← اگر بخواهیم اجازه دهیم یک الگو چندین بار تکرار شود، به جای $x_t \in \{0,1\}$، عبارت $x_t \in \{0,1,\dots,k\}$; int را می‌نویسیم.

$t=1,\dots,N$

$$x_t = \begin{cases} 1 ; & \text{اگر با الگوی } t \text{ برش داده شود} \\ 0 ; & o.w \end{cases}$$

$$\text{پارامتر} \rightarrow d_{it} = \begin{cases} 1 ; & \text{تقاضای } i \text{ ام رو الگوی } t \text{ ام} \\ 0 ; & \end{cases}$$

$$\text{Min} \sum_{t=1}^{N} x_t$$

$$\text{s.t.} \quad \sum_{t=1}^{N} a_{it} \cdot x_t \geq 1 \quad ; \; i=1,\dots,I$$

$$\text{الزامی نیست} \rightarrow \left(\sum_{t=1}^{N} x_t \leq k \right)$$

$$(x_t \in \{0,\dots,k\}) \text{ یا } (x_t \in \{0,1\})$$

* تعداد متغیرهای مدل فوق، بسیار زیاد است زیرا تعداد الگوهایی که می‌توانیم با آن‌ها برش بزنیم، می‌تواند خیلی زیاد باشد ⟸ لذا در این جا بحث Column generation مطرح می‌شود *

⟸ ایده‌ی اصلی این روش این است که می‌گوید اگر مثلاً 500,000 متغیر داریم، ابتدای کار با تعدادی متغیری که حس بهتری داشته باشیم کار خودمان را شروع

می کنیم و سپس بررسی می کنیم آیا ستونی وجود دارد که اگر آن را به مسئله وارد کنیم جواب مدل بهتر شود یا خیر ؟؟

← به زیرمسئله‌ی مربوطه، Pricing problem گویند ← زیرا بر روی shadow price ها کار می کند.

* Branch and price :

اگر زیرمسئله‌ی Pricing problem (تعیین آن که کدام یک از ستون‌های توانند جواب مسئله را بهبود دهند) با Branch and Bound ترکیب شود، اصطلاحاً به آن Branch and price گویند ← یعنی در هر Node در درخت Branch and Bound، از column generation استفاده می کنیم. ← پس ایده‌ی کلی آن است که یک برنامه‌ریزی عدد صحیح داریم که تعداد متغیرهایش به صورت فزاینده‌ای زیاد است و مجبوریم در هر Node از column generation استفاده کنیم *

* مثلاً ابتدا فقط با pattern های یک‌عضوی و دوعضوی شروع می کنیم و بررسی می کنیم که آیا جواب حاصل، بهینه است یا خیر ؟؟ حال اگر بهینه نیست کدام ستون را به مسئله اضافه کنیم که جواب بهتری بدست آید ؟

اگر دوگان مسئله‌ی قبلی را بنویسیم، تعداد محدودیت‌هایش فوق العاده زیاد و تعداد می
متغیرهایش محدود است. لذا ستونی که می تواند در جواب مسئله بهبود ایجاد کند معادل همان محدودیت نقض شده در مسئله‌ی دوگان است *

* Constraint generation = Row generation *

← چون در column generation، زیرمسئله به نام Pricing problem شناخته می شود، ما نیز به الگوریتمی که از آن بهره می گیرد، Branch and price گوییم.

* Branch and cut and price :

* هم تعداد متغیرها و هم تعداد محدودیت‌ها فوق العاده زیاد است. ← پس در LP، ابتدا متغیرها را فیکس می کنند و محدودیت‌ها را اضافه می کنند، پس وقتی محدودیت‌ها را اضافه کرد، به سراغ متغیرها رفته و آن‌ها را اضافه می کند، پس دوباره به سراغ محدودیت‌های بعد و ... ⇐ این چرخه آن قدر ادامه می یابد که هم CUT و هم Pricing problem متوقف شوند *

Maral

* جلسه‌ی هفدهم:

* دوگان مدل صفحه‌ی ۲۹:

- ابتدا مدل ریلکس شده را می‌نویسیم:

$$\text{Min} \sum_{t=1}^{N} x_t$$

S.t.:

$$\sum_{t=1}^{N} \alpha_{it} \cdot x_t \geq 1 \quad ; \forall i = 1, \dots, n \qquad u_i$$

$$0 \leq x_t \leq 1 \xrightarrow[\text{Min سازی است}]{\text{چون هستند}} 0 \leq x_t \quad ; \forall t = 1, \dots, N \qquad v_t$$

x_t بزرگتر از ۱ نفی شود

* دوگان مسئله‌ی فوق:

$$\text{Max} \sum_{i=1}^{n} u_i - \sum_{t=1}^{N} v_t$$

S.t.:

$$\sum_{i=1}^{n} \alpha_{it} u_i - v_t \leq 1 \quad ; \forall t = 1, \dots, N$$

$$u_t, v_t \geq 0 \quad ; \forall t = 1, \dots, N$$

← حال چون ضریب v_t، منفی است و تابع هم Max سازی است؛ خود تابع v_t را صفر می‌کند. پس دوگان مسئله به طور خلاصه عبارت است از:

$$\text{Max} \sum_{i=1}^{n} u_i \qquad (2)$$

$$\text{s.t.} \left| \begin{array}{ll} \sum_{i=1}^{n} \alpha_{it} \cdot u_i \leq 1 & ; \forall t = 1, \dots, N \\ u_t \geq 0 & \end{array} \right.$$

* حال به جای آن که بخواهیم مسئله را برای کل N، حل کنیم، مسئله اصلی را به ازای N' که تعداد کمتری از N دارد حل می کنیم. یعنی:

$$\text{Min} \sum_{t=1}^{N'} x_t \qquad (1)$$

$$\text{s.t.} \left| \begin{array}{ll} \sum_{t=1}^{N'} \alpha_{it} \cdot x_t \geq 1 & ; i = 1, \dots, n \qquad \bar{u}_i \\ x_t \geq 0 & ; t = 1, \dots, N' \end{array} \right.$$

* حال فرض می کنیم جوابی که از مدل فوق به دست می آید $\hat{x}_t$ باشد و برای $(N-N')$ حالت دیگر نیز جواب را صفر در نظر می گیریم.

$$\left| \begin{array}{ll} x_t = \hat{x}_t & t = 1, \dots, N' \\ x_t = 0 & t = N'+1, \dots, N \end{array} \right.$$

* یک ایده برای کم کردن تعداد متغیرها از N به N'، آن است که فقط الگوهای یک تایی و دوتایی را در نظر بگیریم (آن هایی که feasible باشند)

(shadow price)

← اکنون باید دوگان مدل قبل ($\overline{u}_i$)، در تمام محدودیت‌های مدل دوگان گفته شده (یعنی همه‌ی N محدودیت و نه فقط N' از آن‌ها) صدق کند، می‌دانیم که در N' محدودیت، اول صدق می‌کند و α_{it} آن‌ها را داریم ولی بقیه‌ی α_{it} ها را نداریم و نمی‌دانیم که $\overline{u}_i$ در آن‌ها صدق می‌کند یا خیر.

★ براساس قضیه‌ی دوگان قوی:

← به عبارتی اگر بخواهیم بنویسیم $\begin{cases} x_t=\hat{x}_t \ ; \ t=1,\dots,N' \\ x_t=0 \ ; \ t=N'+1,\dots,N \end{cases}$ برای مدل ① بهینه است

باید shadow price مربوط به محدودیت آن در تمام محدودیت‌های مدل ② صدق کند. (در همه‌ی N محدودیت) ⟸ ولی چون همه‌ی α_{it} ها را نداریم، کافیست مدل زیر (کوله پشتی) حل شود:

- شناسایی محدودیتی که نقض می‌شود یعنی شناسایی α_{it} مربوط به آن.

- α_{it} ها، هر یک مرتبط با یک طرح هستند، و هر طرح خصوصیتی که دارد آن است که مجموع عرض‌های آن باید از L، کوچکتر مساوی باشد. حال u_i را به $\overline{u}_i$ و α_{it} را به z_i تبدیل می‌کنیم. حال می‌خواهیم بررسی کنیم که آیا ممکن است یک z_i پیدا کنیم که در این مدل صدق کند؟ حال اگر جواب مسئله‌ی زیر، بزرگتر از یک شود یعنی z_i مربوط به آن، همان الگویی است که ما به دنبال آن هستیم. (در اصل ما از قبل α_{it} را نداریم (هیچ الگوی خاصی مدنظر ما نیست) و با حل این مدل یکی از الگوهایی که مدل دوگان را نقض می‌کند به دست می‌آوریم)

$$\text{Max} \sum_{i=1}^{n} \overset{\alpha_{ist}}{z_i} \cdot \overline{u}_i$$

$$\text{s.t.} \quad \sum_{i=1}^{n} z_i \cdot a_i \le L$$

$$z_i \in \{0,1\} \quad ; \forall i=1,\dots,n$$

★ اگر جواب مسئله‌ی فوق، کوچکتر یا مساوی یک شود، یعنی تمام محدودیت‌های مدل دوگان ارضا شده‌اند، ولی اگر جواب مسئله‌ی فوق از یک بزرگتر شود، یعنی محدودیتی نقض شده است، لذا ★Z حاصل از آن دقیقاً همان α_{it} ها نشان می‌دهد که مجموعه‌ی آن‌ها سبب نقض شدن محدودیت ($\sum \alpha_{it} u_i \le 1$) می‌شوند★

Maral!

Subject:
Yare: Month: Day:
Page: ( )

* حال مدل کوله پشتی صفر و یک را می‌توان با الگوریتم polynomial time در زمان چند جمله ای (البته با استفاده از برنامه ریزی پویا) حل نمود *

قضیه ی قوی دوگان:

اگر یک جواب شدنی برای مسئله ی primal داشته باشیم و جواب های دوگان مدل نیز در مدل Dual؛ feasible باشد و مقدار تابع هدف مدل primal و Dual نیز برابر باشند ← جواب به دست آمده حتماً بهینه است *

* جلسه ی هفدهم:

* جلسه ی نوزدهم:

Maral

حال مسئله کوله پشتی صفر و یک را می‌توان با الگوریتم polynomal time در زمان چند جمله ای (البته با استفاده از برنامه ریزی پویا) حل کرد

قضیه‌ی قوی دوگان:

اگر یک جواب شدنی برای مسئله‌ی primal داشته باشیم و جواب های دوگان مسئله نیز در مسئله Dual، feasible باشد و مقدار تابع هدف، مسئله Primal و Dual نیز برابر باشند ← جواب به دست آمده حتماً بهینه است*

* جلسه‌ی هفدهم:

—

* جلسه‌ی نوزدهم:

* شرایط کافی برای تشخیص آن که یک چندوجهی Integral هست یا خیر؟؟

- مثلاً از این شرط در مسئله‌ی حمل و نقل (Transportation) استفاده می‌شود تا اثبات کنیم چندوجهی آن عدد صحیح است*

- ماتریس TU:

← یک ماتریس TU است اگر تمام زیردترمینان های آن 0، 1 یا -1 باشد*

- یعنی ابتدا زیر ماتریس مربوط ماتریس اصلی را به دست آوریم و سپس دترمینان آن را نیز حساب کنیم ← که در این صورت به آن زیر دترمینان گویند*

* ماتریس psd:

- یک ماتریس psd است اگر و فقط اگر به ازای هر $x \in \mathbb{R}^n$ ، $x^T A x \geq 0$ باشد. یعنی:

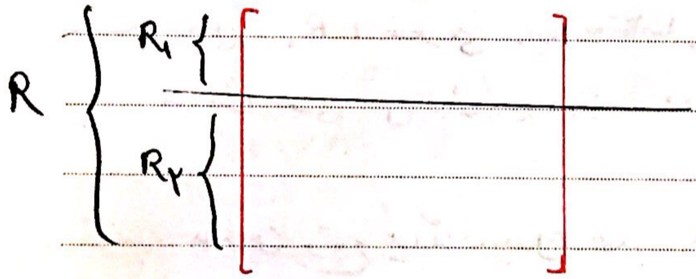
$$\text{Psd} \Rightarrow \forall x \in \mathbb{R}^n \Rightarrow x^T A x \geq 0 \equiv \lambda_i (A + A^T) \geq 0 \;; \forall i$$

* شروط لازم و کافی برای TU بودن یک ماتریس:

- ماتریس $A_{m\times n}$، TU است، اگر و فقط اگر به ازای هر $R \subseteq \{1,\dots,m\}$
بتوان R را به نحوی افراز کرد که:

$$\left[\begin{array}{c} R_1 \cup R_2 = R \\ R_1 \cap R_2 = \emptyset \\ \text{می توانند تهی باشند } R_2 \text{ و } R_1 \end{array}\right]$$

$$\forall j = 1,\dots,n \qquad \sum_{i\in R_1} a_{ij} - \sum_{i \in R_2} a_{ij} \in \{-1, 0, +1\}$$



مثال: فرض کنید A، ماتریس وقوع یک گراف جهت دار باشد که یال موازی هم ندارد؛ در این صورت R_1 را برابر با کل A و R_2 را برابر با $\emptyset$ در نظر می گیریم $\Rightarrow$ با این تفاسیر شرط بالا برقرار خواهد شد و لذا TU می باشد. (البته حتی با وجود یال های موازی نیز، TU خواهد ماند زیرا برای هر یال، به طور جداگانه در ماتریس وقوع، ستون تعریف می شود)

* شرط لازم برای TU بودن یک ماتریس، آن است که تمام عناصر آن، صفر، 1- یا 1 باشد؛ چرا که هر عنصر، خودش یک ماتریس مربعی 1x1 می باشد*

- ضرایب محدودیت ها در ماتریس حمل و نقل نیز TU می باشد. محدودیت های این مسئله عبارت است از:

$$\begin{cases} \sum_{i=1}^{m} x_{ij} = d_j & ; \forall j = 1,\dots,n \\ \sum_{j=1}^{n} x_{ij} = c_i & ; \forall i = 1,\dots,m \end{cases}$$

- ماتریس ضرایب شامل $(m+n)$ سطر می‌باشد.

مثال: به عنوان مثال یک مسئله‌ی حمل و نقل با ۲ مقصد و ۳ مبدأ را در نظر می‌گیریم داریم:

$$x_{11}+x_{21}+x_{31}=d_1$$
$$x_{12}+x_{22}+x_{32}=d_2$$
$$x_{11}+x_{12}=c_1$$
$$x_{21}+x_{22}=c_2$$
$$x_{31}+x_{32}=c_3$$

مرتب شده می‌نویسیم ⟶

$$R_1\left\{\begin{array}{l} x_{11}+x_{21}+x_{31}=d_1 \\ x_{12}+x_{22}+x_{32}=d_2 \end{array}\right.$$
$$x_{11}+x_{12}=c_1$$
$$x_{21}+x_{22}=c_2$$
$$x_{31}+x_{32}=c_3$$

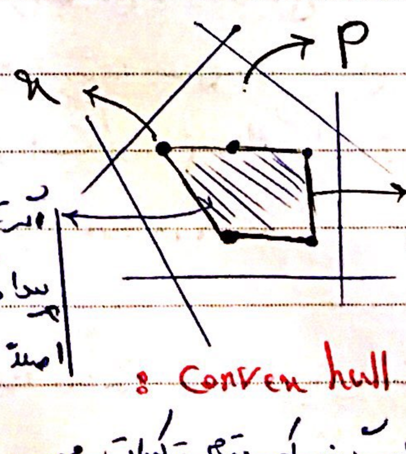
← حال R_1 را سطرهای مربوط به تقاضا و R_2 را سطرهای مربوط به عرضه در نظر می‌گیریم که شرط صفحه‌ی قبل برای آن برقرار خواهد بود ⟸ لذا TU است.

قضیه: ماتریس وقوع یک گراف بدون جهت TU است اگر و فقط اگر آن گراف دوبخشی باشد. (از طرفی گراف مسئله‌ی حمل و نقل نیز دوبخشی است)

چند وجهی Integral از Wolsey ⟶

$X=\{x \in Z^n : x \in P\}$ که آن‌ها را نوشته‌ایم. ⟶ یک سری محدودیت است

$Con(X) \subseteq P$ ⟶ دوست داریم $con(X)$ دقیقاً با P برابر باشد.



ما به دنبال پیدا کردن چنین مجموعه‌ای هستیم.

اگر تابع هدف خطی باشد؛ نقاط گوشه‌ای این پوسته اهمیت پیدا می‌کند ولی اگر تابع هدف غیرخطی باشد؛ این پوسته‌ی Integral اصلاً به درد نمی‌خورد زیرا جواب بهینه در این حالت، لزوماً نقطه گوشه‌ای نیست.

* Convex hull :

به مجموعه‌ای گویند که تمام ترکیبات محدب هر دو نقطه از آن، درون خود آن مجموعه قرار گیرد. *

قضیه: ماتریس $A_{m\times n}$ ، TU است اگر و فقط اگر چندوجهی $P=\{Ax\le b, x\ge 0\}$
به ازای همه‌ی $b\in Z^n$ ، صحیح باشد* (شرط لازم و کافی)

← استفاده‌ای که از این قضیه می‌شود آن است که اگر تمام bها صحیح و A نیز TU باشد
آن‌گاه چندوجهی حاصل از $\{Ax\le b, x\ge 0\}$ نیز Integral خواهد شد*

قضیه: اگر ماتریس A ، TU باشد آن‌گاه چندوجهی $P=\{Ax=b, x\ge 0\}$
به ازای همه‌ی $b\in Z^n$ ، صحیح می‌باشد. (شرط کافی)

همه‌ی نقاط گوشه‌ای متناظر با یک جواب اساسی شدنی می‌باشد ← برای اثبات، از این قضیه استفاده می‌شود

$$x=\begin{pmatrix} x_B \\ x_N \end{pmatrix}=\begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$$

$$B^{-1}=\frac{1}{\det(B)}\left(\begin{bmatrix} M_{ij} \end{bmatrix}\times(-1)^{i+j}\right)$$

سطر و ستون i و j را حذف می‌کنیم و سپس دترمینان ماتریس
باقی‌مانده را حساب می‌کنیم ← چون TU است لذا همواره 0 و ±1 خواهد بود، پس
در $(-1)^{i+j}$ هم ضرب شود باز هم 0 و ±1 خواهد ماند. لذا B^{-1} عدد صحیح می‌شود
و اگر b نیز عدد صحیح باشد بنابراین $B^{-1}b$ نیز عدد صحیح باقی خواهد ماند. پس
چندوجهی Integral خواهد ماند*

* جلسه بیستم:

Constraint programming:

← یک حوزه است که در computational computer science (علم کامپیوتر مبتنی بر محاسبات)
برای اولین بار ایجاد شد ← می‌خواهیم بدون آن که ریاضیات زیادی بلد باشیم، بتوانیم مسئله را
حل کنیم*

← مسئله ی CP یک Discrete Optimization است، زیرا همه ی متغیرهایی که در آن تعریف می شوند گسسته هستند*

← در این برنامه ریزی سعی می شود محدودیت های معمول را در یک فضای کاملا user friendly بنویسیم، به همین دلیل به آن برنامه ریزی محدودیتی گویند.

* در IP، lower bound داریم (یعنی در آن ساختار کران ها را می دهد) ولی در CP، کران بایستی محاسبه باشد.

* CP، توسط جست و جوی مستقیم همگرایانه عمل می نماید*

* Set function :

* تاکنون function هایی که می خواندیم، عدد می دادیم و پس عدد می گرفتیم
یعنی: real valued-real function*

← ولی set function ها، مجموعه می گیرند و عدد پس می دهند.
یعنی: real valued-set function*

$f: 2^E \to R$

در این حالت، خود x، تابعی از S است

$$\min f(S) \quad \Rightarrow \quad \min f(x)$$
$$\left| \begin{array}{l} S \in K \\ K \subseteq 2^E \end{array} \right. \qquad x \in X$$

← مثلاً مسئله ی Knapsack را می توان در قالب set function مدل کرد زیرا به دنبال یافتن زیرمجموعه ای از کالاها هستیم که هم حجم آن از مقدار مجاز تجاوز نکند و هم بیشترین ارزش را داشته باشد.

Subject:

Yare: Month: Day: Page: ( )

* جلسه‌ی بیست و یکم:

$f: 2^E \rightarrow \mathbb{R}$ فرض کنید مجموعه‌ای مانند E داریم و f از 2^E به R تعریف می‌شود:

پیمانه‌ای $\forall X,Y \in 2^E \quad f(X) + f(Y) = f(X \cup Y) + f(X \cap Y)$

(۱) زیرپیمانه‌ای $\forall X,Y \in 2^E \quad f(X) + f(Y) \geqslant f(X \cup Y) + f(X \cap Y)$

زبرپیمانه‌ای $\forall X,Y \in 2^E \quad f(X) + f(Y) \leqslant f(X \cup Y) + f(X \cap Y)$

* تعریف: تابع f بر روی مجموعه‌ی محدب S، محدب گفته می‌شود اگر:

$\forall x,y \in S, \ \alpha \in [0,1]$

f ترکیب محدب هر دو نقطه از S، کوچکتر مساوی ترکیب محدب خود f باشد* $\rightarrow f(\alpha x + (1-\alpha) y) \leqslant \alpha f(x) + (1-\alpha) f(y)$

- بررسی شرط فوق، بسیار مشکل خواهد بود. بنابراین تعاریف فوق را به شکل زیر تغییر می‌دهیم:

[مطلوبیتی که اضافه کردن e به S خواهد داشت برابر است با مطلوبیت حاصل از اضافه کردن e به T

پیمانه‌ای $\forall S \subset T \subset E, \forall e \in E \setminus T \quad f(S+e) - f(S) = f(T+e) - f(T)$

(۲) زیرپیمانه‌ای $\quad '' \quad f(S+e) - f(S) \geqslant f(\underbrace{T+e}_{=T \cup \{e\}}) - f(T)$

~~زبرپیمانه‌ای $\quad '' \quad f(S+e) - f(S) \leqslant f(T+e) - f(T)$~~

- حال برای حل مسائل مختلف؛ اعم از اثبات آن که یک تابع زیرپیمانه ای است یا پیمانه ای یا ... باید از تعریف های مجموعه ی دوم (اخیر) استفاده کنیم *

* اثبات تعاریف دوم: (می خواهیم اثبات کنیم مجموعه تعاریف ۱ و ۲ معادل یکدیگرند)

۲ ⟵ ۱

- اگر در تعاریف ابتدای کار داشته باشیم:

$X = S + e$

$Y = T$

داریم:

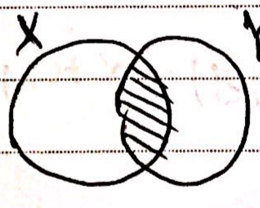
$$f(X) + f(Y) = f(X \cup Y) + f(X \cap Y)$$

$$\Rightarrow f(S+e) + f(T) = f((S+e) \cup (T)) + \cancel{f((S+e) \cap T)}$$

$$f(S+e) + f(T) = f(S+e) + f(T) + \emptyset$$

۱ ⟵ ۲ (چگونه می رسیم از مجموعه تعاریف ۲ به ۱)

- در این حالت داریم:



فرض:
$T = X$
$S = X \cap Y$
$S \subset T$

$Y \setminus X = \{e_1, \dots, e_k\}$

$$f(T + e_1) - f(T) \leq f(S + e_1) - f(S)$$

$$\Rightarrow f(X + e_1) - f(X) \leq f(X \cap Y + e_1) - f(X \cap Y)$$

$$\Rightarrow f(X \cap Y) - f(X) \leq f(X \cap Y + e_1) - f(X + e_1)$$ الف

$$\Rightarrow f(X+e_1+e_2)-f(X+e_1) \leq f(X\cap Y+e_1+e_2)-f(X\cap Y+e_1)$$

ب

$$\Rightarrow f(X\cap Y+e_1)-f(X+e_1) \leq f(X\cap Y+e_1+e_2)-f(X+e_1+e_2)$$

* از الف و ب نتیجه می شود:

$$f(X\cap Y)-f(X) \leq f(X\cap Y+e_1+e_2)-f(X+e_1+e_2)$$

- حال اگر همین ترتیب این روند را ادامه دهیم خواهیم داشت:

$$f(X\cap Y+e_1+e_2+\dots+e_{k-1})-f(X+e_1+\dots+e_{k-1}) \leq \underbrace{f(X\cap Y+Y\setminus X)}_{f(Y)}-\underbrace{f(X+Y\setminus X)}_{f(X\cup Y)}$$

- اثبات تکمیل است.

سوال: اثبات کنید حاصل تبدیل زیر یک تابع زیر پیمانه‌ای خواهد بود:

$f(S)=g(|S|)$

تابع مقعر

* جلسه‌ی بیست و دوم:

Super Modular = زبر پیمانه‌ای

Sub Modular = زیر پیمانه‌ای

Modular = پیمانه‌ای

(۳۵)

Subject:
Yare: Month: Day: Page: ( )

← اگر مسئله‌ی تصمیم‌گیری به فرم Sub Modular Minimization نوشت می‌شود برای آن یک الگوریتم طراحی کرد که آن را در زمان چندجمله‌ای P حل کند.

Sub Modular Maximization ⟸ مسئله‌ی کوری پیش

۲- فرم کلی مسئله‌ی Set Function :

$$\text{Min } f(S) \qquad \text{Max } f(S)$$

$$S \subseteq E \qquad S \subseteq E$$

$$f: 2^E \to R \qquad f: 2^E \to R$$

⇓ ⇓

اگر Sub Modular باشد، به خوبی حل می‌شود

این مسئله را به راحتی نمی‌توان حل نمود.

← باید ببینیم می‌توانیم مسئله را به فرم Set Function تبدیل کنیم یا خیر؟؟

* Profit Maximization Facility Location (بدون در نظر گیری ظرفیت برای facility ها)

یک سری (مجموعه‌ای) از costumer ها داریم و می‌خواهیم مجموعه‌ای از تسهیلات را ایجاد کنیم تا سود خدمت دهی ؟ max شود. (به عبارتی در این نوع مسئله، اجباراً تمام مشتری‌ها ؟ باید خدمت دهی شوند) ⟸ می‌خواهیم از بین set of potential Location service ها ؟ تسهیلات را در مکان‌هایی قرار دهیم که سود حاصل از خدمت دهی max شود* (سود حاصل از هر مشتری متفاوت خواهد بود)

⟸ به دنبال یافتن f(S) برای این مسئله هستیم.

مثلاً در مسئله‌ی کوله پشتی به فرم زیر است:

$$f(S) = \sum_{i \in S} c_i \rightarrow$$ همان‌طور که مشخص است یک تابع ماژولار می‌باشد؛ چرا که اگر یک e به آن اضافه شود مطلوبیت آن بهبود می‌یابد *

زیرمجموعه‌ای از آیتم‌ها

$S \subseteq A = \{1, \dots, n\}$

Maral

* در مسئلهٔ generilized Assignment problem با داشتن n کالا و m ظرف می‌توانیم set function را به صورت زیر تعریف کنیم:

$$f(S_1,\dots,S_m)=\sum_{j=1}^{m}\underbrace{c(S_j)}_{\sum_{t\in S_j} c_t}$$

این تابع هدف ماژولار است و همچنین یکنواست زیرا هر چقدر S افزایش یابد میزان f نیز افزایش می‌یابد.

s.t.:
$S_1\subseteq\{1,2,\dots,n\}$
$S_2\subseteq\{1,2,\dots,n\}$
$\vdots$
$S_m\subseteq\{1,2,\dots,n\}$
$S_i\cap S_j=\phi \;;\; i\neq j$

$\underbrace{\omega(S_j)}_{\sum_{t\in S_j}\omega_t}\leq c_j \quad ; j=1,\dots,m \longrightarrow$ از ظرفیت ظرف تجاوز نشود

$S_1=$ مجموعه‌ای از اعضای اتا n هستند که به ظرف اول؛ تخصیص یافته اند*

$S_2=$ 〃 〃 〃 〃 〃 〃 〃 دوم؛ 〃 〃 *

$\vdots$

$\xrightarrow{\text{یک تابع یکنوا}} S\subseteq T \Rightarrow f(S)\leq f(T)$

* Profict Maximization Uncapcitated Facility-Location problem :

$S=$ مجموعه‌ای از نقاط کاندید برای استقرار تجهیز $(S\subseteq E)$

$R=$ مجموعه‌ی مشتریان بالقوه

$c_e=$ هزینه‌ی احداث مرکز eام

$b_{re}=$ سود حاصل از سرویس دهی به مشتری rام از طریق مرکز eام *

~~* مشتریان را به مرکزی تخصیص می‌دهیم که سود حاصل از آن max شود*~~

Maral

Subject:
Yare: Month: Day:

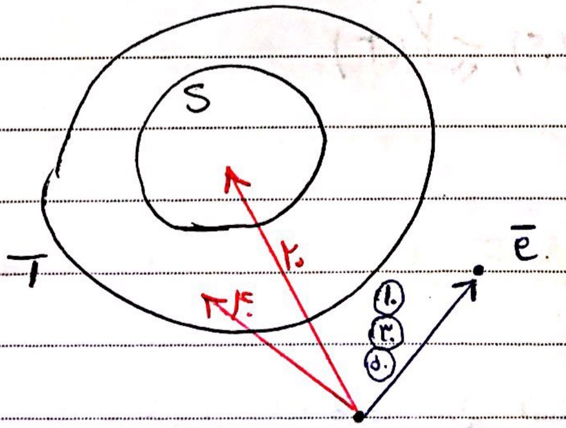
$$\max_{S\subseteq E} f(S) = \sum_{r\in R} \max_{e\in S}\{b_{re}\} - \sum_{e\in S} c_e$$

* ثابت کنید تابع $f(S)$، از نوع Sub Modular است.

← از تعریف مطلوبیت استفاده می‌کنیم:

- اگر یک facility جدید اضافه شود ($\bar{e}$)، هم به مجموعه‌ی S و هم به مجموعه‌ی T، یک هزینه‌ی ثابت استاندارد، تغییر $c(\bar{e})$ اضافه می‌شود
پس هزینه‌ی ناشی از اضافه شدن چیز جدید هم در S و هم در T یکسان است.

- ولی اضافه کردن عضو جدید، احتمال آن که تخصیص مجموعه‌ی S را تغییر دهد، بسیار بیشتر از آن است که تخصیص مجموعه‌ی T را تغییر دهد.



- حال تخصیص به $\bar{e}$، ۳ حالت دارد:
(۱) یا برابر با (۱۰) است:
هیچ یک از دو تخصیص را تغییر نمی‌دهد* ← ۱۰ > ۲۰ > ۴۰

(۲) یا برابر با (۳۰) است:

۲۰ < ۳۰ < ۴۰ → بنابراین تخصیص در S را تغییر می دهد
ولی نمی تواند سبب تغییر تخصیص در T شود ⭑

(۳) یا برابر با (۵۰) است:

۲۰ < ۴۰ < ۵۰ → هر دو تخصیص S و T را تغییر می دهد
ولی مطلوبیت تغییر تخصیص از S نسبت به T بیشتر است؛ زیرا:

(۵۰ - ۲۰) > (۵۰ - ۴۰)

⇐ پس اگر اعداد فوق را با a، b، c و d به طور کلی جایگذاری کنیم و باز بررسی نماییم، چون مطلوبیت تابع S [$S \subseteq T$] به ازای افزایش یک تجهیز بیشتر است لذا تابع $f(S)$، SubModular می باشد ⭑

* جلسه ی بیست و سوم:

* Min S-t cut :

$$\min f(S+s)$$

$$S \subseteq V \setminus \{s,t\}$$

$$f(V) = \sum_{e \in E(V,\bar{V})} c_e$$

* با همان روند قبلی می توان اثبات کرد که تابع فوق هم SubModular است.

Subject: Page: ( )
Yare: Month: Day:

- اگر مجموعه ای نظیر E داشته باشیم، به آن گاه $F \subseteq 2^E$ را یک مجموعه‌ی مستقل گوییم، اگر تحت عملگر زیرمجموعه بسته باشد. ⟸ یعنی اگر $A \in F$ باشد، هر زیرمجموعه ای از A نیز عضو F باشد.

- مثلاً در یک گراف، مجموعه‌ی زیرگراف های آن 2^E می باشد. حال مثلاً زیرگراف یک جنگل نیز، خودش یک جنگل است، لذا مجموعه‌ی زیرگراف های جنگلی یک گراف، یک مجموعه‌ی مستقل است ⟸ یعنی تحت عملگر زیرمجموعه گیری بسته باشد (subset operator) ⟸ یعنی اگر در این مجموعه، یک A ای خصوصیت خاصی دارد، تمام زیرمجموعه های آن نیز، همان خصوصیت را داشته باشد.

* مجموعه‌ی ستون های یک ماتریس را در نظر بگیرید (مثلاً مجموعه‌ی E که 2^E تا زیرمجموعه دارد) اکنون به دنبال مجموعه‌ی بردارهایی هستیم که مستقل خطی باشند (می دانیم اگر یک مجموعه ای از بردارهای مستقل خطی داشته باشیم، حتماً تمام زیرمجموعه های آن نیز از یکدیگر مستقل خطی هستند)

⟵ اگر ماتریس 500 ستون و 100 سطر داشته باشد، بزرگترین مجموعه‌ی مستقل نهایتاً می تواند 100 عضو داشته باشد* (زیرا باید ابعاد همگی مربعی باشد ⟵ حداکثر 100×100)

* در فضای $\mathbb{R}^n$، نهایتاً می توان n بردار مستقل ایجاد کرد*

⟸ پس به طور کلی مجموعه ای از زیرمجموعه های E را یک Independent Set (مجموعه‌ی مستقل) می نامیم اگر تحت عملگر زیرمجموعه گیری بسته باشد. به E و F نیز یک سیستم مستقل گویند*

Subject: Page: ( )
Yare: Month: Day:

* ماتروید‌ها (Matroid) :

- همان مجموعه های مستقل با یک خصوصیت اضافه تر هستند :

(i) - این که اگر ۲ مجموعه باشند که یکی تعدادش از دیگری بیشتر باشد، (مثلاً F یک مجموعه‌ی مستقل است به گونه ای که x و y عضو F هستند و x تعداد بیشتری از y دارد) ⟸ آن گاه حتماً یک چیزی در x هست که در y نیست و اگر آن را به y اضافه کنیم؛ مجدداً مجموعه‌ی جدید که حاصل می شود در F خواهد بود *

- مثلاً مجموعه‌ی زیرگراف های جنگلی یک گراف و یا مجموعه‌ی زیرمجموعه های مستقل سطرهای یک ماتریس؛ این خصوصیت را دارا هستند *

(به عنوان مثال مجموعه ای شامل ۹۰ ستون و مجموعه‌ی دیگر شامل ۱۰ ستون داریم که اتفاقاً هر دو درون سیستم خودشان نیز هستند و مستقل اند؛ مطمئناً سطری در مجموعه‌ی اول وجود دارد که اگر به مجموعه‌ی ۱۰ تایی اضافه شود، مجموعه‌ی حاصل همچنان مستقل باقی خواهد ماند.)

(یا اگر یک گراف داشته باشیم که جنگل باشد (A) و یک گراف جنگل دیگر با تعداد یال های کمتر دیگر نیز داشته باشیم (B)؛ حتماً یکی از یال های A را می توان به B اضافه کرد به گونه ای که مجموعه‌ی حاصل درخت باقی بماند)

- این ماتروید ها؛ یک سری ویژگی های خاصی دارند که مثلاً بهینه سازی روی آن‌ها را در زمان چند جمله ای می توان انجام داد.

⟸ پس اگر F؛ مجموعه ای از زیر مجموعه های E باشد آن گاه (F,E) را یک مجموعه‌ی مستقل نامیم اگر نسبت به عملگر زیرمجموعه بسته باشد و در نهایت یک ماتروید هستند اگر یک خصوصیت اضافه تر نیز داشته باشند.