★ خوشه :

یک زیرمجموعه ای از یک گراف است که بین تمام رئوس در آن یال وجود داشته باشد یا به عبارت دیگر کامل باشد ★

★ مجموعه ی مستقل :

زیرمجموعه ای از یک گراف است که بین هیچ یک از دو راس آن یال وجود ندارد ★

★ عدد خوشه ای = بزرگترین خوشه ای که در یک گراف وجود دارد.

★ عدد استقلال = بزرگترین مجموعه ی مستقل که در یک گراف وجود دارد ★

← خوشه در برابر گراف مادر معنا پیدا می کند ★

★ اعداد رمزی m و n : ($r(m,n)$) :

- کوچکترین عددی که یا یک خوشه ی m تایی در خودش و یا یک خوشه ی n تایی در گراف مکملش وجود دارد ★

$r(3,3) = 6$ ⟵ یعنی اگر ۶ راس داشته باشیم، حتماً می توانیم یا در خودش یک خوشه ی ۳ تایی و یا در گراف مکملش یک خوشه ی ۳ تایی پیدا کنیم

(اثبات مهم است)

★ جلسه ی سوم :

N

- یک سری گراف های خاص را Network گویند ★

- مثلاً یک گراف جهت دار را مثبت درنظر بگیریم و رأس های آن را به ۳ دسته تقسیم کنیم، این رئوس یا منبع یا چاهک یا رئوس میانی هستند. منبع و چاهک را حتماً باید داشته باشیم ★ (ولی وجود رئوس میانی اختیاری است)

* خوشه :

یک زیرمجموعه ای از رئوس در یک گراف است که بین تمام رئوس آن یال وجود داشته باشد یا به عبارت دیگر کامل باشد *

* مجموعه ی مستقل :

زیرمجموعه ای از یک گراف است که بین هیچ یک از دو راس آن یال وجود ندارد *

* عدد خوشه ای = بزرگترین خوشه ای که در یک گراف وجود دارد.

* عدد استقلال = بزرگترین مجموعه ی مستقل که در یک گراف وجود دارد *

← خوشه در برابر گراف مادر معنا پیدا می کند *

* اعداد رمزی m و n : ($r(m,n)$) :

- کوچکترین عددی که یا یک خوشه ی m تایی در خودش و یا یک خوشه ی n تایی در گراف مکملش وجود دارد *

$r(3,3) = 6$ ⟵ یعنی اگر ۶ راس داشته باشیم حتماً می توانیم یا در خودشان یک خوشه ی ۳ تایی و یا در گراف مکملش یک خوشه ی ۳ تایی پیدا کنیم

(اثبات مهم است)

* جلسه ی سوم :

N

- یک سری گراف های خاص را Network * گویند

- مثلاً یک گراف جهت دار را که مثبت در نظر بگیریم و راس های آن را به ۳ دسته تقسیم کنیم. این رئوس یا منبع یا چاهک یا رئوس میانی هستند. منبع و چاهک را حتماً باید داشته باشیم * (ولی وجود رئوس میانی اختیاری است)

← پس شبکه ها، گرافی است که حتماً منبع و چاهک را داشته باشد.

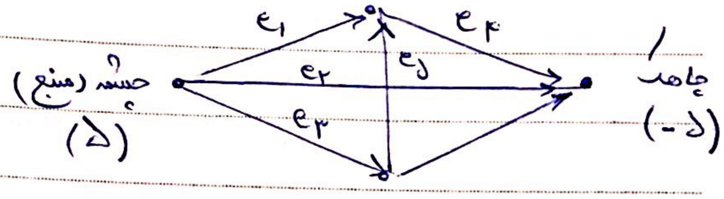
* هر یال نیز یک ظرفیت شارش دارد * (C)

$V = X \cup I \cup Y$

منبع ← X ، رئوس میانی ← I ، چاهک ← Y

$X \neq \phi$

$Y \neq \phi$



* Flow (شارش):

① * شارش ها حتماً باید هم نامنفی باشند و هم از ظرفیت یال ها کمتر باشند *

$0 \leq f_{e_1} \leq c_{e_1}$

② * همچنین باید قانون تعادل در مورد شارش ها برقرار باشد تا بتوانیم به آن "شارش مجاز" بگوییم *

$f_{e_1} + f_{e_2} + f_{e_3} = 5$

$f_{e_1} + f_{e_5} = f_{e_4}$

$\vdots$

* لیست مسائل معروف در گراف تئوری:

① Minimum Spanning tree problem ← همه این ها را می توانیم یک weighted هم بهشان اضافه کنیم.

* طریقه ی بیان مسئله:

یک درخت فراگیر می خواهیم که همه رئوس را پوشش دهد، حال کوچکترین آن را می خواهیم *

② Maximum wheigt forest problem

بنابراین در forest ها باید ماکسیمم شود، اگر بخواهیم آن را minimum کنیم جواب صفر خواهد شد؛ چون تهی نیز یک جنگل به حساب می آید!!!

جلسه ۲۱

۳ Shortest Path problem

- اگر وزن ها مثبت باشد به متعلق به کلاس P است *

۴ Longest Path problem

- این مسئله NP-hard است *

۵ Maximum weighted matching

۶ Minimum weight perfect matching

- مرتبط با مسئله Assignment problem است *
- متعلق به کلاس P

۷ Maximum wheigtd cut problem

- متعلق به کلاس NP-hard

۸ Minimum wheigted cut problem

- متعلق به کلاس P

۹ Traveling Sales man problem

- یک گراف کامل وزن دار یالی داریم و به دنبال دور همیلتونی هستیم که کمترین هزینه را داشته باشد
- متعلق به کلاس NP-hard

- پس زمینه همه VRP ها و مسیریابی ها به همین مسئله TSP است *
- علت این که گراف را کامل می گیرند برای آن است که دور همیلتونی حتماً داشته باشد و اگر یالی هم موجود نباشد به آن را با یک هزینه بسیار زیاد وارد می کنند *

۱۰) Max weighted subgraph planner:

- در layout کاربرد دارد *
- زیرگرافی از یک گراف همبند وزن دار را به دست آوریم که بیشترین وزن را داشته باشد *
- متعلق به کلاس NP-hard

۱۱) Minimum cost flow problem:

- یک شبکه در نظر می‌گیریم که علاوه بر ظرفیت هر یال، هزینه‌ی شارش هر واحد در هر یال در آن مشخص است. یک سری چشمه و چاهک نیز داریم ← دنبال آن هستیم که این شارش را به گونه‌ای انجام دهیم که هزینه های آن، کمینه شود *
- متعلق به کلاس P
- مسئله‌ی Transportation، حالت خاص این مسئله است اگر هیچ راس میانی نداشته باشیم.

۱۲) Maximum cost flow problem:

- متعلق به کلاس P
- یک Network داریم که فقط ظرفیت هر یال را داریم. یک سری چشمه و چاهک هم داریم که می خواهیم به گونه‌ای عمل کنیم که بیشترین جریان در گراف شارش پیدا کند *

* جلسه‌ی هفتم:

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

* جلسه‌ی هشتم:

(——)

* جلسه‌ی نهم:

* راهکارهای convex کردن (بررسی محدب‌سازی)

- در مسائل Location-Inventory با انبارگردانی دوره‌ای، مدل به صورت زیر است:

$$\min \underbrace{c^T x}_{\text{خطی}} + \underbrace{\sqrt{b^T y}}_{\text{مقعر}}$$

مقعر

$y \in \{0,1\}$

$b \geq 0$

همان‌گونه که مشخص است تابع هدف فوق یک تابع مقعر است، پس ما باید با استفاده از راهکاری آن را به یک تابع محدب تبدیل کنیم.

$\xrightarrow{\text{چون}} y_i \in \{0,1\} \xrightarrow{\text{پس}} \forall i \quad y_i = y_i^2$

$$\xrightarrow[\text{مدل محدب با استفاده از جایگزینی فوق}]{\text{تبدیل مدل فوق به یک}} \min \underbrace{\underbrace{c^T x}_{\text{خطی}} + \alpha \underbrace{\sqrt{\sum_{i=1}^{n} b_i y_i^2}}_{\text{محدب}}}_{\text{محدب}}$$

* به راهکار فوق convesification procedure گویند ⟸ از متغیر صفر و یک استفاده کرد.

Maral

Scanned by CamScanner

* خطی کردن قدر مطلق در دوحالت:

$$\text{Min } c^T x + z$$

$$\text{s.t:}\ \begin{cases} |a^T x + b| \le z \xrightarrow{\text{خطی شد}} -z \le a^T x + b \le z \\ z \ge 0 \end{cases}$$

در این حالت، تمایل z برای رفتن به $+\infty$ است تا بتواند تابع هدف را min کند، لذا باید با محدودیت $[\bigcirc \ge z]$، جلوی z را بگیریم.

$$\text{Min } c^T x - z$$

$$\text{s.t:}\ \begin{cases} |a^T x + b| \ge z \\ z \ge 0 \end{cases} \longrightarrow \begin{cases} a^T x + b \ge z \\ \text{or} \\ a^T x + b \le -z \end{cases} \xRightarrow{\text{خطی کردن}} \begin{cases} a^T x + b + yM \ge z \\ -(a^T x + b) + (1-y)M \ge z \\ M = \text{Large number} \\ y \in \{0,1\} \end{cases}$$

- اکنون می خواهیم با مدل هایی با ساختارهای عجیب و غریب آشنا شویم به گونه ای که هیچ نرم افزاری نمی تواند این مدل ها را در سایز بزرگ حل کند*

پس به سراغ روش های پیشرفته تری برای حل مدل ها خواهیم رفت*

← الگوریتم های لاگرانژ و بندرز را به راحتی می توان در گمز پیاده کرد ⟸ لذا الگوریتم های بسیار مناسبی خواهند بود.

- برای حل به روش بندرز، مسئله باید ساختار خاصی داشته باشد.

- Branch and cut و Branch and price، از نظر اجرای بسیار مشکل هستند به گونه ای که باید حتی در ساختار آن ها دست برد.

- لاگرانژ هم می بایست مسئله اش یک ساختار خاص داشته باشد.

★ ۴. الگوریتم حل پیشرفته:

۱- لاگرانژ: محدودیت های سخت را شناسایی می کند و به تابع هدف می برد. (accurate method است)

۲- بندرز

۳. Branch & cut

۴- Branch & price

★ آزادسازی لاگرانژ: هیچ گاه تضمین نمی کند که به جواب بهینه می رسد ولی هرکجا که متوقف شود می توانیم فاصله خودمان با جواب بهینه را پیدا کنیم ولی از طرف دیگر برای روش بندرز، اثبات می شود که به جواب بهینه همگرا خواهیم شد ★

★ روش Lshape: همان تجزیه ی بندرز است که برای مسائل Two stage stochastic به کار می رود ★

★ مسئله Minimum weighted spanning tree problem

- یک گراف وزن دار یالی: همبند داریم ← می خواهیم گرافی پیدا کنیم که کوچکترین درخت فراگیر باشد.

$G=(V, E)$ می خواهیم مسئله فوق را مدل کنیم.

$$x(e)=\begin{cases} 1 & \text{اگر } e \text{ در زیرگراف گنجانده شود} \\ 0 & \text{O.W} \end{cases}$$

★ شرط لازم فضای مسئله را بزرگتر می کند. شرط $\left(\sum x(e)=n-1\right)$ یک شرط لازم برای فراگیر بودن درخت است ★

★ شرط کافی فضای مسئله را کوچکتر می کند ★

★ پس شرط فراگیربودن را باید با شرط درخت بودن در کنار هم قرار دهیم تا شرایط لازم و کافی ایجاد گردد ★

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

* چون گفتیم تعداد یال‌ها $(n-1)$ عدد باشد، لذا تضمین می‌کنیم دور هم نداشته باشد، دیگر به درخت رسیده‌ایم *

* این مسئله را با Simplex می‌توان به جواب بهینه رسید *

S = زیرمجموعه‌ای از رئوس

V = کل رئوس

$$\text{Min} \sum c(e).x(e)$$

$$\text{s.t.} \quad \cancel{\sum x(e) \leq n-1} \xrightarrow{\text{تبدیل به}} \sum x(e) = n-1$$

$$\sum_{e \in E(G(S))} x_e \leq |S|-1 \qquad S \subseteq V, \ |S| \geq 3$$

* Maximum weight Matching problem
* Minimum weight perfect Matching problem

← چگونه مدل می‌شود؟

* Maximum weight forest ← به دنبال جنگلی می‌گردیم که بیشترین وزن یالی را داشته باشد *

* جلسه‌ی دهم:

فرض بر این است که این دو فصل را بلد هستیم

کتاب wolsey ← فصل ۲، فصل ۳، ~~فصل ۴~~، ~~فصل ۵~~، ~~فصل ۶~~، ~~فصل ۷~~

(تمرکز ما بر روی Lp-Based Branch & Bound است)

Maral ~~فصل ۸~~، فصل ۹ و فصل ۱۰ و فصل ۱۱، ~~فصل ۱۲~~ و ~~فصل ۱۳~~

★ این سه فصل حتماً خوانده شود ★

(بخش ۹-۴ خوانده نشود)

← روش سیمپلکس یک الگوریتم Greedy (حریصانه) نیست، زیرا در هر مرحله بهترین جواب را انتخاب نمی‌کند بلکه در جهت بهترین شیب تابع حرکت می‌کند *

* در شاخه و کران ابتدا مسئله را ریلکس می‌کنیم (محدودیت Integer را ریلکس می‌کنیم) بعد، سپس مسئله را حل می‌کنیم و حین حل، شروع به شاخه زدن می‌کنیم. (البته اگر جواب به دست آمده صحیح نبود، شاخه می‌زنیم)

⇐ چیزی که اهمیت دارد آن است که ریلکسیشن با جواب بهینه، فاصله‌اش کمتر شود یعنی جوابی که از مسئله ریلکس شده به دست می‌آوریم تا حد امکان کمترین فاصله را با جواب بهینه داشته باشد. ← حال برای این کار، فرمول‌های متفاوتی وجود دارد *

* بهترین حالت آن است که پس از ریلکس کردن، همان جواب بهینه به دست آید *

* ۲ روش وجود دارد:

(۱) از ابتدا انتخاب formulation صحیح انجام دهیم. (اینجا بحث آن پیش می‌آید که کدام formulation، ریلکس بهتری ایجاد می‌کند)

* انتخاب صحیح ساختار formulation *

(۲) استفاده از ساختار گسسته‌ی مسئله برای اضافه کردن محدودیت‌هایی که سبب می‌شود ریلکسیشن مسئله بهبود پیدا کند ← به این موضوع Valid inequality گویند.

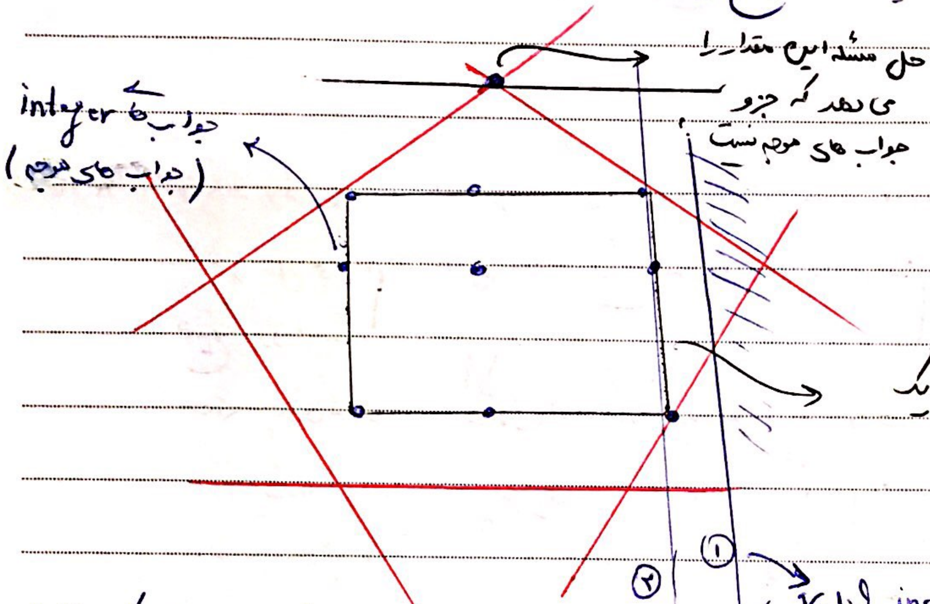
- مثلاً وقتی یک سری محدودیت داریم که از جمع آن‌ها، می‌توانیم به محدودیت دیگری برسیم.

reduntant constraint → هندسی (Geometric) → محدودیت‌هایی هستند که اگر آن‌ها را برداریم فضای شدنی مسئله تغییر نمی‌کند

reduntant constraint → جبری → محدودیت‌هایی که از جمع جبری محدودیت‌های دیگر به دست می‌آیند.

شامل جبری هم می‌شود ولی عکس آن صحیح نیست *

← برای ما Valid inequality هایی ارزش دارند که از ساختار گسسته مسئله به دست آیند وگرنه آن هایی که از جمع جبری سایر محدودیت ها به دست می آیند زیاد تفاوتی ایجاد نخواهند کرد *



← این محدودیت یک Valid inequality هست که فضای مسئله را کوچک تر می کند ولی در تابع هدف زیاد تأثیری نخواهد داشت *

← این هم یک Valid inequality است که VI شماره ۱ را مغلوب می کند و دیگر کمبود پیدا نخواهد کرد زیرا اگر کمی تغییر کند مقداری از فضای موجه گسسته را از بین می برد پس به آن Strong Valid inequality گویند.

← پس هدف، اول آن است که در صورت امکان این formulation را به دست آوریم که بدون آن که فضای موجه بهینه به دست بیاید ⟸ یعنی هدف اول آن است که دقیقاً همین مستطیل وسط به دست آید ← ولی این کار، کار آسانی نخواهد بود!!!!

لذا با Valid inequality هایی قصد داریم آرام آرام به فضای موجه بهینه دست یابیم و بالاخره به Strong Valid inequality برسیم *

* به بی نهایت حالت می توان مسئله را مدل کرد ولی همه آن ها ارزش یکسانی ندارند *

← در numerical Result است که می‌فهمیم بین تقریب formulation و
بازه‌هایی که برای رسیدن به جواب بهینه صرف می‌کنیم، چگونه یک بالانس ایجاد کنیم *

* ص ۱۴۲ کتاب ـــــ combiotenal، این formulation را توضیح داده است *

← پس به طور کلی به محدودیت‌هایی که فضای حل مسئله را کوچک‌تر می‌کنند Valid inequality
گویند و از بین آن‌ها، آن‌هایی که دیگر امکان بهبود ندارند را Strong Valid inequality گوییم.

* کتاب wolsey، فصل ۹:

X = فضای شدنی که می‌خواهیم بهینه‌سازی را روی آن انجام دهیم. (تقریباً مجموعه‌ی ۹ نقطه‌ی
صفحه‌ی قبل می‌شد)

- به دنبال آن هستیم که این Relaxation ایجاد کنیم که کمترین فاصله را با
convex hull داشته باشد.

* در فضای پیوسته اگر یک محدودیت strong نباشد، Redundant است *

* facet:

face این که درجه‌ی آن یکی از درجه‌ی فضای شدنی کمتر باشد را facet گوییم *

← پس در یک چندوجهی اگر غیر از facet defining داشته باشیم؟ زائد و به حساب می‌آید *
یعنی در حالت کسته آن محدودیت‌هایی لازم است که facet defining هستند *

* affinely independent:

تعدادی بردار را می‌گوییم affinely independent هستند، وقتی یکی را از بقیه کم کنیم، به سایر نقاط
باقی مانده Linear independent باشد *

$(x_1, x_2, \ldots, x_n) \rightarrow \left\{ \begin{array}{l} x_1 - x_n \\ x_2 - x_n \\ \vdots \\ x_{n-1} - x_n \end{array} \right\} \rightarrow$ اگر این مجموعه Linear indep باشد آن‌گاه $(x_1, \ldots, x_n)$ affinely indep هستند.

(۱۷)

Linear independent $\Longrightarrow$ affinely independent
$\not\Longleftarrow$

* پس در این قسمت باید بدانیم convex hull (X) از چه بعدی (ابعادی) است؟؟

* در فصل 9 کتاب wolsey، بخش (9-3) : 0_1 knapsack problem خوانده شد ولی Mixed integer نیازی به خواندن نیست. (9-4 خوانده نشود)

* lifting procedure یا strenghting به روش‌هایی گفته می‌شود که طی آن می‌خواهیم یک valid inequality را تجدید کنیم *

* در ابتدای کار همه‌ی valid inequality ها را وارد مسئله نمی‌کنیم!!! زیرا هم تعداد آن‌ها زیاد و هم تشخیص آن‌ها سخت است $\Longrightarrow$ لذا ابتدا مسئله را در حالت Relaxation حل می‌کنیم و می‌بینیم آیا جواب به دست آمده، یکی از valid inequality های مورد نظر ما را نقض می‌کند؟؟ $\longrightarrow$ پس به دنبال valid inequality می‌گردیم که نقض شده است و سپس آن را اضافه می‌کنیم و دوباره مسئله را حل می‌کنیم و به همین ترتیب اضافه می‌کنیم $\Longleftarrow$ به این راهکار، Sepration valid inequality گویند *

(covering inequality ها، چنین مشکلی دارند و لذا باید از راهکار Sepration برایشان استفاده کرد)

* چند وجهی صحیح:

یک چند وجهی صحیح است اگر همه‌ی نقاط گوشه‌ای آن صحیح باشد *

* محدودیت کوله پشتی، هر کجا ظاهر شود، می‌توان از covering inequality ها استفاده کرد

(مثلاً محدودیت بودجه از نوع محدودیت کوله پشتی است)

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

10) max weighted subgraph planner :

- در layout کاربرد دارد *
- زیرگرافی از یک گراف همبند وزن‌دار را به دست آوریم که بیشترین وزن را داشته باشد *
- متعلق به کلاس NP-hard

11) Minimum cost flow problem :

- یک شبکه در نظر می‌گیریم که علاوه بر ظرفیت هر یال، هزینه‌ی شارش هر واحد در هر یال در آن مشخص است. یک سری چشمه و چاهک نیز داریم و به دنبال آن هستیم که این شارش را به گونه‌ای انجام دهیم که هزینه‌های آن، کمینه شود *
- متعلق به کلاس P
- مسئله‌ی Transportation، حالت خاص این مسئله است اگر هیچ راس میانی نداشته باشیم

12) Maximum cost flow problem :

- متعلق به کلاس P
- یک Network داریم که فقط ظرفیت هر یال را داریم. یک سری چشمه و چاهک هم داریم و می‌خواهیم به گونه‌ای عمل کنیم که بیشترین جریان در گراف شارش پیدا کند *

* جلسه‌ی هفتم :

* برنامه ریزی عدد صحیح (IP) :

- شاخه و کران (Branch and bound) خوانده شود.

* برنامه ریزی ریاضی :

$$\min f(x)$$
$$\text{s.t.}\quad g_i(x) \le 0,\quad h_j(x) = 0,\quad x \in X \xrightarrow{\text{در حالت پیوسته}} X = \mathbb{R}^n \cap D$$

اشتراک دامنه‌های همه‌ی توابع در برنامه ریزی ریاضی

Maral

دامنه تابع کوادراتیک $= \mathbb{R}^n$

دامنه‌ی توابع لگاریتمی $\subseteq \mathbb{R}^n$

* $x_{n\times1}$ را به صورت زیر نمایش می‌دهیم:

$$\begin{pmatrix} x'_{n_1\times1} \\ x''_{n_2\times1} \end{pmatrix} \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}$$

$n_1 + n_2 = n$

n_1	n_2	
0	n	→ Integer programming
n	0	→ continuous programming
$0 < n_1, n_2 < n$		→ Mixed integer programming (MIP)

← در این حالت هیچ متغیر پیوسته‌ای نداریم.

← براساس ساختار و مشخصات برنامه‌ریزی ریاضی، این اسم‌ها می‌توانند خیلی خاص تر تا یک سوند.
مثلاً اگر تابع هدف خطی باشد لغت Linear قبل از programming قرار می‌گیرد.
و اکثر الگوریتم‌هایی که طراحی می‌شوند، براساس Relaxation ها می‌باشد *

← به عنوان مثال MILP را خیلی خوب می‌توان حل نمود، چراکه LP را می‌توانیم خیلی خوب حل کنیم ← پس چون ذات، MILP، از LP است، خیلی راحت می‌توانیم آن را حل نماییم *

⇐ پس تعیین نوع مدل، بسیار اهمیت پیدا می‌کند *

* ما انتظار داریم بتوانیم Mixed integer convex programming ها را حل کنیم زیرا الگوریتم‌های برای حل convex programming ها داریم *

← وقتی convex optimization می‌گوییم، پیوسته بودن (continuous) در ورنش مستتر هست.

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

← مثلاً در شاخه و کران، در هر مسئله (Node) یک convex opt حل می‌شود*

* ولی در MINLP، از شاخه و کران نمی‌شود استفاده کرد!!! زیرا ممکن است در بهینه‌ی محلی گیر کنیم*

* نکته: Global optimization به continuous opt هایی که convex نیستند می‌پردازند*

← اگر ساختار IP یا MIP ها convex باشد، چیزی که مطمئن هستیم آن است که حداقل روی کاغذ، یک الگوریتمی برای حل دقیق آن وجود دارد*

* منظور از Relaxation آن است که یک محدودیت را به طور کلی بردایم*
* ولی منظور از Relaxation Integer prog آن است که $x \in Z$ را برداریم و به جای آن $x \in R$ بگذاریم*

← البته در شاخه و کران، در هر Node یک Relaxation حل می‌شود*

← پس اگر مسئله convex باشد، می‌توانیم روی آن از شاخه و کران استفاده کنیم و مطمئن باشیم که به جواب بهینه خواهیم رسید ولی اگر convex نباشد، ممکن است جواب local به دست آوریم*.

* نکته‌ی کلیدی آن است که در شاخه و کران باید در هر Node، جواب دقیق و درست به دست آید تا خیالمان راحت باشد به جواب بهینه‌ی کلی خواهیم رسید*

← اولین قدم برای بهره‌مندی از مواهب IP، آن است که مسئله را در قالب IP مدل کنیم و سپس برای حل آن الگوریتم طراحی کنیم*

Maral

← در numerical Result است که می‌فهمیم بین تعریف formulation و
یا زمان‌هایی که برای رسیدن به جواب بهینه صرف می‌کنیم، چگونه یک بالانس ایجاد کنیم *

* ص ۱۴۲ کتاب ـــــ combiotenal و این formulation را توضیح داده است *

← پس به طورکلی به محدودیت‌هایی که فضای حل مسئله را کوچک‌تر می‌کنند valid inequality گوییم و از بین آن‌ها، آن‌هایی که دیگر امکان بهبود ندارند را strong valid inequality گوییم.

* کتاب wolsey فصل ۹:

X = فضای شدنی که می‌خواهیم بهینه‌سازی را روی آن انجام دهیم. (تعریف مجموعه‌ی ۹ نقطه‌ای صفحه‌ی قبل می‌شد)

- به دنبال آن هستیم که این Relaxation ایجاد کنیم که کمترین فاصله را با convex hull داشته باشد.

* در فضای پیوسته اگر یک محدودیت strong نباشد یعنی Redundant است *

* facet:

face ای که درجه‌ی آن یکی از درجه‌ی فضای شدنی کمتر باشد را facet گوییم *

← پس در یک چند وجهی اگر غیر از facet defining داشته باشیم؟ زائد به حساب می‌آید *
یعنی در حالت کلی تنها آن محدودیت‌هایی لازم است که facet defining هستند *

* affinely independent:

تعدادی بردار را می‌گوییم affinely independent هستند اگر وقتی یکی را از بقیه کم کنیم، سایر نقاط باقی مانده Linear independent باشند *

$$(x_1, x_2, \dots, x_n) \to \left\{\begin{array}{l} x_1 - x_n \\ x_2 - x_n \\ \vdots \\ x_{n-1} - x_n \end{array}\right\} \to$$

اگر این مجموعه Linear indep. باشد آن‌گاه $(x_1, \dots, x_n)$ affinely indep. هستند.

Linear independent $\Longrightarrow$ affinely independent
$\not\Longleftarrow$

* پس در این قسمت باید بدانیم convex hull (X) از چه دیم (ابعاد) است؟؟

* در فصل 9 کتاب wolsey: بخش (9-3) 0_1 knapsack problem: خوانده شود ولی mixed integer نیازی به خواندن نیست. (9-۴ خوانده نشود)

* lifting procedure یا strengthening به رویه‌ای گفته می‌شود که طی آن ما می‌خواهیم یک valid inequality را تقویت کنیم *

* در ابتدای کار همه‌ی valid inequality ها را وارد مسئله نمی‌کنیم!!! زیرا هم تعداد آن‌ها زیاد است و هم تشخیص آن‌ها سخت است ⟸ لذا ابتدا مسئله را در حالت Relaxation حل می‌کنیم و می‌بینیم آیا جواب به دست آمده یکی از valid inequality های مورد نظر ما را نقض می‌کند؟؟ ⟵ پس به دنبال valid inequality می‌گردیم که نقض شده است و سپس آن را اضافه می‌کنیم و دوباره مسئله را حل می‌کنیم و به همین ترتیب valid inequality اضافه می‌کنیم ⟸ به این راهکار Separation valid inequality گویند *

(covering inequality ها: چنین مشکلی دارند و لذا باید از راهکار Separation برایشان استفاده کرد)

* چندوجهی صحیح:
یک چندوجهی صحیح است اگر همه‌ی نقاط گوشه‌ای آن صحیح باشد *

* محدودیت کوله‌پشتی: هرگاه ظاهر شود، می‌توان از covering inequality ها استفاده کرد
(مثلاً محدودیت بودجه از نوع محدودیت کوله‌پشتی است)

* جلسه ی یازدهم:

* مسئله ی Traveling Salesman problem (TSP):

- می خواهیم از یک شهر به شهر دیگری برویم به گونه ای که هیچ شهری تکرار نشود و هزینه ها min شود*
- به دنبال یک دور همیلتونی هستیم که کمترین وزن یالی را داشته باشد ← بیان مسئله ی TSP به زبان گراف تئوری

* مسئله ی TSP، یک مسئله ی پایه ای برای مسائل VRP می باشد*
* پایه ای ترین حالت مسئله ی TSP، یک TSP متقارن است*

در TSP متقارن ⇐ $c_{ij} = c_{ji}$

هر مسیر هم رفت دارد و هم برگشت

* در TSP، فرض می شود که گراف کامل است، به این دلیل که حتماً دور همیلتونی داشته باشد و اگر حتی در واقعیت هم کامل نباشد، یال هایی که وجود ندارند را با وزن بینهایت در گراف در نظر می گیرند*

$G=(V,E)$ گراف ساده

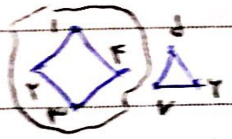
← می توانیم گراف مسئله ی TSP را ساده در نظر بگیریم، زیرا طوقه در این مسئله که اصلاً به درد ما نمی خورد، اگر هم بین دو راس، ۲ یال داشته باشیم، می توانیم یال با وزن کمتر را در گراف نگه داریم و گراف در نهایت گراف ساده خواهد شد*

* مدل ریاضی مسئله TSP:

* از order، 2^n است *

$\delta(v)$ = مجموعه یال‌های متصل به v (یال‌هایی که از v خارج شده‌اند)

$\delta(X)$ = مجموعه یال‌هایی که از مجموعه X خارج می‌شوند:



$X=\{1,2,3,4\} \Rightarrow \delta(X)=0$

$$\min Z=\sum_{e\in E} c_e x_e$$

s.t.

$$\sum_{e\in\delta(v)} x_e = 2 \quad ; \forall v\in V$$

خود V نمی‌تواند باشد

(۲) $$\sum_{e\in\delta(X)} x_e \geq 2 \quad ; \forall X\subset V \; ; \; 3\leq |X| \leq \left\lfloor \frac{|V|}{2} \right\rfloor$$

$$x_e\in\{0,1\} \quad ; e\in E$$

* می‌توانیم به جای محدودیت شماره‌ی (۲) در مدل فوق، از محدودیت زیر استفاده کنیم:

$$\sum_{e\in E(G(X))} x_e \leq |X|-1 \quad ; \forall X\subset V \; ; \; 3\leq |X| \leq \left\lfloor \frac{|V|}{2} \right\rfloor$$