & i \neq l \\ \frac{r_{lk}}{r_{lj}} & i = l \end{cases}$$

به طرز مشابه می‌توان p'_k را از p_k بدست آورد.

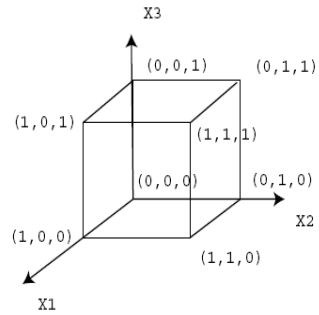
۴-۳-۲ قوانین محور اصلی

همانطور که گفته شد، زمانی که α_0 برابر صفر شود، ممکن است در دور بیفتیم. اجرای یکی از قوانین زیر نبود هر گونه دوری را تضمین می‌کند.

۱) ستونی را به مجموعه وارد کن که کمترین p_j را داشته باشد:

$$j = \arg \min_{j: p_j < 0} p_j$$

این قانون از به دور افتادن جلوگیری می‌کند.



شکل ۴: مثالی از یک *polytope* با تعداد رئوس نمایی

۲) ستونی را به مجموعه وارد کن که بیشترین سود را به تابع مقدار بدهد:

$$j = \arg\max_{j:p_j<0} (z' - z)$$

این قانون از به دور افتادن جلوگیری می‌کند.

۳) ستون با کمترین اندیس را به مجموعه وارد کن:

$$j = \min\{j : p_j < 0\}$$

و کسی را حذف کن که کمترین اندیس را داشته باشد:

$$e_l = \min\{e_l : \frac{x_{e_l}}{r_{lj}} = \min_{\{i|r_{lj}>0\}} \frac{x_{e_i}}{r_{ij}}\}$$

این قانون نسبت به نبود هر گونه دوری تضمین می‌کند.

۴-۳-۳ $Simplex$ یک الگوریتم از زمان چندجمله‌ای نمی‌باشد

از آنجایی که گفتیم این الگوریتم یکی از بهترین روش‌ها برای حل برنامه‌های خطی می‌باشد، این سوال به طور طبیعی مطرح شده است که خوب بودن این الگوریتم توسط روش‌های ریاضی و چندجمله‌ای بودن آن ثابت شود. ما قصد داریم نشان دهیم الگوریتم $Simplex$ نمی‌تواند به جواب بهینه در زمان چندجمله‌ای با استفاده از یکی از قوانین محور

اصلی برسد. قبل از هر چیز این سوال مطرح است که آیا تعداد رئوسی که روی *polytope* وجود دارد چندجمله‌ای برحسب بعد آن می‌باشد؟ ولی جواب «خیر» می‌باشد. *polytope* هایی وجود دارند که تعداد رئوس آنها نمایی می‌باشد. یک *polytope* ساده برای این مدعا یک *polytope* با شروط

$$۰ \leq x_j \leq ۱ \quad j = ۱, ۲, \ldots, n$$

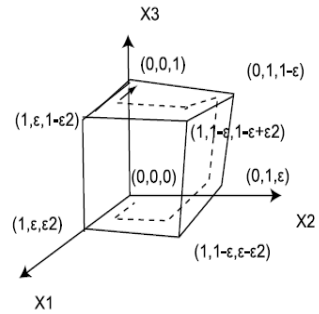
می‌باشد. این *polytope* دارای $۲n$ $facet$ برای هر چندجمله‌ای و دارای ۲^n راس با قرار دادن هر x_j برابر ۰ یا ۱ می‌باشد. مثال برای یک ۳ وجهی در شکل ۳ آمده است. داشتن تعداد نمایی راس نمی‌تواند لزوماً نکته منفی باشد، چون این الگوریتم همه رئوس را طی نمی‌کند. برای مثال در شکل ۴ ممکن است یک مسیر از $A \to D$ بجای $A \to B \to C \to C \to D$ برپایه یکی از اصول محور انتخاب شود. برای اینکه نشان دهیم این الگوریتم در خطر است باید مثالی زده شود که برای آن دنباله‌ای بلند از رئوس را طی کند که تابع مقدار در راستای این دنباله کاهش پیدا کند، که این مثال نشان می‌دهد این الگوریتم تعداد زیادی تکرار لازم دارد. برای رسیدن به این هدف ما یک و یک تابع مقدار که در یک مسیر کاهشی طولانی بیفتد را مثال می‌زنیم. ما سعی می‌کنیم یک ساختار ترکیبیاتی زیبا برای این n وجهی پیدا کنیم. در اینجا یک همچین ساختاری که توسط کلی[۱] و مینتی[۲] داده شده است، نشان می‌دهیم. فرض کنید $\epsilon \in (۰, \frac{۱}{۲})$ و نابرابری‌های زیر را در نظر بگیرید:

$$\begin{array}{ccc}
۰ & \leq x_۱ \leq & ۱ \\
\epsilon x_۱ & \leq x_۲ \leq & ۱ - \epsilon x_۱ \\
 & \vdots & \\
\epsilon x_{i-۱} & \leq x_i \leq & ۱ - \epsilon x_{i-۱} \\
 & \vdots & \\
\epsilon x_{n-۱} & \leq x_n \leq & ۱ - \epsilon x_{n-۱}
\end{array}$$

---

۱ Klee

۲ Minty



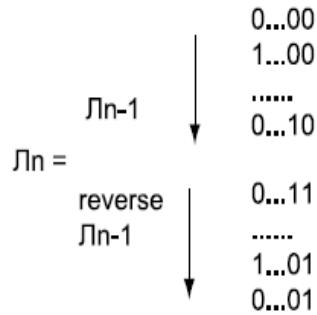
شکل ۵: نمایی بودن الگوریتم $Simplex$

این مثال در حالت ۳ بعدی در شکل ۵ نشان داده شده است. ما یک برنامه خطی با شروط بالا که تابع مقدار آن $-x_n$ است تعریف می‌کنیم. وقتی این برنامه خطی را با اگوریتم $Simplex$ حل کنیم، ما در هر مرحله برای وارد کردن و خارج کردن هر ستون در هر مرحله اختیار داریم. نشان می‌دهیم دنباله‌ای از رئوس وجود دارد که $Simplex$ در آن تعداد نمایی حرکت انجام دهد. در ابتدا هر راس این مثال در حالت n بعدی را با دنباله‌ای $\{۰, ۱\}^n$ نشان می‌دهیم. این کار را با مچ کردن ϵ با ۰ انجام می‌دهیم. برای مثال در حالت ۳ بعدی داریم:

Ver	$Encode$
$(۰, ۰, ۰)$	۰۰۰
$(۱, \epsilon, \epsilon^۲)$	۱۰۰
$(۱, ۱-\epsilon, \epsilon-\epsilon^۲)$	۱۱۰
$(۰, ۱, \epsilon)$	۰۱۰
$(۰, ۱, ۱-\epsilon)$	۰۱۱
$(۱, ۱-\epsilon, ۱-\epsilon+\epsilon^۲)$	۱۱۱
$(۱, \epsilon, ۱-\epsilon^۲)$	۱۰۱
$(۰, ۰, ۱)$	۰۰۱

در نتیجه وقتی ϵ به ۰ مچ شود، این دو چندوجهی به هم مچ می‌شود. حال ما می‌خواهیم یک مسیر هامیلتونی در $\{۰, ۱\}^n$ پیدا کرده و نشان دهیم در این مسیر مقدار x_n افزایش می‌یابد، که برای این‌کار از کد گری[۳] استفاده می‌کنیم:

[۳] Gray



شکل ۶: کد گری

برای مثال این مسیر در حالت ۳ بعدی به صورت زیر است:

$$\Pi_۳ = ۰۰۰ \to ۱۰۰ \to ۱۱۰ \to ۰۱۰ \to ۰۱۱ \to ۱۱۱ \to ۱۰۱ \to ۰۰۱$$

لم ۴-۳ . تابع مقدار برنامه خطی بالا وقتی از یک bfs به bfs دیگر در مسیر هامیلتونی گفته شده در بالا، حرکت می‌کنیم، کاهش پیدا می‌کند.

در ابتدا ببینیم که چگونه می‌توانیم bfs را در برنامه خطی بالا پیدا کنیم. این برنامه دارای $۲n$ شرط است. طبق چیزی که در تمارین مطرح می‌شود، bfs توسط n نابرابری مستقل خطی انتخاب می‌شود که برابری باشند. برای هر جفت نابرابر $x_i \geq \epsilon x_{i-۱}$ و $x_i \leq ۱ - \epsilon x_{i-۱}$، ما می‌توانیم یکی از آنها را انتخاب کنیم. در نتیجه در کل ما ۲^n انتخاب داریم. براحتی می‌توان دید هر کدام از این ۲^n انتخاب مستقل خطی می‌باشند. در نتیجه در این مثال ۲^n جواب ممکن مقدماتی داریم که به ۲^n راس ما اشاره می‌کند.

اثبات. این کار را با استقرا انجام می‌دهیم. برای پایه استقرا در $n = ۲$، مسیر همیلتونی $\Pi_۲ = ۰۰ \to ۱۰ \to ۱۱ \to ۰۱$ را داریم. برنامه خطی ما به صورت زیر است :

$$\min -x_۲$$

$$\begin{array}{rcr} x_۱ & \geq & ۰ \\ x_۱ & \leq & ۱ \\ x_۲ & \geq & \epsilon x_۱ \\ x_۲ & \leq & ۱ - \epsilon x_۱ \end{array}$$

به راحتی می‌توان نشان داد که تابع مقدار در این مسیر کاهش می‌یابد. یعنی (x_1, x_2) ترتیب زیر را طی می‌کنند:

$$(0,0) \to (1,\epsilon) \to (1, 1-\epsilon) \to (0,1)$$

حال به اثبات فرض استقرا می‌پردازیم. فرض می‌کنیم فرض برای چند وجهی ما در بعد $n-1$ برقرار باشد. سعی می‌کنیم آن را برای حالت n بعدی ثابت کنیم.

ما حرکت را از راس $0\ldots0$ به $010\ldots0$ شروع می‌کنیم که بخش اول Π_n است. طبق فرض استقرا این یک مسیر کاهشی برای بعد $n-1$ است. از آنجایی که عنصر آخر در این مسیر همیشه 0 است ما می‌دانیم که همیشه باید $x_n = \epsilon x_{n-1}$ را انتخاب کنیم. طبق فرض استقرا x_{n-1} در حال افزایش است و در نتیجه تا به اینجا برنامه خطی ما در بعد n کاهش می‌یابد. سپس در این مسیر از $010\ldots0$ به $011\ldots0$ می‌رویم که x_n در آن از ϵx_{n-1} به $1-\epsilon x_{n-1}$ تغییر می‌کند که چون $\epsilon < \frac{1}{2}$ است باز x_n افزایش پیدا می‌کند و تابع مقدار کاهش پیدا می‌کند. در آخر نیمه دوم مسیر را از $11\ldots0$ به $01\ldots0$ می‌رویم. $n-1$ بیت اول این مسیر عکس مسیر Π_{n-1} می‌باشند. در نتیجه x_{n-1} در طول این نیمه مسیر کاهش پیدا می‌کند و چون بیت آخر همیشه ۱ است، در نتیجه $x_n = 1-\epsilon x_{n-1}$ است. در نتیجه x_n در طول این نیمه مسیر نیز افزایش پیدا کرده و در نتیجه اندازه تابع مقدار در طول مسیر کاهش پیدا کرده و حکم ثابت می‌شود. □

در این مثال نشان دادیم که الگوریتم $Simplex$ یک مسیر نمایی را طی کرد. این نکته را بدون اثبات رها می‌کنیم که بهترین اصول محور اصلی هم ما را در این مسیر قرار می‌دهند. در نظر داشته باشید که هیچ تضمینی برای راسی که برنامه‌ریزی خطی را از آن شروع کنیم، وجود ندارد. در نتیجه حرکت از راسی که در مثال انتخاب شده، مشکلی نداشته و در نتیجه ممکن است این الگوریتم در این مسیر نمایی حرکت کند.

۴-۴ تحلیل $Smooth$

بدترین حالت زمان اجرا، بیشترین زمان T را برای الگوریتم A در تمام حالات ورودی I_n به ما می‌گوید، در حالی که زمان متوسط یک الگوریتم، مقدار زمانی را که به طور متوسط طول می‌کشد، می‌گوید. تحلیل $Smooth$ بیشترین زمان اجرا را در بازه اصلی میانگین زمان اجرا بررسی می‌کند. این بازه توسط متغیر σ مشخص می‌شود. ما می‌توانیم تصور کنیم که σ فاصله از حالت میانگین را بازگو می‌کند. این ما را برای بررسی الگوریتم در حالات مختلف تواناتر می‌کند. در فرم ریاضی این تحلیل‌ها عبارت‌اند از:

۱) بدترین حالت $(A;n)$ $= \max_{w\in I_n} T(x)$

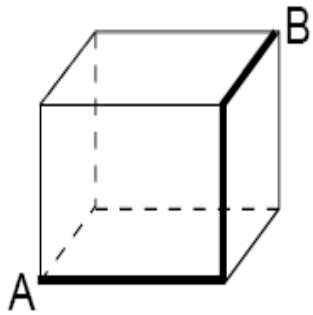
۲) حالت میانگین $(A;n)$ $= \text{avg}_{w\in I_n} T(x)$

۳) حالت $Smooth$ $(A;n;\sigma)$ $= \max_{w\in I_n} \text{avg}\, T($ باشد x همسایه σ در $w)$

براحتی می‌توان دید که $Eqn.\ ۱ \geq Eqn.\ ۳ \geq Eqn.\ ۲$. برای سوالات برنامه‌ریزی خطی مقدار σ بستگی به چگونگی حل مسئله دارد.

قضیه ۴-۱ . حالت $Smooth$ $(n;\sigma;$ الگوریتم $Simplex)$ = poly $(m, n, \frac{1}{\sigma})$.

قضیه بالا می‌گوید که الگوریتم $Simplex$ هر برنامه خطی را در زمان چندجمله‌ای بر حسب m و n و $\frac{1}{\sigma}$ حل می‌کند. در واقع بدترین حالت وقتی است که مقدار σ برابر ۰ شود. اگر σ زیاد باشد آنوقت به حالت میانگین نزدیک می‌شویم و در حالتی که σ نه زیاد است و نه کم، یک چیزی بین بدترین حالت و حالت میانگین می‌باشیم. در مسائل برنامه‌ریزی خطی بدترین حالت نمایی و حالت میانگین چندجمله‌ای می‌باشد. این قضیه به ما می‌گوید که الگوریتم $Simplex$، الگوریتم نسبتاً خوبی است.



شکل ۷: حرکت روی جواب‌های ممکن مقدماتی

۴-۴-۱ نتایج $Kalai$

فرض کنید $\Delta(n,m)$ بیشینه طول یک قطر در $polyhedron$ ی باشد که توسط m شرط و n متغیر تعریف می‌شود. در این صورت $\Delta(n,۲n) \geq n$ می‌باشد. ما این نتیجه را با یک مثال نشان می‌دهیم.

مثال ۴-۲ . شروط زیر یک چندوجهی با بعد n را که در شکل ۵ نشان داده شده است، تعریف می کنند:

$$\begin{aligned} ۰ \leq \quad x_۱ \quad &\leq ۱ \\ &\vdots \\ ۰ \leq \quad x_n \quad &\leq ۱ \end{aligned}$$

هر مسیر بین دو نقطه در این $polyhedron$ حداکثر n مرحله دارد. از آنجایی که قطر یک $polyhedron$ بلندترین مسیر بین کوتاهترین مسیرهای بین رئوس می‌باشد، در نتیجه بیشترین قطر ممکن در این مثال n می‌باشد.

حدس . در حالت کلی $\Delta(n,m) = \Theta(n+m)$.

بوضوح، مسیری که توسط $\Delta(n,m)$ تعریف می‌شود در جهت تابع مقدار حرکت نمی‌کند. در واقع این کوتاهترین مسیر بین دو نقطه می‌باشد. بیشینه قطر جهت‌دار $\Delta'(n,m)$ برابر بزرگترین مسیر بین کوتاهترین مسیرها در جهت تابع مقدار $\langle c,x\rangle$ بین دو

نقطه در یک $polyhedron$ که با m شرط و n نقطه تعریف می‌شود، می‌باشد. در نتیجه $\Delta'(n,m) \geq \Delta(n,m)$ می‌باشد.

در الگوریتم $Simplex$ ما از یک راس $polyhedron$ شروع کردیم و به راس دیگر رفتیم که مقدار بیشتری تابع مقدار را داشت، و این کار را ادامه دادیم تا به جواب بهینه برسیم. اگر الگوریتم $Simplex$ در τ تکرار برای هر برنامه خطی با m شرط و n متغیر تمام شود، در اینصورت $\tau \geq \Delta'(n,m) \geq \Delta(n,m)$ می‌باشد.

در سال ۱۹۹۲، الگوریتم $Kalai$ از ایده قطر برای گرافی که برای یک چندوجهی ساخته می‌شود، بدست آمد. این الگوریتم یک الگوریتم $Simplex$ احتمالی بود، که حد بالای تکرار برای آن از $\exp(O(\sqrt{n \log n}))$ می‌باشد. این اولین باندی بود که برای $\Delta'(n,m)$ بدست آمد.

فصل ۵

الگوریتم‌های ترکیبیاتی احتمالی

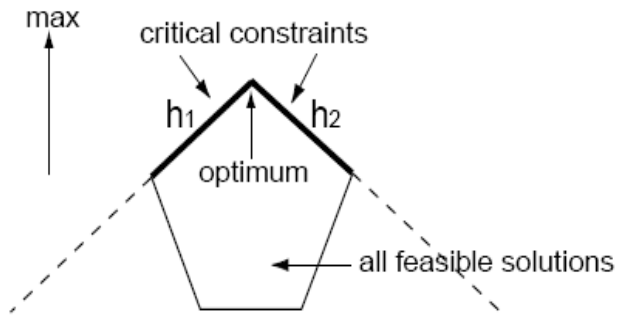
در این بخش قصد داریم مثالی از الگوریتم‌های احتمالی در حل برنامه‌ریزی خطی را نشان دهیم.

۵-۱ الگوریتم‌های ترکیبیاتی احتمالی

برنامه‌ریزی خطی $Ax \leq b$ با n متغیر و m شرط را که تعداد متغیرهای آن کم است، در نظر بگیرید. فرض کنید H مجموعه شروط باشد. به یاد آورید که اگر x یک راس باشد، در این صورت n نابرابری مستقل خطی به صورت برابری برای x در می‌آیند. جواب برای سوال با این n شرط برابر جواب مسئله اصلی ما خواهد بود. ما به این نابرابری‌ها، شروط بحرانی می‌گوییم. شروط بحرانی در شکل ۱ نشان داده شده‌اند.

۵-۱-۱ الگوریتم $Seidel$

این الگوریتم یک الگوریتم احتمالی ساده برای برنامه‌های خطی می‌باشد که برای n های کوچک خوب کار می‌کند.



شکل ۱: حرکت روی جواب‌های ممکن مقدماتی

الگوریتم $Seidel\ (H)$

به طور احتمالی یک $h \in H$ را انتخاب کن (H مجموعه شروط می‌باشد)

$x = Seidel\ (H - \{h\})$ قرار بده.

اگر x در h صدق می‌کرد

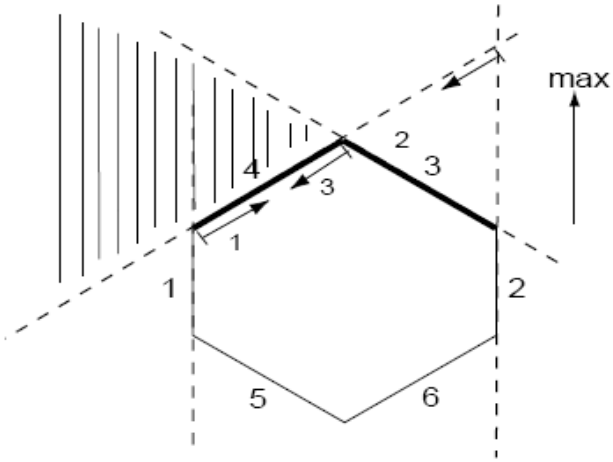
x را برگردان

در غیر این صورت

تصویر همه شروط H را در شرط h گرفته و H' را از آنها بساز.

$Seidel\ (H')$ را برگردان.

در این الگوریتم اگر x در h صدق کند که جواب بهینه مسئله اصلی می‌باشد. در غیر اینصورت جواب حتماً باید در ابر صفحه‌ای که h تعریف می‌کند قرار گیرد. یعنی h در جواب بهینه به صورت برابری می‌باشد. در این حالت ما تمام شروط دیگر را بر این ابر صفحه تصویر می‌کنیم و در نهایت ما برنامه خطی با $n - ۱$ متغیر و $m - ۱$ شرط را به طور بازگشتی حل می‌کنیم.

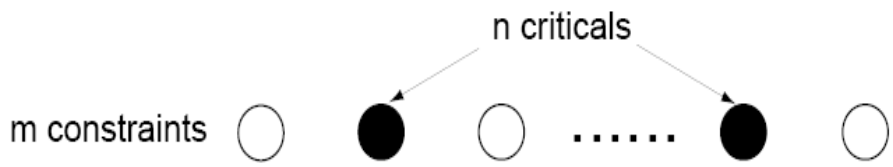


شکل ۲: الگوریتم $seidel$

با یک مثال که در شکل ۲ نشان داده شده، به این الگوریتم می‌پردازیم. در این مثال تابع مقدار به بالا اشاره می‌کند و $polytope$ با ٦ ابرصفحه محاط شده است. بوضوح جواب بهینه در محل برخورد ابرصفحه ۳ و ۴ می‌باشد. در واقع این دو ابرصفحه، دو شرط بحرانی این مسئله می‌باشند.

هرکدام از ابرصفحه‌های ۱ و ۲ و ۵ و ۶ می‌تواند حذف شود بدون اینکه x غیر ممکن شود. اگر در حلقه اول $h = ۴$ باشد، جواب x باید در قسمت هاشور خورد باشد و شرط h را برقرار نمی‌کند. پس ما می‌فهمیم که ابرصفحه ۴ یکی از شروط بحرانی می‌باشد و تصویر تمام شروط دیگر بر این شرط را در نظر می‌گیریم. در نتیجه $polytope$ به یک برنامه خطی با بعدی به اندازه یکی کمتر از قبلی تبدیل شده است و جواب بهینه نیز در این حالت ثابت باقی می‌ماند.

فرض کنید k شرط بحرانی در بین m شرط وجود داشته باشد. در بهترین حالت ما دوست داریم که الگوریتم $m - k$ شرط غیر بحرانی را قبل از انتخاب شرط بحرانی h دور بیندازد. در این حالت ما تعداد کمتری شرط برای تصویر کردن روی h داریم و الگوریتم سریع‌تر تمام می‌شود. در بدترین حالت، الگوریتم k شرط بحرانی را قبل از شروط دیگر



شکل ۳: نابرابری‌های بحرانی در الگوریتم $seidel$

انتخاب می‌کند و برای هر کدام از این شروط ما نیاز داریم که تمام شروط غیر بحرانی را به روی آن تصویر کنیم و این باعث می‌شود که الگوریتم کندتر شود. در ادامه ما پیچیدگی این الگوریتم را بررسی می‌کنیم.

۵-۱-۲ تحلیل الگوریتم $Seidel$

فرض کنید A ماتریسی $m \times n$ باشد و $m \geq n$. در هر بازگشت تعداد شروط یکی کم می‌شود. در هر مرحله زمان $O(n)$ طول می‌کشد تا صادق بودن x را در h چک کنیم. وقتی x در h صدق نمی‌کرد، زمان $O(mn)$ لازم است تا تمام شروط دیگر را بر شرط h تصویر کنیم. در نتیجه پیچیدگی این الگوریتم بازگشتی به صورت زیر است:

$$T(n,m) = T(n,m-1) + O(n) + \begin{cases} 0 & if\ x\ satisfies\ h \\ O(mn) + T(n-1,m-1) & otherwise \end{cases}$$

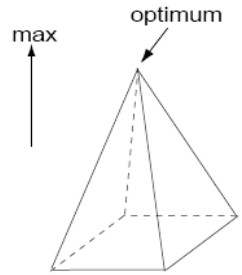
رتبه A هیچ‌وقت نمی‌تواند از n بیشتر شود، یعنی حداکثر n شرط از m شرط، بحرانی می‌باشد. به بیان دیگر هر راس از $polytope$ با حداکثر n شرط مشخص می‌شود. در نتیجه حالتی که x در شرط h صدق نکند به احتمال $\frac{n}{m}$ اتفاق می‌افتد. در نتیجه داریم:

$$T(n,m) = T(n,m-1) + O(n) + \frac{n}{m}(O(mn) + T(n-1,m-1))$$

لم ۵-۱ . الگوریتم $Seidel$ در زمان نمایی $O(n!m)$ اجرا می‌شود.

اثبات. با استقرای قوی روی $n+m$ حکم را ثابت می‌کنیم.

اگر $n = 1$ باشد، الگوریتم تنها نیاز به چک کردن m مقدار دارد. در نتیجه الگوریتم در زمان $O(m)$ اجرا می‌شود که همان $O(n!m) = O(m)$ است. اگر $m = 0$ باشد هم



شکل ۴: الگوریتم $seidel$

$T(n,\circ) = O(\circ)$. فرض کنید برای مقادیر کمتر از $n+m$ حکم درست باشد، یعنی اگر $i+j < n+m$ باشد، $T(i,j) = O(i!j)$ است. می‌خواهیم نشان دهیم $T(n,m) = O(n!m)$. داریم:

$$T(n,m) = T(n,m-۱) + O(n) + \frac{n}{m}(O(mn) + T(n-۱,m-۱))$$

$$T(n,m) = O(n!(m-۱)) + O(n) + O(n^۲) + \frac{n}{m}O((n-۱)!(m-۱))$$

$$= O(n!(m-۱)) + O(n!)$$

$$= O(n!m)$$

و در نتیجه حکم ثابت می‌شود. □

به خاطر داشته باشید که مجموعه شروط بحرانی می‌توانند یکتا نباشند. همانطور که در شکل ۴ دیده می‌شود، نقطه‌ای که در بالای هرم قرار دارد، نقطه بهینه است. اگر H' را مجموعه ۴ ابرصفحه که در نقطه بهینه برخورد کرده‌اند بگیریم، هر ۳ تا از H' نیز نقطه بهینه مسئله را مشخص می‌کنند، یعنی اگر الگوریتم $Seidel$ یکی از این ۴ شرط را از H' حذف کند، ۳ شرط باقیمانده شروط بحرانی می‌باشند.

۵-۱-۳ پیشرفت $Matousek, Sharir, Welzl$

در سال ۱۹۹۲، $Matousek, Sharir, Welzl$ الگوریتم $Seidel$ را توسط یک ساختار پیشرفته‌تر، پیشرفت دادند. این پیشرفت به یک پیچیدگی کوچکتر منجر شد که برابر $exp(۲\sqrt{n \ln \frac{m}{\sqrt{n}}} + O(\sqrt{n} + \ln m))$ می‌باشد.

فصل ۶

Duality

در این بخش به مبحث *Duality* در برنامه‌ریزی خطی می‌پردازیم. قضیه *Duality* را ثابت می‌کنیم و همچنین درباره *Complementary slackness* توضیح داده، لم *Farkas* را ثابت کرده که رابطه بسیار نزدیکی با یکدیگر دارند. و در آخر قضیه کمبیشینه *Neumann* را توضیح خواهیم داد.

۶-۱ قضیه *Duality*

اجازه دهید در ابتدا کار را با ۲ مثال شروع کنیم.

مثال ۶-۱ . برنامه خطی زیر را در نظر بگیرید:

$$\min -x_۲$$

$$x_۱ + x_۲ \leq ۸$$

$$-۳x_۱ + ۲x_۲ \leq ۶$$

چگونه می‌توانیم حد پایینی برای جواب بهینه پیدا کنیم؟ داریم:

$$۳(x_۱ + x_۲) \leq ۲۴$$

$$۳(x_۱ + x_۲) + (-۳x_۱ + ۲x_۲) \leq ۳۰$$

و در نتیجه:

$$-x_۲ \geq -٦$$

و به این ترتیب ٦− حد پایینی برای جواب بهینه می‌باشد.

مثال ۶-۲ . برنامه خطی زیر را در نظر بگیرید:

$$\max ۵x_۱ + ٦x_۲ + ۹x_۳ + ۸x_۴$$

$$x_۱ + ۲x_۲ + ۳x_۳ + x_۴ \leq ۵$$

$$x_۱ + x_۲ + ۲x_۳ + ۳x_۴ \leq ۳$$

$$x_i \geq ۰ \ , \ \ i = ۱, ۲, ۳, ۴$$

چگونه می‌توانیم یک حد بالایی برای جواب بهینه بدست آوریم ؟ داریم:

$$۵x_۱ + ٦x_۲ + ۹x_۳ + ۸x_۴ \leq ۸(x_۱ + ۲x_۲ + ۳x_۳ + x_۴) \leq ۴۰$$

و همچنین داریم:

$$۵x_۱ + ٦x_۲ + ۹x_۳ + ۸x_۴ \leq ٦(x_۱ + x_۲ + ۲x_۳ + ۳x_۴) \leq ۱۸$$

یعنی نابرابری دوم حد بالای قوی‌تری را به ما نشان داد. جواب بهینه $x_۱ = ۱, x_۲ = ۲, x_۳ = x_۴ = ۰$ را در نظر بگیرید، داریم:

$$۵x_۱ + ٦x_۲ + ۹x_۳ + ۸x_۴ = ۵ + ۱۲ + ۰ + ۰ = ۱۷$$

و در نتجیه این یک حد بالای دیگر می‌باشد. همچنین اگر نابرابری اول را با ۴ برابر نابرابری دوم جمع کنیم، داریم:

$$۵x_۱ + ٦x_۲ + ۹x_۳ + ۸x_۴ \leq ۵x_۱ + ٦x_۲ + ۱۱x_۳ + ۱۳x_۴ \leq ۱۷$$

در نتیجه این می‌تواند کار خوبی باشد که به دنبال یک حد بالای خوب برای سوالات بیشینه سازی، و یا یک حد پایین برای مسائل کمینه سازی بگردیم. فرض کنید $y_۱, y_۲ \geq ۰$ دو ضریب باشند که می‌خواهیم از دو معادله در این مثال با هم ترکیب کنیم. در نتیجه داریم:

$$y_۱(x_۱ + ۲x_۲ + ۳x_۳ + x_۴) + y_۲(x_۱ + x_۲ + ۲x_۳ + ۳x_۴)$$
$$= (y_۱ + y_۲)x_۱ + (۲y_۱ + y_۲)x_۲ + (۳y_۱ + ۲y_۲)x_۳ + (y_۱ + ۳y_۲)x_۴$$

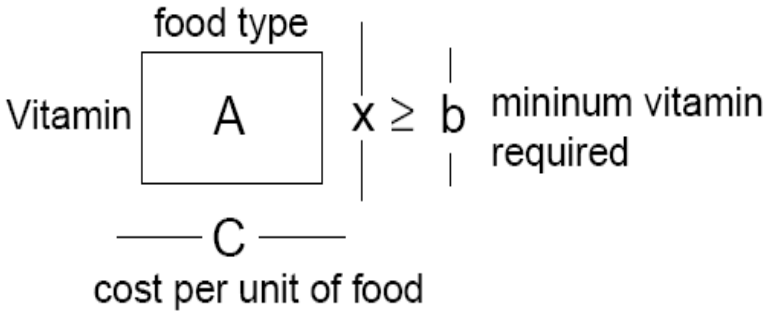
در نتیجه برای اینکه ضریب هر متغیر بیشتر از تابع مقدار باشد، باید

$$y_۱ + y_۲ \geq ۵$$
$$۲y_۱ + y_۲ \geq ٦$$
$$۳y_۱ + ۲y_۲ \geq ۹$$
$$y_۱ + ۳y_۲ \geq ۸$$

از طرفی تابع مقداری که در بالا داشتیم طبق شروط بالا، باید کمتر از $۵y_۱ + ۳y_۲$ باشد. در نتیجه حد بالا پیدا کردن برای سوال بالا مثل کمینه کردن مقدار $۵y_۱ + ۳y_۲$ با توجه به شروط بالا می‌باشد. این نکته ما را به برنامه‌ریزی خطی زیر راهنمایی می‌کند:

$$\min ۵y_۱ + ۳y_۲$$
$$y_۱ + y_۲ \geq ۵$$
$$۲y_۱ + y_۲ \geq ٦$$
$$۳y_۱ + y_۲ \geq ۹$$
$$y_۱ + ۳y_۲ \geq ۸$$
$$y_۱, y_۲ \geq ۰$$

دقت کنید که ضرایب در تابع مقدار، مقادیر در سمت راست نابرابری‌های مسئله اصلی می‌باشند. همچنین سمت چپ نابرابری‌ها ترانسپوز ماتریس A می‌باشد و سمت راست آنها نیز ضرایب در تابع مقدار مسئله اصلی است. جواب این مسئله یک حد بالا برای مسئله اصلی می‌باشد. به این مسئله *dual* مسئله اصلی گفته می‌شود و به مسئله اصلی *primal* می‌گوییم.



شکل ۱: مسئله رژیم غذایی

بیایید به مسئله رژیم غذایی که در بخش اول مطرح شد بازگردیم.

مثال ۶-۳ . (مسئله رژیم غذایی) یک انسان می‌خواهد برای خود یک برنامه غذایی داشته باشد، به طوری که در هر روز از هر ویتامین به مقدار کافی بخورد و همچنین مجموع هزینه غذاهایی که می‌خورد کمینه شود. n غذای متفاوت وجود دارد که هر گرم از غذای j ام قیمتی برابر $c_j \in R$ دارد. انسان برای زنده ماندن نیاز به b_i میلیگرم از ویتامین i ام در هر روز دارد. این آدم برای اینکه مقدار هزینه غذای خود را کمینه کند و زنده بماند چه باید بکند؟

این مسئله یک سوال برنامه‌ریزی خطی می‌باشد که به صورت زیر نمایش داده می‌شود و در شکل ۱ نیز نمایش داده شده است.

$$\min\langle c, x\rangle$$

$$Ax \geq b$$

$$x_i \geq ۰ \quad , \quad ۱ \leq i \leq n$$

چگونه می‌توانیم حد پایین قیمت را بدست آوریم. به بیان دیگر چگونه می‌توانیم این مسئله را به *dual* آن تبدیل کنیم.

طبق مشاهدات مثال قبل، قصد داشتیم بردار y را که مقدار $\langle y, b\rangle$ را کمینه می‌کند پیدا کنیم، بطوری که در شروط $yA \leq c$ صدق کند، در نتیجه مسئله جدید به صورت برنامه‌ریزی خطی زیر در می‌آید :

$$\max\langle y, b\rangle$$
$$yA \leq c$$
$$y_i \geq ۰ \ , \ \ ۱ \leq i \leq m$$

برای دیدن نتایج این عمل می‌توانیم بردار y را به عنوان قیمت ویتامین‌ها نگاه کنیم و بردار c را قیمت هر واحد غذا در نظر بگیریم. مسئول فروشگاه می‌خواهد سود خود را بیشینه کند به طوری که قیمت هر نوع غذا به‌صرفه باشد.

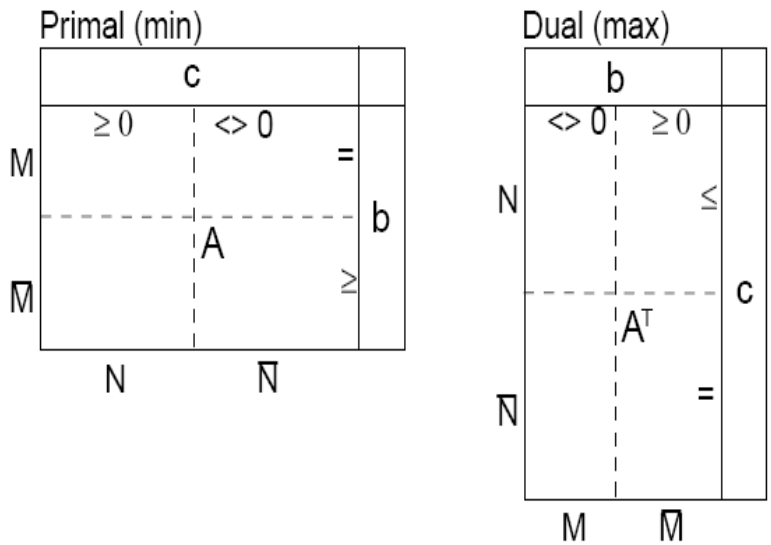
این سوال نیز توسط *dual* خود دارای یک باند پایین خوب می‌شود. به همین ترتیب برای هر سوال بیشینه‌سازی، یک سوال کمینه‌سازی وجود دارد و برعکس که *dual* یکدیگر باشند، که این مسائل حد بالا و پایین برای یکدیگر می‌باشند. در واقع همانطور که خواهیم دید جواب‌های آنها با یکدیگر برابر می‌باشد.

۶-۲ کمی درباره‌ی *Duality*

ما در این قسمت *dual* فرم‌های مختلف یک برنامه خطی را بررسی و روشهای مناسب برای رسیدن از *primal* به *dual* را مطالعه می‌کنیم. این قسمت را با درآمدی بر سوالات وابسته به قضیه *Duality* پایان می‌دهیم.

۶-۲-۱ تبدیل فرم‌های مختلف *primal* به *dual*

هر سوال برنامه‌ریزی خطی می‌تواند به صورت *primal* و *dual* دیده شود. در هر صورت ما به سوال اصلی *primal* و به دیگری *dual* می‌گوییم. یک برنامه‌ریزی خطی که در حالت

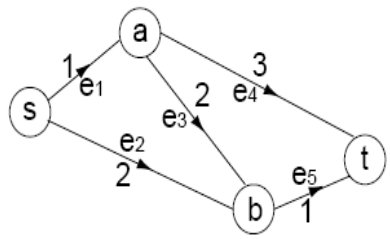


شکل ۲: تبدیل *primal* به *dual*

کلی داده شود، *primal* و *dual* به صورت زیر تعریف می‌شوند:

Primal		*Dual*
$\min\langle c, x\rangle$		$\max\langle \pi, b\rangle$
$a_i x = b_i$	$i \in M$	$\pi_i <> 0$
$a_i x \geq b_i$	$i \in \overline{M}$	$\pi_i \geq 0$
$x_j \geq 0$	$j \in N$	$\pi A_j \leq c_j$
$x_j <> 0$	$j \in \overline{N}$	$\pi A_j = c_j$

در حالت کلی اگر *primal* جواب داشته باشد، *dual* نیز جواب دارد. همچنین ممکن است هر دوی آنها بدون جواب باشند. اگر یکی از آنها بدون حد باشد، حتماً دیگری بدون جواب است. در اینجا ما می‌خواهیم ببینیم که چگونه می‌توان بین *primal* و *dual* رابطه برقرار کرد. توابع مقدار *primal* و *dual* با عدد πAx با هم مرتبط می‌شوند. به طور دقیق‌تر داریم $cx \geq \pi Ax \geq \pi b$. *primal* و *dual* یک برنامه‌ریزی خطی را به صورت شکل ۲ نیز می‌توان نمایش داد که درک ما را از رابطه ریاضی بین آنها راحت‌تر می‌کند.



شکل ۳: مثال کوتاهترین مسیر

۶-۲-۲ موارد استفاده : کوتاهترین مسیر

قضیه $Duality$ دامنه استفاده بسیار گسترده‌ای دارد. ما مفید بودن آن را در مثال کوتاهترین مسیر می‌بینیم.

مثال ۶-۴ . گراف جهت‌دار $G = (V, E)$ در شکل ۳ نشان داده شده است. هزینه $c_j \geq ۰$ به هر یال $e_j \in E$ نسبت داده شده است. ما می‌خواهیم کوتاهریتن مسیر از راس s به راس t را در این گراف پیدا کنیم.

گراف G می‌تواند توسط یک ماتریس A که هر درآیه آن به صورت زیر تعریف می‌شود، مشخص شود:

$$a_{ij} = \begin{cases} +۱ & if\ e_j\ leaves\ v_i \\ -۱ & if\ e_j\ arrives\ v_i \\ ۰ & otherwise \end{cases}$$

$$A = \begin{pmatrix} +۱ & +۱ & ۰ & ۰ & ۰ \\ ۰ & ۰ & ۰ & -۱ & -۱ \\ -۱ & ۰ & +۱ & +۱ & ۰ \\ ۰ & -۱ & -۱ & ۰ & +۱ \end{pmatrix}$$

هر یال e_j دارای شار $f_j > ۰$ است اگر برای ساخت مسیر از s به t انتخاب شود. سوال کمینه کردن مقدار هزینه مسیر است. در نتیجه برنامه خطی برای این سوال به صورت زیر است:

$$\min\langle c, f\rangle$$

$$Af = b$$

$$f_i \geq ۰ \ \ , \ \ i = ۱, ۲, ۳, ۴, ۵$$

که بردار b مشخص کننده شار رد و بدل شده بین رئوس در مسیر نهایی می‌باشد. به بیان دیگر، درآیه راس s باید $+۱$ باشد که به این معنی است که یک واحد شار از آن خارج شده است و درآیه راس t باید -۱ باشد که به این معنی است که یک واحد شار به آن وارد شده است و بقیه درآیه‌ها باید ۰ باشند، به این معنی که هر راس باید هر شاری که وارد می‌کند، خارج کند. یک bfs برای این سوال باید $|E| - |V|$ درآیه صفر داشته باشد.

ممکن است جواب غیر مشخص s' نیز برای این سوال وجود داشته باشد. در واقع ما می‌توانیم از s' به یک جواب ممکن مقدماتی بهینه برای این سوال برسیم. چون همه جواب‌های ممکن در پوش محدب bfs ها قرار می‌گیرند، در اینصورت $s' = \sum_i \lambda_i s_i$ که $۰ \leq \lambda_i \leq ۱$ و $\sum_i \lambda_i = ۱$ است. فرض کنید j اندیسی باشد که bfs با کمترین هزینه را مشخص می‌کند. در اینصورت داریم:

$$c.s' = \sum_i \lambda_i (c.s_i) \geq (c.s_j) \sum_i \lambda_i = c.s_j$$

که این نشان می‌دهد s_j تنها یک bfs نیست و بلکه یک جواب ممکن مقدماتی بهینه می‌باشد. *dual* این سوال به صورت زیر است:

$$\max \langle \pi, b \rangle$$

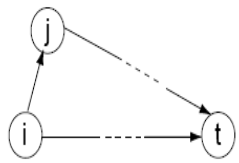
$$\pi A \leq c$$

$$\pi_i <> ۰ \ \ , \ \ i \in V$$

ما می‌توانیم این برنامه خطی را به صورت زیر بازنویسی کنیم:

$$\max \pi_s - \pi_t$$

$$\pi_i - \pi_j \leq c_{ij} \ \ , \ \ \forall (i,j) \in E$$



شکل ۴: شرط نامساوی مثلثی در مسئله کوتاهترین مسیر

$$\pi_i <> 0 \quad , \quad i \in V$$

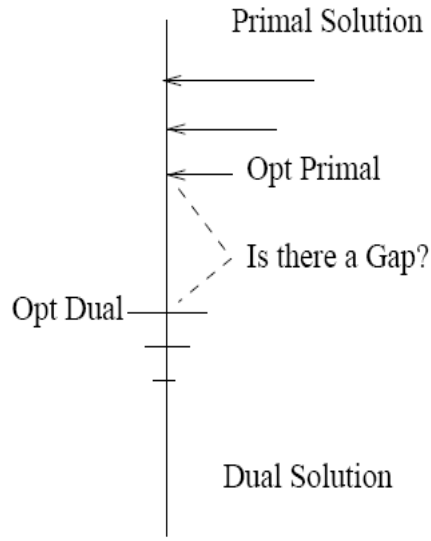
در این‌صورت ممکن بودن $dual$ را بررسی می‌کنیم. فرض کنید d_{ij} کوتاهترین مسیر بین دو راس در G باشد. برای هر راس v در V، فرض کنید $\pi_v = d_{vt}$. شروط $dual$ بصورت $d_{it} - d_{jt} \leq c_{ij}$ در می‌آیند که طبق شکل شرط نامساوی مثلثی را بازگو می‌کنند. به بیان دیگر اگر یک کوتاهترین مسیر بین راس i و t وجود داشته باشد، هزینه آن نباید از هزینه رفتن از i به t با عبور از راس j بیشتر باشد. در نتیجه شروط ممکن می‌باشند.

و در نهایت می‌خواهیم نشان دهیم که جواب بهینه برای $dual$ ، جواب مسئله کوتاهترین مسیر از s به t می‌باشد. از آنجایی که d_{st} یک جواب ممکن برای $dual$ است، پس $d_{st} \leq \pi_{opt}b$، که $\pi_{opt}b$ جواب بهینه ما می‌باشد. در این مثال بوضوح، کوتاهترین مسیر از s به t مسیر $s \to b \to t$ است. براساس $dual$ دو نابرابری زیر را داریم:

$$\pi_s - \pi_b \leq c_{sb}$$

$$\pi_b - \pi_t \leq c_{bt}$$

از جمع این دو نابرابری بدست می‌آید که $\pi_s - \pi_t \leq c_{sb} + c_{bt} = d_{st}$. در نتیجه از آنجایی که $\pi_{opt}b = \pi_s - \pi_t$ می‌باشد، داریم $\pi_{opt}b \leq d_{st} \leq \pi_{opt}b$. در نتیجه d_{st} جواب بهینه ما می‌باشد. پس جواب بهینه برای $dual$ ، کوتاهترین مسیر از s به t می‌باشد.



شکل ۵: نسبت *primal* و *dual* در برنامه‌ریزی خطی

۶-۳ قضیه ضعیف و قوی *Duality*

طبق آنچه که در شکل ۵ می‌بینیم، می‌توان ایده خوبی برای پیدا کردن جواب *primal* و *dual* پیدا کرد.

این شکل ما را به قضیه‌های زیر راهنمایی می‌کند. قبل از اینکه این قضایا را بررسی و اثبات کنیم، به این نکته توجه کنید که ما تمام اثبات‌های خود را در فرم *standard* برنامه‌ریزی خطی انجام می‌دهیم، ولی این نکته ما را محدود نمی‌کند که قضایا در فرم‌های دیگر ثابت نشوند، همانطور که قبلاً ثابت کرده بودیم که این فرم‌ها می‌توانند به هم تبدیل شوند.

قضیه ۶-۱ .(قضیه ضعیف *Duality*)
اگر x یک جواب ممکن برای *primal* و y یک جواب ممکن برای *dual* باشد، در اینصورت $\langle c, x\rangle \geq \langle y, b\rangle$.

اثبات. طبق مشاهدات ما از $primal$ و $dual$ ، سریعاً به این نتیجه می‌رسیم که :

$$\langle y, b\rangle = yAx \leq \langle c, x\rangle$$

□

طبق این قضیه ما نمی‌توانیم بفهمیم که آیا بین جواب بهینه برای $primal$ و جواب بهینه برای $dual$ وقتی که هر دوی این مسائل دارای جواب ممکن هستند، فاصله‌ای وجود دارد یا خیر. قضیه‌ی قوی $Duality$ که در زیر بیان می‌شود، به ما می‌گوید که چنین فاصله‌ای وجود ندارد.

قضیه ۶-۲ .(قضیه قوی $Duality$)
اگر یک برنامه‌ریزی خطی دارای جواب بهینه باشد، در اینصورت $dual$ آن نیز دارای جواب بهینه است و همچنین جواب بهینه آنها با هم برابر است.

یکی از نکات جالبی که در این قضیه وجود دارد، کاربرد آن در الگوریتم $Simplex$ می‌باشد. در واقع ما می‌توانیم از این خاصیت در توقف الگوریتم $Simplex$ استفاده کنیم.

اثبات. فرض کنید $primal$ و $dual$ به صورت زیر باشند:

$Primal$	$Dual$
$\min\langle c, x\rangle$	$\max\langle y, b\rangle$
$Ax = b$	$yA \leq c$
$x \geq 0$	$y <> 0$

فرض کنید مسئله $primal$ ما دارای جواب باشد، در اینصورت در قسمت توقف الگوریتم $Simplex$ با پایه مشخصه I و ستون‌های باقیمانده N نابرابری‌های زیر را داریم:

$$p = c_N - c_I A_I^{-1} A_N \geq 0$$

$$c_I A_I^{-1} A_N \leq c_N$$

در نتیجه، از آنجایی که $c_I A_I^{-1} A_I = c_I$ داریم:

$$c_I A_I^{-1} A \leq c$$

اگر قرار دهیم $y = c_I A_I^{-1}$، در اینصورت $yA \leq c$ می‌باشد. در نتیجه اگر *primal* دارای جواب بهینه باشد، در اینصورت ما می‌توانیم یک جواب بهینه $c_I A_I^{-1}$ را برای *dual* پیدا کنیم. از آنجایی که $x = A_I^{-1} b$ ، رابطه زیر بدست می‌آید :

$$\langle y, b \rangle = c_I A_I^{-1} b = \langle c, x \rangle$$

پس به این نتیجه می‌رسیم که اگر *primal* دارای جواب بهینه باشد، *dual* دارای جواب ممکن است و همچنین جوابی دارد که مقدار آن با جواب بهینه *primal* برابر می‌باشد، در نتیجه *dual* نیز دارای جواب بهینه می‌باشد. □

۶-۴ رده‌بندی‌های ممکن *primal* و *dual*

زوج مسائل *primal* و *dual* را در نظر بگیرید. توضیحات ما درباره امکان جواب، در سه حالت بدون جواب، دارای جواب ولی نامحدود و دارای جواب محدود می‌باشد. جدول زیر حالات ممکن این دو مسئله را بازگو می‌کند:

		$Primal$		
		$Has\ Optimum$	$Unbounded$	$Infeasible$
$Dual$	$Has\ Optimum$	$\checkmark^*(Same\ Optimum)$	$\times^\circ$	$\times^*$
	$Unbounded$	$\times^\circ$	$\times^\circ$	$\checkmark^\circ$
	$Infeasible$	$\times^*$	$\checkmark^\circ$	$\checkmark$

می‌توان دید رده‌ی $(Infeasible, Infeasible)$ هم اتفاق می‌افتد: فرض کنید *primal* جواب ندارد، در اینصورت ما *dual* آن را می‌نویسیم و آنقدر شرط اضافه می‌کنیم تا *dual* دیگر جواب نداشته باشد، در اینصورت *primal* برای *dual* جدید کماکان بدون جواب است.

براحتی می‌توان دید *dual* برای *dual* یک مسئله، *primal* آن می‌باشد. در نتیجه جدول بالا باید متقارن باشد.

۶–۵ *Complementary slackness*

برای توضیحات در این بخش ما از برنامه‌ریزی خطی در فرم *canonical* آن استفاده می‌کنیم. *primal* و *dual* زیر را در نظر بگیرید:

Primal	*Dual*
$\min\langle c, x\rangle$	$\max\langle y, b\rangle$
$Ax \geq b$	$yA \leq c$
$x \geq 0$	$y \geq 0$

قبلاً شروطی را برای بهینه بودن یک بردار توضیح دادیم. در ادامه این صحبت‌ها قضیه زیر شرط متوازنی را بین x, y که دو جواب بهینه برای *primal* و *dual* می‌باشند، نشان می‌دهد.

قضیه ۶–۳. (*Complementary slackness*)
فرض کنید x و y به ترتیب جواب‌های ممکن برای *primal* و *dual* باشند، در اینصورت جواب‌های بهینه هستند اگر و فقط اگر:

$$\forall i\ ,\ x_i = 0 \ \ or\ \ (yA)_i = c_i$$

$$\forall j\ ,\ y_j = 0 \ \ or\ \ (Ax)_j = b_j$$

اثبات. از آنجایی که x و y جواب ممکن هستند، در نتیجه:

$$\langle c, x\rangle \geq yAx \geq \langle y, b\rangle$$

طبق قضیه قوی *Duality* ، x و y بهینه هستند اگر و فقط اگر $\langle c, x\rangle = \langle y, b\rangle$. برابری اول را در نظر بگیرید، یعنی $\langle c, x\rangle = yAx$ که می‌توانیم آن را به صورت زیر بنویسیم:

$$\langle c - yA, x\rangle = 0$$

از آنجایی که $x \geq 0$ و $yA \leq c$، می‌توانیم نتیجه بگیریم که برای هر iی یا $x_i = 0$ است و یا $\langle y, a_i \rangle = c_i$ می‌باشد. به همین ترتیب برابری دوم نیز ثابت می‌شود. □

۶-۶ امکان جواب در یک سیستم خطی و لم $Farkas$

حال بیایید به این سوال جواب دهیم که چه وقت یک سیستم از برابری‌ها و نابرابری‌ها دارای جواب است. از قبل می‌دانیم که اگر برابری‌ها را در مقادیری ضرب کنیم و با هم جمع کنیم و اگر به برابری درستی نرسیم، در این صورت می‌توان نتیجه گرفت که سیستم بدون جواب است.

برای مثال سیستم زیر را در نظر بگیرید:

$$x_1 + 2x_2 = 5$$

$$x_3 - x_2 = -3$$

$$x \geq 0$$

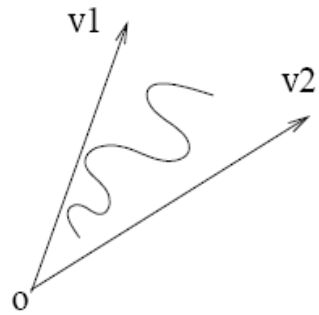
اگر برابری اول را با ۲ برابر برابری دوم جمع کنیم، داریم:

$$x_1 + 2x_2 + 2x_3 - 2x_2 = 5 - 2 \times 3$$

$$\Rightarrow x_1 + 2x_3 = -1$$

و از آنجایی که $x \geq 0$، می‌فهمیم که این سیستم جواب ندارد.

این مثال یک راه ساده را برای فهمیدن اینکه آیا یک سیستم دارای جواب است، به ما نشان می‌دهد. برای اینکه این راه را به یک روش فرمال تبدیل کنیم، می‌گوییم که اگر $\exists\, y\ ,\ yA \geq 0$ و $\langle y, b \rangle < 0$، در اینصورت $Ax = b$ که $x \geq 0$ جواب ندارد. این می‌تواند به صورت یک شرط ضروری ایجاد شود. یعنی، عکس این شرط نیز برقرار است: اگر یک سیستم خطی جواب نداشته باشد، در اینصورت یک اثبات خطی برای آن وجود دارد. لم $Farkas$ این نکته را بازگو می‌کند.



شکل ۶: cone یک دسته بردار

۶-۶-۱ لم $Farkas$

لم $Farkas$ در سال ۱۸۹۴ توسط $J.Farkas$ بدست آمد. نگاه کردن به این لم به عنوان دید هندسی قضیه $Duality$ می‌تواند مفید باشد.

قضیه ۶-۴ . $\forall\, y\ ,\ yA \geq 0 \Rightarrow \langle y, b\rangle \geq 0 \Leftrightarrow$ جواب دارد $Ax = b\ ,\ x \geq 0$.

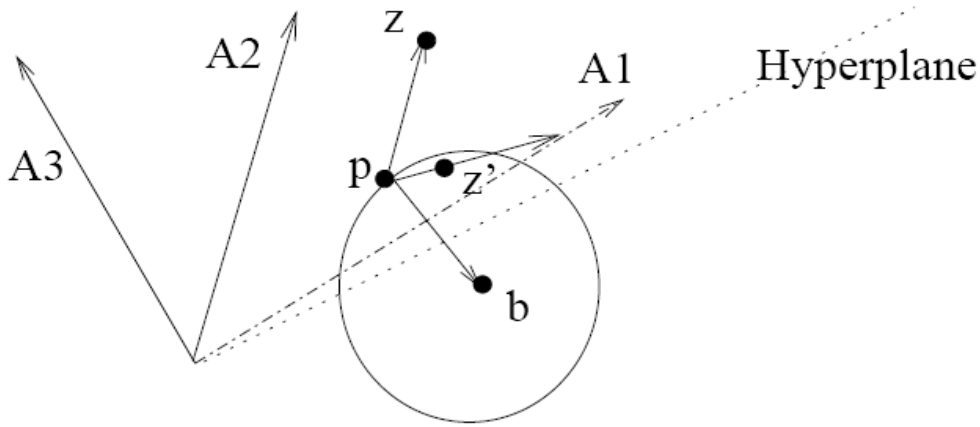
اثبات. قسمت $\Leftarrow$ مشخص است: اگر x یک جواب برای $Ax = b$ باشد که $x \geq 0$، و اگر $yA \geq 0$ باشد، در نتیجه $yAx \geq 0$. در نتیجه از آنجایی که $yAx = \langle y, b\rangle$، داریم $\langle y, b\rangle \geq 0$.

حال طرف دیگر این قضیه را ثابت می‌کنیم، که بسیار جذاب است. در واقع شاید غافلگیر کننده باشد، اگر این قضیه را از قضیه قوی $Duality$ اثبات کنیم.

برای بردارهای $v_1, \ldots v_n$، تعریف می‌کنیم $cone(v_1, \ldots, v_n) = \{\sum_i \lambda_i v_i | \lambda_i \geq 0\}$. شکل ٦ این تعریف را نشان می‌دهد.

اگر ستون‌های A را به صورت بردار در نظر بگیریم، در اینصورت غیرممکن بودن $x \geq 0$ که $Ax = b$ معادل آن است که $b \notin cone(A_1, \ldots, A_n)$.

از آنجایی که $\langle y, A_i\rangle \geq 0$ و $\langle y, b\rangle \leq 0$، می‌توانیم y را به صورت یک ابرصفحه که از مرکز رد می‌شود و تمام A_i ها در یک طرف آن و b در طرف دیگر آن قرار دارد، نگاه کرد.



شکل ۷: لم $farkas$

فرض کنید $K = cone(A_1, \ldots, A_n)$ باشد و p نزدیک‌ترین نقطه به b در K باشد. این نقطه وجود دارد، زیرا فاصله یک تابع پیوسته، و K یک مجموعه بسته می‌باشد. در ابتدا نشان می‌دهیم که به ازای هر نقطه $z \in K$ داریم $\langle z-p, b-p\rangle \leq 0$. اگر شرط مذکور برقرار نباشد، ما می‌توانیم نقطه z' که فاصله آن به b از p کمتر است را در K پیدا کنیم. در شکل ۷ رابطه بین b و p و z و z' نشان داده شده است.

فرض کنید $Aw = p$ و $y = p - b$ می‌باشد. از این نکته می‌توان نتیجه گرفت که $\forall\, x \geq 0 \ : \ \langle Ax - Aw, y\rangle \geq 0$ می‌باشد. در نتیجه، برای $x = w + e_i$ داریم:

$$\forall i \ , \ \langle Ae_i, y\rangle \geq 0 \Rightarrow \forall i \ , \ \langle A_i, y\rangle \geq 0$$

در نتیجه y مورد نظر، K را در یک طرف خود دارد. حال ما نیاز داریم نشان دهیم b در طرف دیگر آن قرار دارد.

می‌دانیم که $\langle p-b, b-p\rangle < 0$ است (چون $b \notin cone(A_1, \ldots, A_n)$). چون p نزدیکترین نقطه به b در K می‌باشد و $\vec{0} \in K$ است، از توضیح قبلی داریم که $\langle p-b, p-\vec{0}\rangle \leq 0$. از جمع این دو رابطه با هم داریم:

$$\langle y, b\rangle = \langle p-b, b\rangle = \langle p-b, b-p\rangle + \langle p-b, p-\vec{0}\rangle < 0$$

که نشان می‌دهد b در طرف دیگر y قرار دارد.

در نتیجه اگر $Ax = b$ و $x \geq 0$ جواب نداشته باشد، وجود دارد y بطوریکه $yA \geq 0$ و $\langle y, b\rangle < 0$ باشد. $\square$

فصل ۷

قضیه کم‌بیشینه‌ی $Von\ Neumann$، اصل کم‌بیشینه Yao

در این بخش قضیه کمبیشینه $Von\ Neumann$ را ثابت می‌کنیم. در ادامه به سوال خواص الگوریتم‌های احتمالی جواب داده و اصل کمبیشینه Yao را ثابت می‌کنیم. در آخر هم شروع به توضیح الگوریتم $Ellipsoid$ کرده و نشان می‌دهیم که این الگوریتم از زمان چندجمله‌ای می‌باشد.

۷-۱ اصل کمبیشینه $von\ Neumann$

بازی جمع ۰، یک بازی دو نفره است، که در آن هر بازیکن یک مجموعه متناهی از روش‌ها دارد. سود نفر اول بر اساس روش‌هایی است که دو نفر انتخاب می‌کنند و سود نفر دوم برابر منفی روش نفر اول است. در نتیجه مجموعه سود آنها منفی است. بازی سنگ و

کاغذ و قیچی یک بازی جمع ۰ می‌باشد، که امتیازها در جدول زیر آمده است:

		Column Player		
		Paper	*Stone*	*Scissor*
Row Player	*Paper*	۰	۱	-۱
	Stone	-۱	۰	۱
	Scissor	۱	-۱	۰

اگر بازیکن سطری استراتژی i ام و بازیکن ستونی استراتژی j ام را انتخاب کنند، سود برای بازیکن سطری برابر a_{ij} می‌باشد. اگر بازیکن سطری اول بازی کند، سود زیر را دارد:

$$\max_i \min_j a_{ij}$$

و اگر دوم بازی کند سود زیر را دارد:

$$\min_j \max_i a_{ij}$$

برای دو حالت بالا داریم: $\forall i \quad \min_j a_{ij} = -۱$ و $\forall j \quad \max_i a_{ij} = ۱$. در نتیجه به راحتی می‌توان نشان داد که بهترین کار این است که به عنوان نفر دوم بازی کنیم.

حالتی را در نظر بگیرید که یک بردار احتمالی حرکت یک بازیکن را مشخص کند. در نتیجه حرکت هر نفر به صورت احتمالی محاسبه می‌شود، و ترتیب حرکت بازیکن‌ها کمتر اهمیت دارد. تعریف می‌کنیم $\Delta_k = \{\alpha \in R^k | \alpha \geq ۰ \ , \ \sum \alpha_i = ۱\}$.

برای حالت روش‌های احتمالی، دو عنصر زیر را با هم مقایسه می‌کنیم:

بازیکن ستونی اول بازی کند: $\max_{x \in \Delta_m} \min_{y \in \Delta_n} yAx$

بازیکن ستونی دوم بازی کند: $\min_{y \in \Delta_n} \max_{x \in \Delta_m} yAx$

که x و y بردارهای احتمال هستند که $yAx = \sum x_j y_i a_{ij}$ می‌باشد.

به راحتی می‌توان دید که $\max_x min_y yAx \leq \min_y \max_x yAx$ می‌باشد. ولی آیا برابری برقرار می‌باشد؟ قضیه زیر به این سوال پاسخ مثبت می‌دهد.

قضیه ۷-۱ . (**قضیه** کمبیشینه $Von\ Neumann$)
وجود دارد بردارهای x و y به طوری که

$$\max_{x\in\Delta_m}\min_{y\in\Delta_n} yAx = \min_{y\in\Delta_n}\max_{x\in\Delta_m} yAx$$

کلیت بازی ۰ و ۱ را در حالت ۲ نفره که در بخش قبل توضیح دادیم به یاد آورید. ما یک ماتریس m در n، A که درآیه‌ی a_{ij} آن سود بازیکن سطری وقتی که استراتژی i ام را انتخاب کرد و بازیکن ستونی استراتژی j ام را انتخاب کرد نشان می‌دهد. در همین حال سود برای نفر ستونی برابر $-a_{ij}$ می‌باشد. فرض کنید x_j و y_i احتمال انتخاب استراتژی j ام و i ام توسط بازیکن ستونی و سطری باشد. در نتیجه سود در این حالت yAx می‌باشد. چه اتفاقی می‌افتد اگر یک بازیکن بازی احتمالی را انتخاب کرده و بازیکن دیگر استراتژی را که دوست دارد انتخاب کند؟ و چه اتفاقی می‌افتد اگر ما ترتیب بازیکن‌ها را عوض کنیم؟ قضیه مذکور می‌تواند به این پرسش‌ها پاسخ دهد.

اثبات. دقت کنید که برای یک استراتژی انتخاب شده x، سود $\min_{y\in\Delta_n} yAx$ یک برنامه‌ریزی خطی ساده است که دارای مجموعه دامنه $\{y \geq ۰\ ,\ \sum_j y_j = ۱\}$ می‌باشد. چون رئوس این $polytope$ ، $\{e_i\}_{i=۱}^n$ می‌باشند، که e_i یک بردار با درآیه ۱ در مکان i ام و ۰ در بقیه جاها می‌باشد. به بیان دیگر، ما می‌توانیم فرض کنیم که نفر دوم یک استراتژی خاص را انتخاب می‌کند، تا بهترین سود را بکند. داریم:

$$\max_{x\in\Delta_m}\min_{y\in\Delta_n} yAx = \max_{x\in\Delta_m}\min_{i}(Ax)_i$$

به همین ترتیب داریم:

$$\min_{y\in\Delta_n}\max_{x\in\Delta_m} yAx = \min_{y\in\Delta_n}\max_{j}(yA)_j$$

در نتیجه ما تنها نیاز داریم که نشان دهیم:

$$\max_{x\in\Delta_m}\min_{i}(Ax)_i = \min_{y\in\Delta_n}\max_{j}(yA)_j$$

فرم برنامه‌ریزی خطی این سوال را به صورت زیر می‌نویسیم:

$$\max\ t$$
$$\sum_j a_{ij}x_j \geq t$$
$$\sum_j x_j = ۱$$
$$x \geq ۰$$
$$t \neq ۰$$

و $dual$ آن به صورت زیر می‌باشد:

$$\min\ w$$
$$y \geq ۰$$
$$w \neq ۰$$
$$w - \sum_i y_i a_{ij} \geq ۰$$
$$\sum_i y_i = ۱$$

به راحتی می‌توان دید که $primal$ دارای جواب است (هر $x \in \Delta_m$ و $t = \min_{ij} a_{ij}$ ممکن است) و دارای حد است (مثلاً $\max_{ij} a_{ij}$). در نتیجه طبق قضیه $Duality$ ، $primal$ و $dual$ دارای جواب‌های بهینه مشترک می‌باشند. ما $dual$ را به صورت زیر بازنویسی می‌کنیم:

$$\min\ w$$
$$\sum_i y_i a_{ij} \leq w$$
$$y \in \Delta_m$$
$$w \neq ۰$$

که همان $\min_{y\in\Delta_m} \max_j \sum_i y_i a_{ij}$ می‌باشد، در نتیجه قضیه کمبیشینه ثابت می‌شود. □

فرض کنید x^* و y^* جواب‌هایی برای مسئله باشند. y^*Ax^* یک جواب مشترک می‌باشد. از اثبات بالا، داریم $t^* = w^* = y^*Ax^*$. اگر قضیه $Complementary\ slackness$ را روی آن اجرا کنیم، برای هر i و j داریم:

$$\sum_j a_{ij}x_j^* > t^* \to y_i^* = ۰$$

$$\sum_i y_i^* a_{ij} < w^* \to x_j^* = ۰$$

و در نتیجه اگر x^* بهینه باشد و I مجموعه‌ای از اندیس‌ها باشد که به ازای آنها $t^* - \sum_j a_{ij}x_j^* < ۰$ ، در اینصورت برای $i \in I$، $y_i^* = ۰$ می‌باشد. این مثل آن است که بگوییم بازیکن y نمی‌تواند یک احتمال مثبت به یک استراتژی که برای x^* بهینه نیست بدهد. این برای بازیکن x هم صادق است. طرف دیگر این حرف نیز صادق است. یعنی اگر x و y در این شروط صدق کنند، هر دو بهینه می‌باشند.

۷-۲ اصل کمبیشینه Yao

مسئله Π را در نظر بگیرید، فرض کنید I مجموعه تمام ورودی‌های ممکن با یک اندازه مشخص و متناهی باشد. همچنین فرض کنید A مجموعه تمام الگوریتم‌های ممکن برای Π باشد، که باز فرض کنید متناهی باشند. فرض کنید اندازه A، n و اندازه I، m باشد. فرض کنید زمانی که طول می‌کشد تا الگوریتم a روی ورودی i اجرا شود $T(a,i)$ باشد. عبارت

$$\min_{a\in A}\max_{i\in I} T(A,I)$$

بدترین زمان اجرا برای بهترین الگوریتم می‌باشد. حال تمام الگوریتم‌های احتمالی $LasVegas$ را برای یک مسئله در نظر بگیرید. اینها الگوریتم‌هایی هستند که همیشه درست می‌باشند و زمان اجرای آنها مقداری احتمالی می‌باشد. هر چنین الگوریتمی می‌تواند به صورت یک $a \in A$ در نظر گرفته شود. برای یک چینش q روی A، ما a_q را

یک الگوریتم احتمالی که توسط q انتخاب می‌شود (a_{q_i} به احتمال q_i) در نظر می‌گیریم. حال $E[T(a_q,i)] = \sum q_i T(a_{q_i},i)$، و امید خطی

$$\min_{q\in\Delta_n}\max_{i\in I} E[T(a_q,i)]$$

دقیقاً زمان اجرای بهترین الگوریتم احتمالی را نشان می‌دهد.

به این ترتیب و طبق اصل کمبیشینه $VonNeumann$، در ادامه معنی

$$\max_{p\in\Delta_m}\min_{a\in A} E[T(a,i_p)]$$

را مشخص می‌کنیم. این یعنی انتخاب ترتیب p از ورودی I که با i_p نشان داده شده، و امید داشتن به اینکه بدترین زمان اجرا برای هر الگوریتم مشخص ممکن بدست آید. در نظر داشته باشید که

$$E[T(a_q,i_p)] = \sum_{ij} p_i t_{ij} q_j = pTq$$

که $t_{ij} = T(a_j,i_i)$ می‌باشد. نگاه کردن الگوریتم از دید بازیکن ستونی و ورودی از دید بازیکن سطری و زمان اجرا از دید سود و در نتیجه استفاده از قضیه کمبیشینه می‌تواند ما را به نتیجه زیر برساند که

$$\min_q\max_p E[T(a_q,i_p)] = \max_p\min_q E[T(a_q,i_p)]$$

بر اساس این نکته، ما می‌توانیم به نتیجه زیر بسنده کنیم، که به آن اصل کمبیشینه Yao گفته می‌شود. این اصل می‌گوید که زمان اجرای بهترین الگوریتم قابل مشخص کردن برای هر ترکیبی از ورودی یک حد پایین برای هر الگوریتم احتمالی برای هر ورودی خاص می‌باشد.

قضیه ۷-۲ .(اصل کمبیشینه Yao)

برای هر $p\in\Delta_m$ و هر $q\in\Delta_n$ داریم:

$$\min_{a\in A} E[T(a,i_p)] \le \max_{i\in I} E[T(a_q,i)]$$

اثبات. این قسمت ساده قضیه $Duality$ می‌باشد:

$$\min_{a\in A} E[T(a,i_p)] \leq \max_{p} \min_{a\in A} E[T(a,i_p)] = \max_{p} \min_{q} E[T(a_q,i_p)]$$

$$= \min_{q} \max_{p} E[T(a_q,i_p)] = \min_{q} \max_{i\in I} E[T(a_q,i)] \leq \max_{i\in I} E[T(a_q,i)]$$

و قضیه ثابت می‌شود. □

با به کار بردن این اصل، ما می‌توانیم ترکیبی را انتخاب کرده و یک حد پایین برای زمان اجرای همه الگوریتم‌های احتمالی در بدترین ورودی بدست آوریم. کاهش به حد پایین برای الگوریتم‌های قابل تشخیص به این معنی است که ما می‌توانیم الگوریتم بهنیه قابل تشخیص را آنالیز کنیم در حالی که برای الگوریتم‌های احتمالی نمی‌توان این کار را کرد. همچنین از آنجایی که این بخش ساده قضیه $Duality$ می‌باشد، بخش قوی آن به ما می‌گوید که یک ترکیب روی ورودی‌ها وجود دارد که بهترین الگوریتم قابل تشخیص دقیقاً بازدهی برابر بهترین الگوریتم احتمالی روی همه ورودی‌ها را دارد.