ارائه دهندگان :

سوده کاکوئی نژاد

اعظم قدرت نما

Linear Programming

Manufacturing with Modls

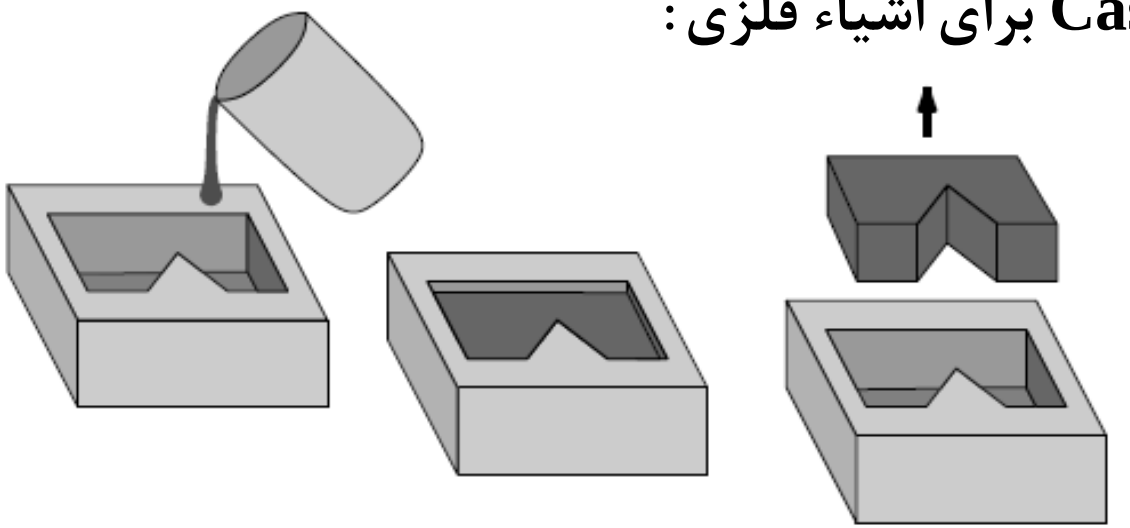
برنامه ریزی خطی _ قالب گیری

بحث مورد مطالعه در این فصل :

بررسی اینکه برای یک شی داده شده یک قالب مناسب وجود دارد که بتوان شی را براحتی از قالب خارج کرد؟

در واقع در این فصل یک نمای هندسی از ساخت اشیاء توسط قالب گیری مطرح می شود.

مراحل Casting برای اشیاء فلزی :



در مورد شکل قبل مراحل به این ترتیب است که ابتدا فلز مایع در قالب ریخته می شود و پس از اینکه مجددا به حالت جامد تبدیل شد باید آن شی را از قالب خارج کنیم.اما باید توجه کرد که مرحله آخر همیشه به این آسانی که به نظر می آید نیست زیرا ممکن است که شی به قالب بچسبد و به عبارتی طوری در قالب قرار گرفته باشد که امکان خروج وجود ندارد در نتیجه مجبور به شکستن قالب خواهیم شد.

اگر امکان خروج شی وجود نداشته باشد،گاهی اوقات می توان از یک قالب دیگر استفاده کرد اما برای همه اشیا این کار امکانپذیر نیست.مثلا برای ساخت کره قالب مناسبی وجود ندارد.

محدودیت های مسئله :

۱. اشیائی که باید ساخته شوند چند وجهی هستند .

۲. قالب ها باید یک قطعه باشند ، به عبارتی قالب ها نباید شامل دو قطعه یا بیشتر باشند .(استفاده از قالب هایی که شامل دو قطعه باشند ساخت اشیایی مثل کره را امکانپذیر می کند که با یک قالب یک قطعه ای امکانپذیر نیست.)

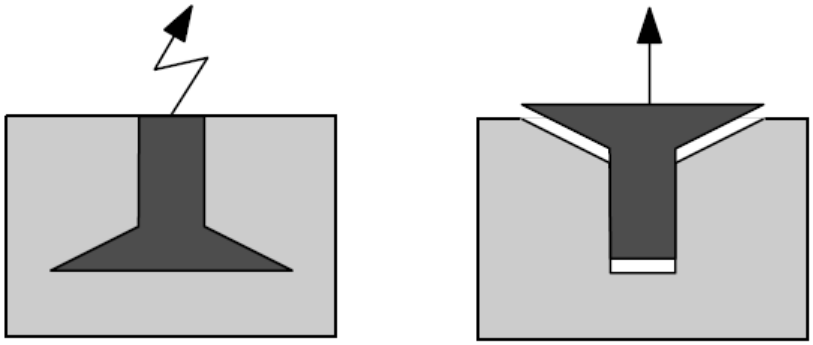
۳. در انتها شی باید با یک انتقال از قالب خارج شود .به این معنا است که نمی توان آن را با پیچاندن از قالب خارج کرد.

4.1 The Geometry of Casting

برای تصمیم گیری در این مورد که یک شی می تواند با Casting ساخته شود باید یک قالب مناسب برایش پیدا کنیم .

شکل حفره داخل قالب با توجه به شکل شی تعیین می شود و جهت های متفاوت از شی قالب های متفاوتی را ایجاد می کند.

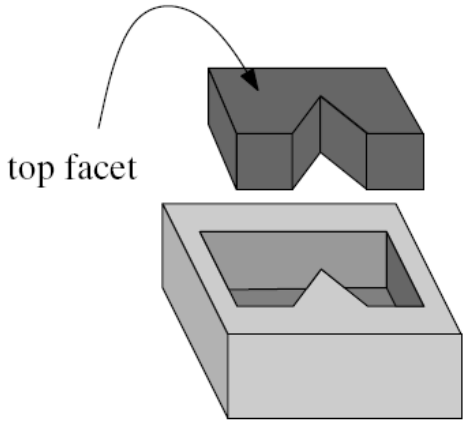
انتخاب چگونگی قرارگیری شی در قالب می تواند مشکل باشد . زیرا در بعضی حالت ها خروج شی از قالب امکان پذیر نیست .



پس با توجه به اینکه کدام سطح شی را به عنوان سطح بالا در نظر بگیریم قالب های متفاوتی خواهیم داشت.

Top facet :

سطح بالایی شی است که باید یک سطح افقی (صاف) باشد و با سطوح قالب تماس نداشته باشد که در هنگام قرار گیری شی در قالب باید موازی با صفحه ی xy باشد .



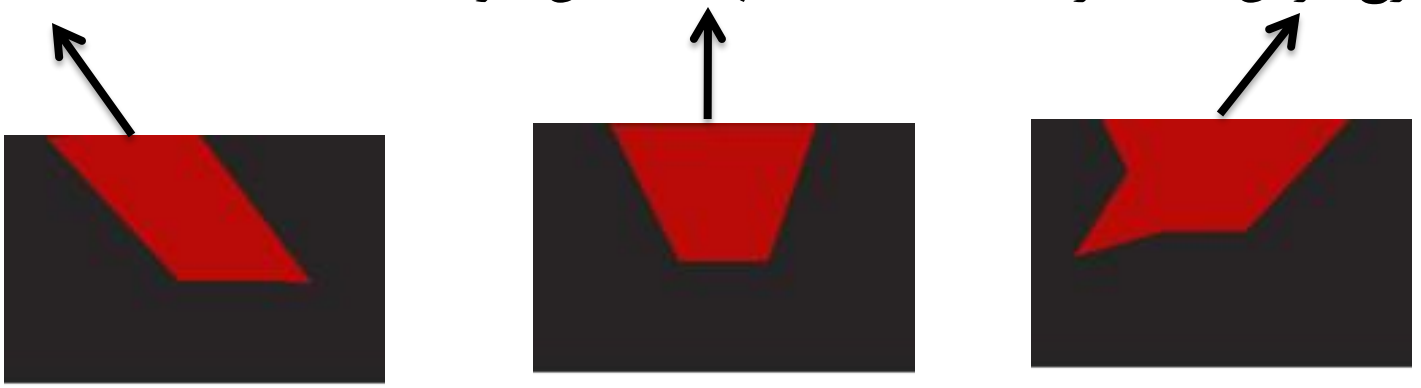
یک شی را castable گوییم اگر بتواند حداقل از یکی از جهت ها از قالب خارج شود .

برای تصمیم گیری در مورد قابلیت Castability یک شی باید همه جهت ها را بررسی کنیم.

Ordinary Facet (عادی):

سطحی از $\mathscr{P}$ که top facet نباشد که متناظر با هر وجه عادی f یک سطح در قالب وجود دارد که با $\hat{f}$ نشان داده می شود .

برای خارج کردن قطعه از قالب باید یک جهت تعیین شود.

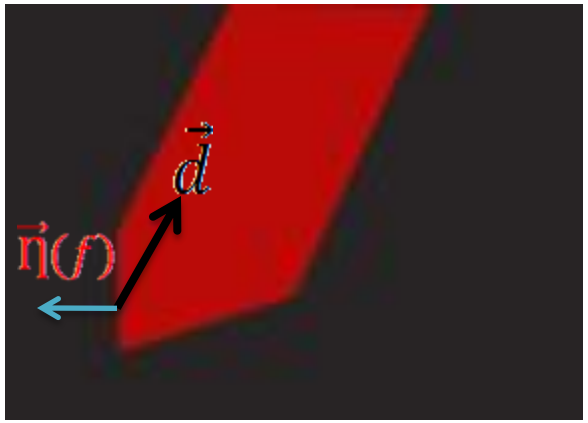


با توجه به این که وجهی از شیء که با قالب تماس ندارد (Top facet) در بالای قالب است، جهت انتخابی باید رو به بالا (یعنی در راستای محور zصعودی) باشد اما این محدودیت یک شرط لازم است و به تنهایی برای داشتن یک جهت خروج معتبر کافی نیست.

فرض کنید که $\mathscr{P}$ یک چند وجهی سه بعدی باشد .

فرض کنید f یک وجه عادی از شی باشد ، برای خارج کردن شی از قالب باید f از روی سطح $\hat{f}$ از قالب بلند شود یا روی آن بلغزد .

❖برای تعیین زاویه بین دو بردار باید ابتدا آن دو بردار را طوری در نظر بگیریم که مبدا آنها مبدا دستگاه مختصات باشد ، سپس این دو بردار در صفحه ی متناظرشان دو زاویه دارند که از بین آن دو باید زاویه کوچکتر را در نظر بگیریم.



❖ $\vec{\eta}(f)$: بردار عمود بر سطح f و به سمت خارج .

❖ یک شرط لازم روی $\vec{d}$ این است که زاویه بین $\vec{d}$ و $\vec{\eta}(f)$ حداقل ۹۰ باشد .

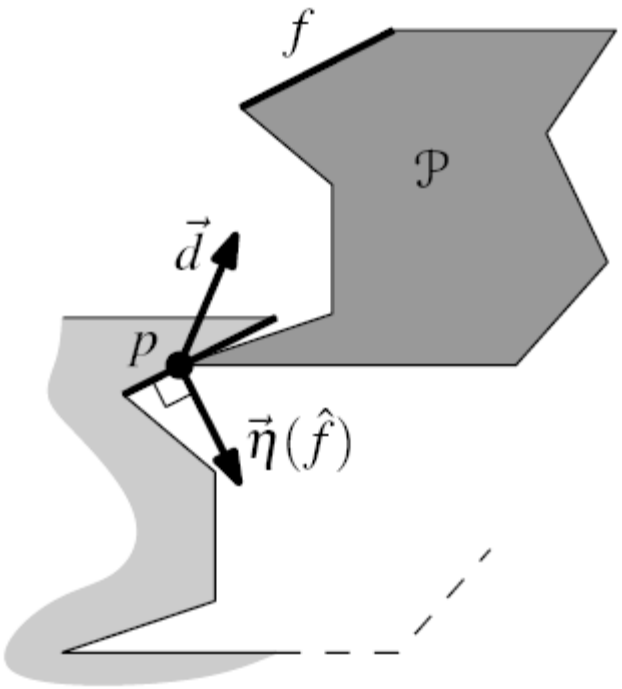
لم۴.۱ : چند وجهی $\mathscr{P}$ به وسیله یک انتقال در جهت $\vec{d}$ میتواند از قالب خارج شود اگر و تنها اگر حداقل یک زاویه ۹۰ با $\vec{\eta}(f)$ به ازای همه سطوح $\mathscr{P}$ بسازد .

اثبات :

❖اگر $\vec{d}$ زاویه کمتر از ۹۰ درجه با $\vec{\eta}(f)$ بسازد آنگاه زمانی که انتقال در جهت $\vec{d}$ باشد هر نقطه q در سطح f با قالب برخورد می کند .

فرض کنید وقتی که $\mathscr{P}$ در جهت $\vec{d}$ حرکت میکند با قالب برخورد کند ،باید نشان دهیم که در این حالت $\vec{d}$ زاویه کمتر از ۹۰ درجه با $\vec{\eta}(f)$ ساخته است .فرض شود p نقطه ای از شی $\mathscr{P}$ باشد که با سطح $\hat{f}$ از قالب برخورد کرده است ، به این معناست که $\mathscr{P}$ تقریباً به

داخل قالب حرکت کرده ، پس $\vec{\eta}(\hat{f})$ یک زاویه بیش از ۹۰ درجه با $\vec{d}$ می سازد آنگاه $\vec{d}$ زاویه ای کمتر از ۹۰ با $\vec{\eta}(f)$ خواهد ساخت .



نتیجه :

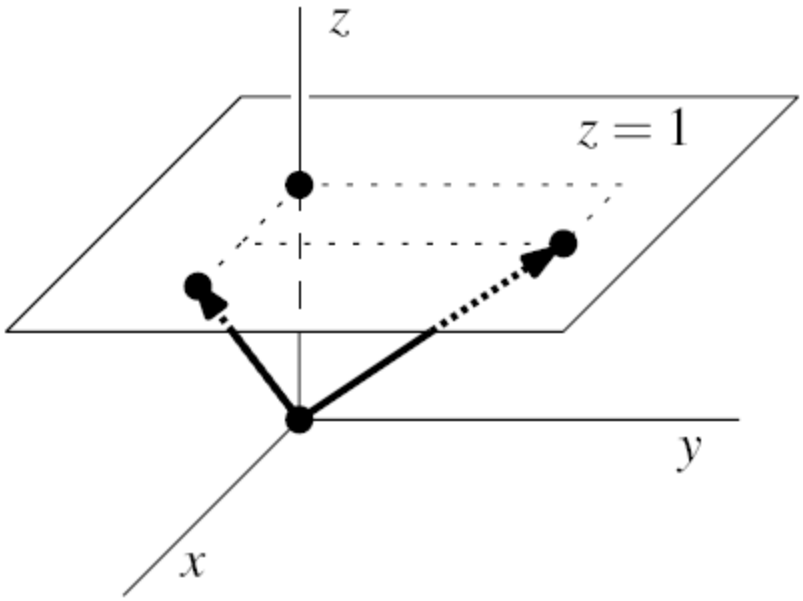
اگر $\mathscr{P}$ بتواند توسط دنباله ای از انتقال های کوچک از قالب خارج شود آنگاه میتوان آنرا توسط یک انتقال نیز خارج کرد .

✓ **روند تبدیل مسئله فوق به یک مسئله هندسی :**

❖هر جهت در فضای ۳ بعدی را می توان توسط یک بردار که از مبدا شروع می شود، نمایش داد .(رو به بالا یعنی دارای مولفه z مثبت)

❖همچنین میتوان همه جهت ها را به عنوان نقاطی در صفحه z=1 در نظر گرفت .

❖نقطه ی(x , y , 1) یک بردار جهت (x , y , 1) را نشان میدهد .



❖بدین ترتیب یک تناظر یک به یک بین نقاط در صفحه z=1 و بردارهای جهت (x , y, 1) برقرار می شود .

فرض کنید $\vec{\eta} = (\vec{\eta}_x , \vec{\eta}_y , \vec{\eta}_z)$ بردار خروجی از یک وجه عادی باشد . بردار جهت $\vec{d} = (d_x , d_y , 1)$ یک زاویه حداقل ۹۰ درجه با بردار $\vec{\eta}$ می سازد اگر وتنها اگر ضرب داخلی آنها مثبت نباشد.

بنابراین یک وجه عادی یک محدودیت به شکل زیر ایجاد می کند :

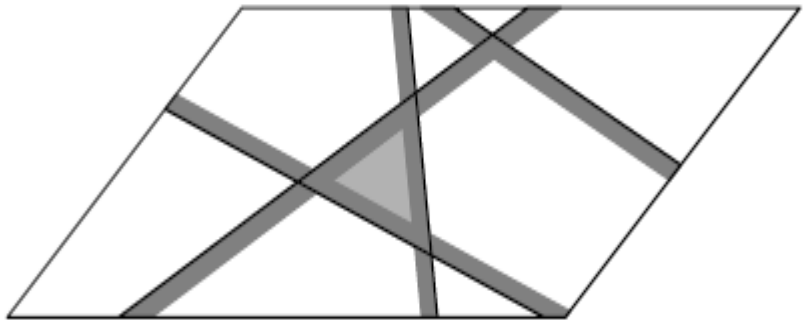
$$\vec{\eta}_x d_x + \vec{\eta}_y d_y + \vec{\eta}_z \leq 0$$

نامعادله فوق یک نیم صفحه روی صفحه $z=1$ را توصیف می کند .(نا معادله فوق برای صفحه های افقی که $\vec{\eta}_x = \vec{\eta}_y = 0$ درست نمی باشد چون در این حالت داریم : $\vec{\eta}_z \leq 0$ اما بیان شد که باید مولفه z مثبت باشد.)

بنابراین هر وجه غیر افقی از $\mathscr{P}$ یک نیم صفحه روی صفحه $z=1$ تعریف میکند ، و هر نقطه در اشتراک این نیم صفحه ها مطابق با یک بردار جهت است که $\mathscr{P}$ می تواند در آن جهت از قالب خارج شود .

اشتراک نیم صفحه ها ممکن است تهی باشد که در این حالت نمی توان $\mathscr{P}$ را از قالب خارج کرد .

❖با توجه به توضیحات قبل مسئله اولیه به یک مسئله هندسی در صفحه تبدیل می شود .



مسئله هندسی :

یک مجموعه از نیم صفحه ها داده شده است و یک نقطه در اشترک آنها را باید پیدا کنیم یا تشخیص دهیم که اشتراکشان تهی است .

❖اگر چندوجهی داده شده n سطح داشته باشد آنگاه مسئله هندسی بیش از n-1 نیم صفحه ندارد زیرا یک سطح به عنوان Top Facet در نظر گرفته می شود و به ازای هر وجه عادی یک نیم صفحه خواهیم داشت.

قضیه ۴.۲ :

فرض کنید $\mathscr{P}$ یک شی با n وجه باشد ،در زمان انتظار $O(n^2)$ و استفاده از $O(n)$ ذخیره سازی می توان تصمیم گرفت که $\mathscr{P}$ ،Castable است یا خیر . علاوه بر این اگر $\mathscr{P}$ ، Castable باشد یک قالب و یک بردار جهت معتبر برای خروج $\mathscr{P}$ در زمان مشابه محاسبه می شود .

توضیح در رابطه با قضیه قبل : برای اینکه تشخیص داده شود که یک شی castable یا خیر ، حداکثر n انتخاب برای Top Facet وجود دارد که با انتخاب هر یک باید اشتراک بین n-1 نیم صفحه به دست آید که با استفاده از الگوریتمی که در بخش ۴.۴ ارائه خواهد شد در زمان $O(n)$ قابل محاسبه است و بنابراین زمان کل $O(n^2)$ است.

برای هر یک از حالات انتخاب برای Top Facet حداکثر باید n-1 نیم صفحه ذخیره شود ، بنابر این به $O(n)$ ذخیرا سازی نیاز دارد.

4.2 Half – Plane Intersection

فرض کنید $H = \{h_1, h_2, \ldots, h_n\}$ مجموعه ای از محدودیت ها خطی با دو متغیر به شکل زیر باشد :

$$a_i x + b_i y \leq c_i$$

که a_iو b_i و c_i ثابت هایی هستند بطوریکه حداقل یکی از a_iیا b_i مخالف صفر باشند .

❖از لحاظ هندسی یک محدودیت به عنوان یک نیم صفحه درR^2 است که توسط خط $a_i x + b_i y = c_i$ کران دار می شود .

❖مسئله پیدا کردن مجموعه همه نقاط $(x, y) \epsilon R^2$ است که در همه n محدودیت به طور همزمان صدق کند یا به عبارتی نقاطی که میان نیم صفحه های مجموعه H مشترک باشد .

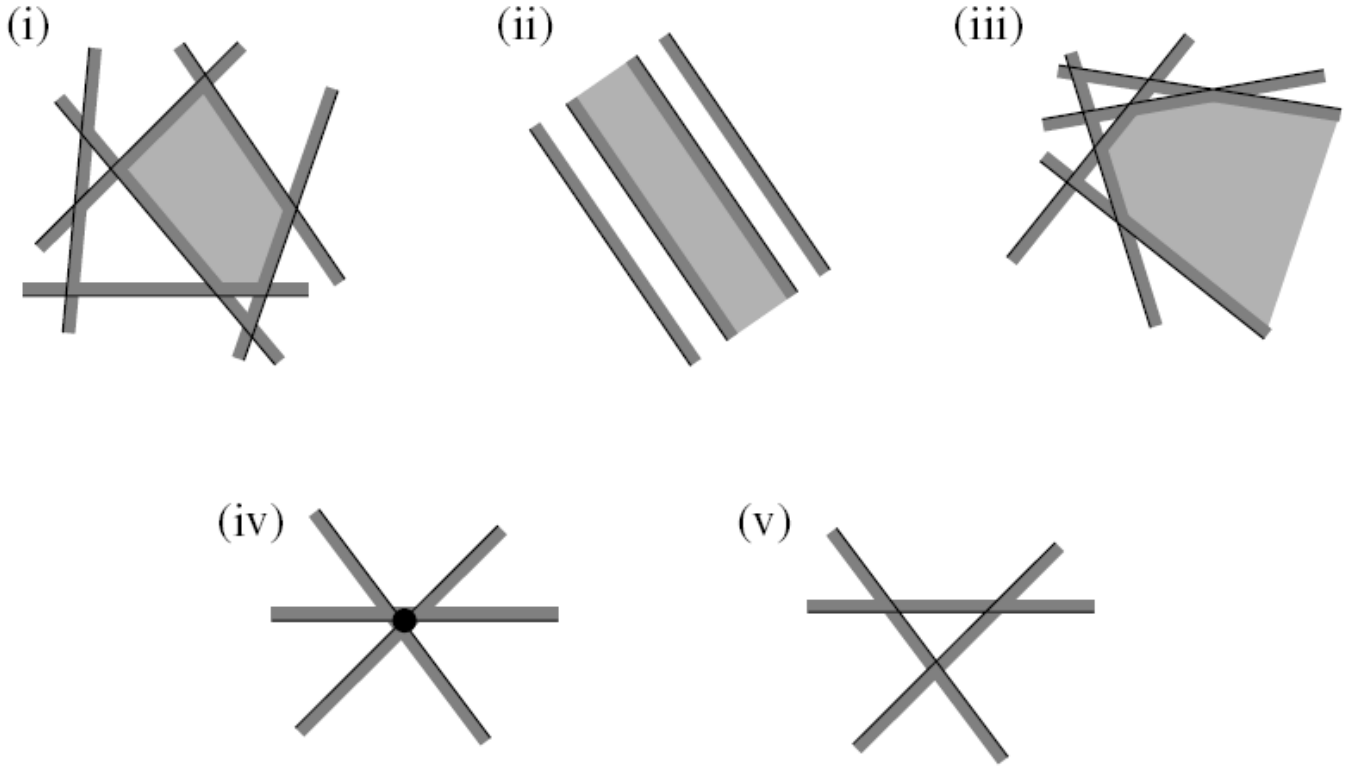
ویژگی ناحیه مشترک :

❖یک نیم صفحه محدب است و اشتراک مجموعه های محدب نیز یک مجموعه محدب است ، بنابراین اشتراک مجموعه ای از نیم صفحه ها یک ناحیه محدب در صفحه است .

❖هر نقطه روی مرز ناحیه مشترک روی خط مرزی (کران) برخی نیم صفحه ها قرار دارد .

❖خطوط مرزی ناحیه مشترک شامل خطوط مرزی نیم صفحه ها خواهند بود .

بخش مشترک از n نیم صفحه ، یک چند ضلعی محدب است که حداکثر توسط n یال کراندار می شود .



❖ناحیه مشترک ممکن است یک نقطه باشد یا یک پاره خط یا تهی باشد .

یک راه آسان برای محاسبه ناحیه مشترک الگوریتم تقسیم وتسخیر است .

Algorithm INTERSECTHALFPLANES(H)
Input. A set H of n half-planes in the plane.
Output. The convex polygonal region $C := \bigcap_{h \in H} h$.
1. **if** $\text{card}(H) = 1$
2. **then** $C \leftarrow$ the unique half-plane $h \in H$
3. **else** Split H into sets H_1 and H_2 of size $\lceil n/2 \rceil$ and $\lfloor n/2 \rfloor$.
4. $C_1 \leftarrow$ INTERSECTHALFPLANES(H_1)
5. $C_2 \leftarrow$ INTERSECTHALFPLANES(H_2)
6. $C \leftarrow$ INTERSECTCONVEXREGIONS(C_1, C_2)

الگوریتم فوق برای محاسبه ناحیه اشتراک ابتدا مجموعه H را به دو مجموعه تقسیم میکند و به صورت بازگشتی برای محاسبه اشتراک دو مجموعه کوچک عمل می کند و سپس روال بیان شده در خط ششم برای محاسبه ناحیه اشتراک مربوط به دو ناحیه چندضلعی که هر کدام از اشتراک حداکثر n/2+1 نیم صحه به دست آمده اند،فراخوانی می شود.

برای پیاده سازی روال مربوط به خط ششم می توان از روش های مختلفی استفاده کرد که یکی از این روش ها استفاده از الگوریتم ارائه شده در فصل دوم است که برای محاسبه overlay دو نقشه بیان شد و برای محاسبه اشتراک چندضلعی ها نیز استفاده شد.

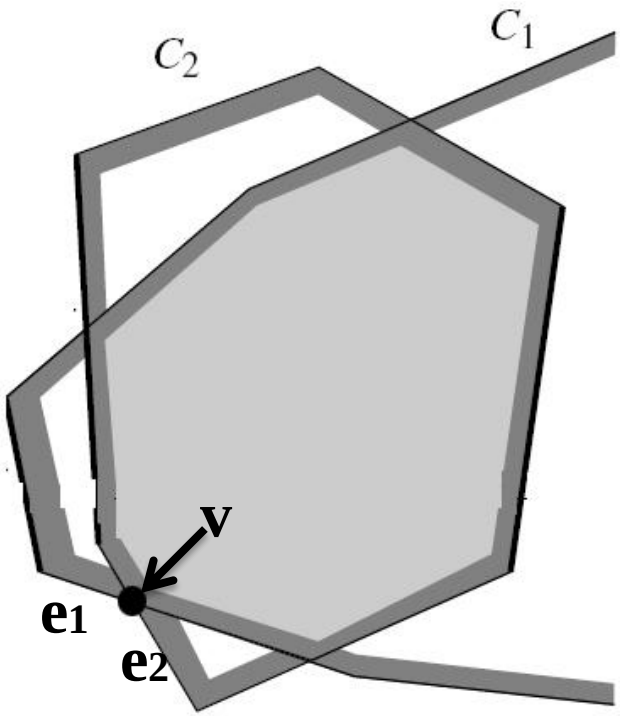
یادآوری _نتیجه ۲.۷:

فرض کنید P_1 یک چندضلعی با n_1 راس و P_2 یک چند ضلعی با n_2 راس باشد و $n=n_1+n_2$ آنگاه $P_1 \cap P_2$ و $P_1 \cup P_2$ و $P_1 \setminus P_2$ در زمان $O(nlogn + klogn)$ محاسبه می شود که k پیچیدگی خروجی است .

❖ در مورد بکارگیری نتیجه فوق برای مسئله مان باید یه این نکته توجه کنیم که ناحیه ها می توانند نامحدود باشند یا یک پاره خط و یا یک نقطه باشد . به عبارتی این ناحیه ها الزاماً چند ضلعی نیستد .

❖فرض کنید که دو ناحیه C_1 و C_2 توسط بازگشت محاسبه شده اند ، و چون هر دوی آنها توسط حداکثر $n/2 +1$ نیم صفحه تعریف شده اند ، حداکثر $n/2+1$ یال دارند .

❖الگوریتم ارئه شده در فصل دوم Overlay، C_1 وC_2 را در زمان$O((n+k)\log n)$ محاسبه می کند که k تعداد نقاط مشترک بین یال های C_1 وC_2 است .



Vباید یک راس از اشتراک C_1 وC_2 و از طرفی $C_1 \cap C_2$ ناحیه مشترک n نیم صفحه است

بنابراین حداکثر n یال و n راس دارد و به این معناست که : $k \leq n$

O (nlogn+klogn) ⟶ O (nlogn)

زمان اجرای الگوریتم :

$$T(n) = \begin{cases} O(1), & \text{if } n = 1 \\ O(n \log n) + 2T(n/2), & \text{if } n > 1 \end{cases}$$

پس : $T(n) = O(n \log^2 n)$

❖ **با توجه به این که چندضلعی ها در این مساله همیشه محدب هستند، آیا الگوریتم کاراتری برای حل مساله وجود دارد؟**

در واقع همانطور که قبلا بیان شد ناحیه اشتراک بین نیم صفحه ها یک ناحیه محدب است.

در اینجا فرض می شود که ناحیه هایی که می خواهیم اشتراک بین آنها را محاسبه کنیم دو بعدی هستند و از حالاتی که ناحیه ها نقطه یا پاره خط باشد صرف نظر می شود.

پس قبل از اینکه الگوریتم جدید مطرح شود باید چگونگی نمایش یک ناحیه محدب را بیان کنیم.

چگونگی نمایش یک چند ضلعی محدب C :

❖مرزهای چپ و راست C را به طور جداگانه در دو لیست مرتب از نیم صفحه ها قرار می دهیم .

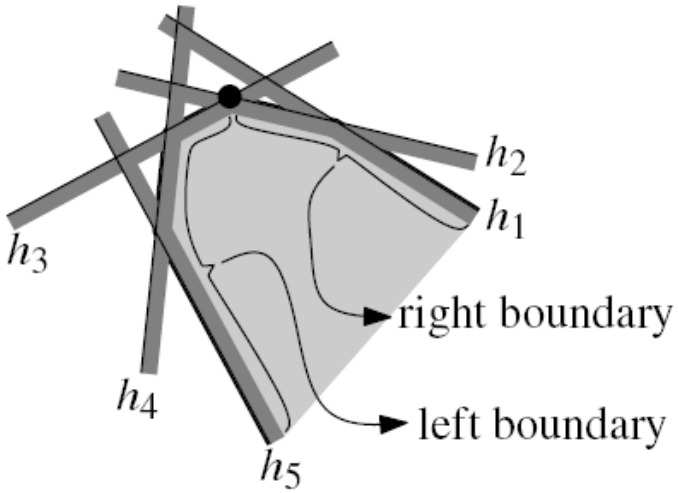
❖از بالا به پایین خطوط مرزی C در لیست ها به ترتیب قرار می گیرد .

$\mathcal{L}_{left}(C)$ = لیست مرتب از خطوط مرزی چپ

$\mathcal{L}_{right}(C)$ = لیست مرتب از خطوط مرزی راست

❖رئوس به طور صریح ذخیره نمی شوند ، می توان آنها را توسط تقاطع های متوالی خطوط مرزی محاسبه کرد .

❖فرض می کنیم چندضلعی ضلع افقی ندارد.



$\mathcal{L}_{\text{left}}(C) = h_3, h_4, h_5$

$\mathcal{L}_{\text{right}}(C) = h_2, h_1$

الگوریتم جدید :

❖مشابه الگوریتم Plane Sweep ارائه شده در فصل ۲ است .

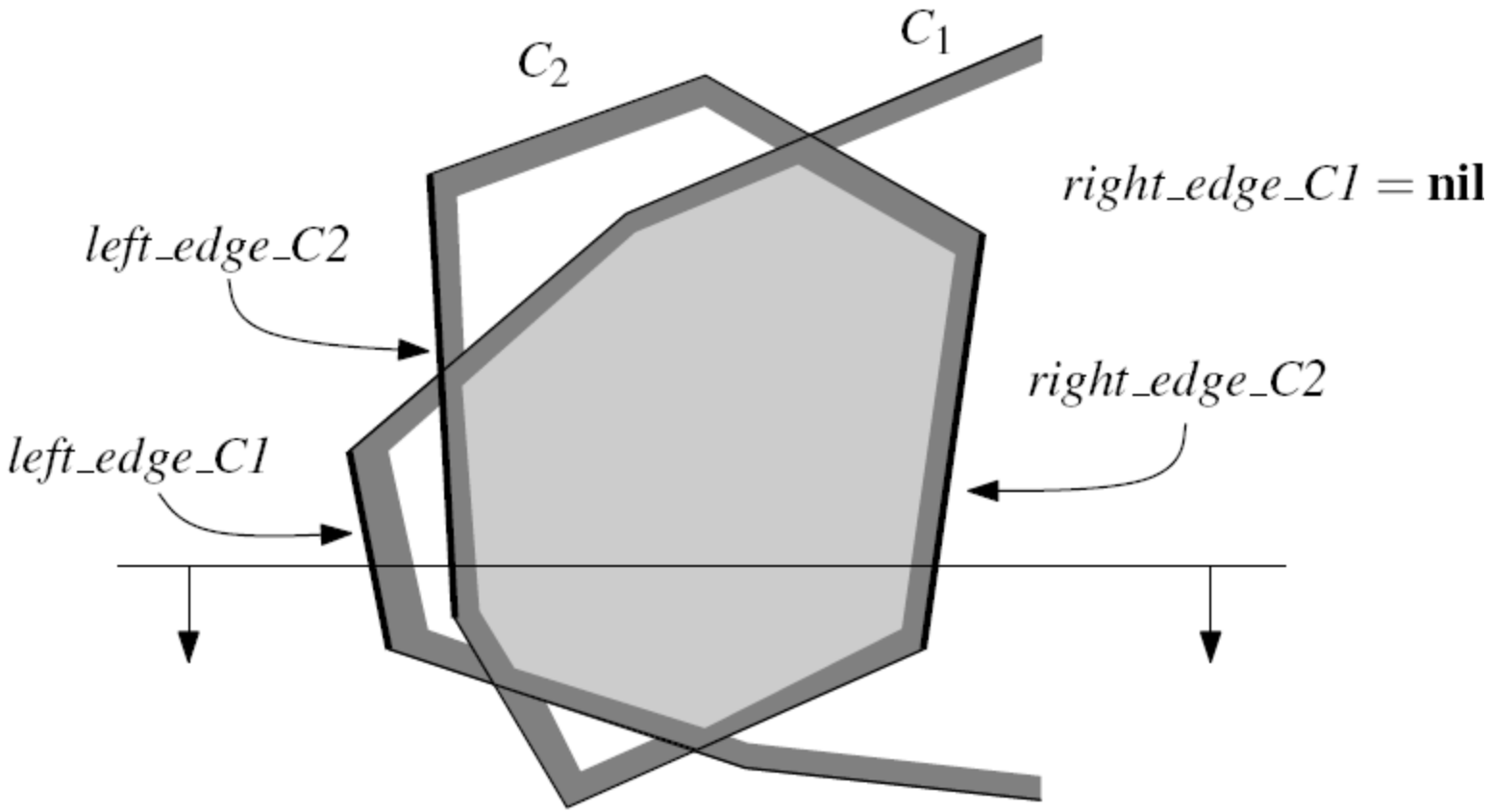
❖یک خط افقی را از بالا به پاین حرکت می دهیم تا یال های C_1 وC_2 توسط آن قطع شوند .

❖چون C1 و C2 محدب هستند حداکثر ۴ یال توسط Sweep Line قطع می شوند ، بنابراین فقط کافی است که اشاره گرهایی به این ۴ یال داشته باشیم . که عبارتند از :

Right_edge_C1, Left_edge_C1

Right_edge_C2, Left_edge_C2

❖اگر Sweep Line مرز چپ یا راست یک ناحیه را قطع نکند آنگاه اشاره گر منطبق با آن nil می باشد .


C_2
C_1
right_edge_C1 = **nil**
left_edge_C2
right_edge_C2
left_edge_C1

❖فرض کنید y_1 مولفه y مربوط به بالاترین راس در C_1 باشد و اگر C_1 از بالا کراندار نبود $y_1=\infty$ و به طور مشابه y_2 برای C_2 تعریف می شود .

$$y_{start} = \min(y_1 , y_2)$$

❖برای یافتن ناحیه مشترک، الگوریتم باید روی بخشی از صفحه که $y \leq y_{start}$ اجرا شود.

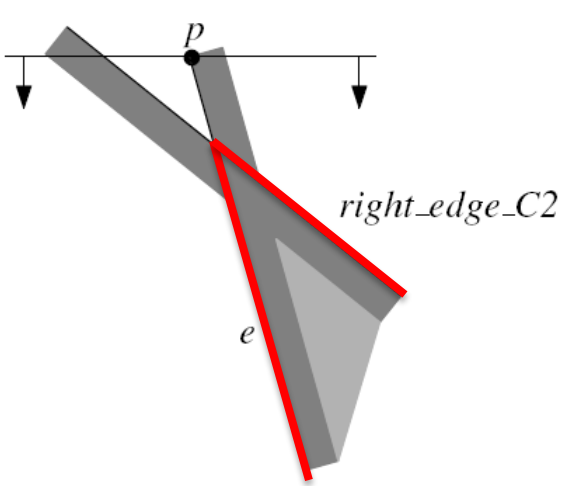
یادآوری :

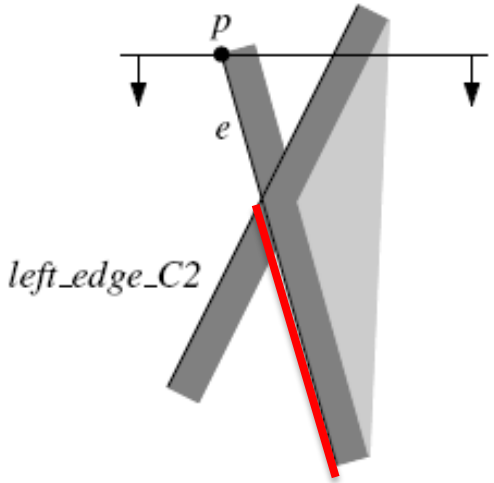
در الگوریتم ارئه شده در فصل ۲ به یک صف برای ذخیره کردن event point ها نیاز بود .

❖در اینجا event ها نقاط شروع یا پایان یال های C_1 یا C_2 هستند که با Sweep Lineبرخورد داشته اند .بنابراین event بعدی بالاترین نقطه از بین نقاط پایین Sweep Line از یال هایی است که توسط Sweep Line قطع شده اند . **بنابراین به صف نیاز نیست .**

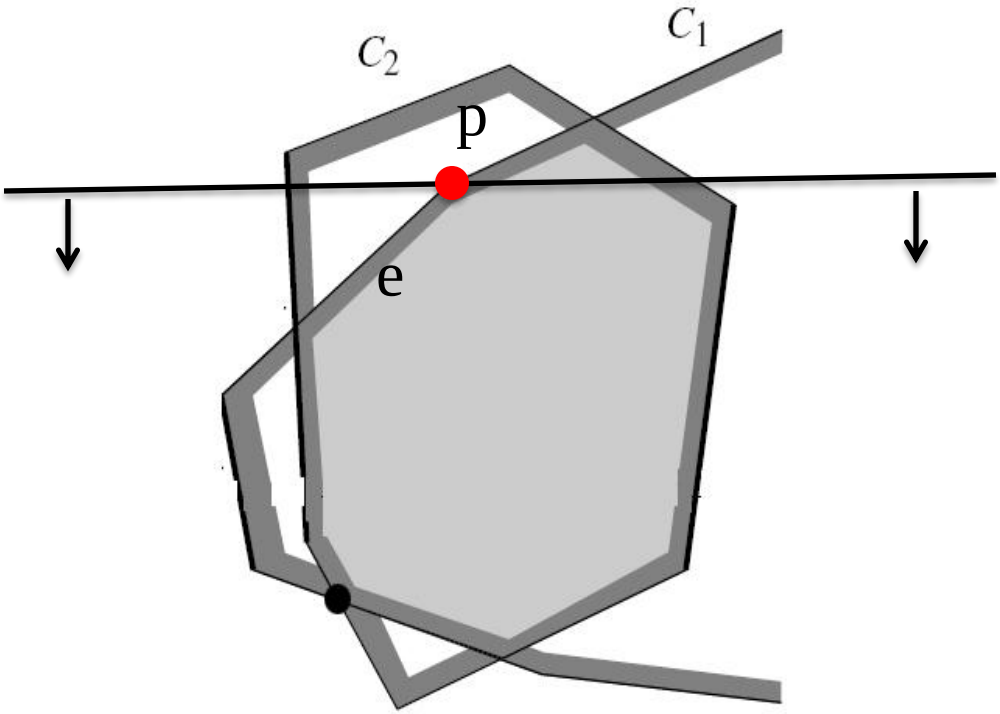
برای یال e ابتدا باید بررسی شود که متعلق به C_1 است یا C_2 و اینکه یک خط مرزی چپ است یا راست و سپس کارهای مربوطه انجام می شود .

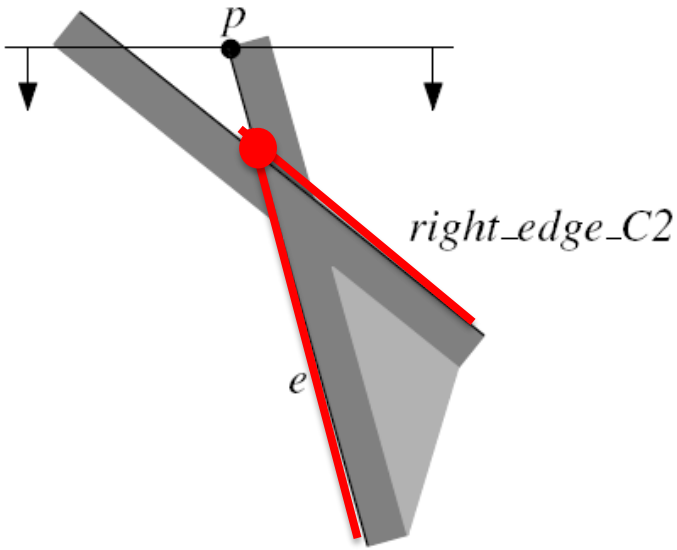
- فرض کنید e یال مرز چپ C_1 باشد و p نقطه بالایی آن باشد بنابراین C یکی از ضلع های زیر را خواهد داشت :
  - ✓ یالی که p نقطه بالایی آن است .
  - ✓ یال هایی که نقطه ی بالای آنها $e \cap right\_edge\_C_2$ است .
  - ✓ یالی که نقطه ی بالای آن $e \cap left\_edge\_C_2$ است .





اگر pدر بین $left\_edge\_C_2$ و$right_edge_C_2$ قرار داشته باشد e به عنوان یالی از C محسوب می شود و سپس نیم صفحه ای که e خط کران آن می باشد به$\mathscr{L}_{left}(C)$ اضافه می شود .





اگر یال e با right_edge_C2 برخورد داشته باشد آنگاه نقطه برخورد یک راس ازCمی باشد و هر دو یال به عنوان یال هایی از C خواهند بود که دو حالت رخ می دهد:

۱)یال ها از نقطه برخورد شروع شوند .

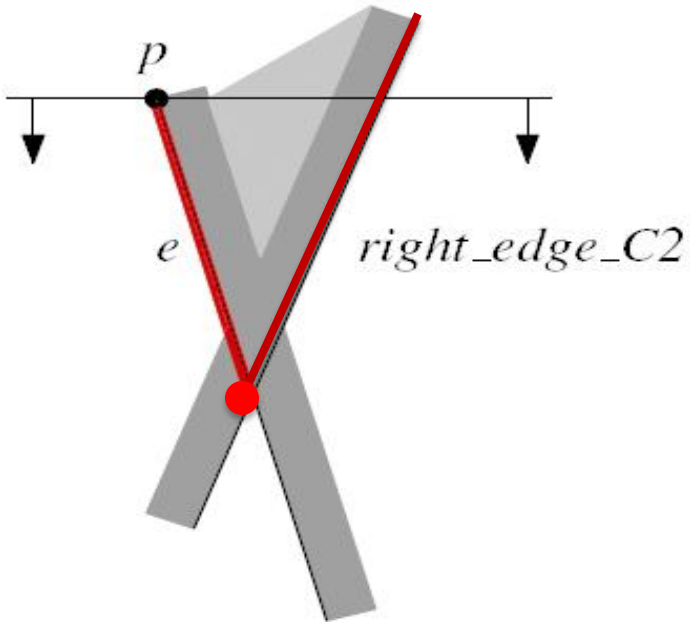
این اتفاق زمانی می افتد که P در سمت راست $right\_edge\_C_2$ قرار داشته باشد .

آنگاه باید نیم صفحه ی تعریف شده توسط e را به $\mathscr{L}left(C)$ و نیم صفحه تعریف شده توسط $right\_edge\_C_2$ را به $\mathscr{L}right(C)$ اضافه کنیم .

۲) هر دو یال به عنوان یالی از C خواهد بود که در نقطه برخورد پایان می یابند.

زمانی این اتفاق می افتد که p در سمت چپ $right\_edge\_C_2$ قرار داشته باشد .

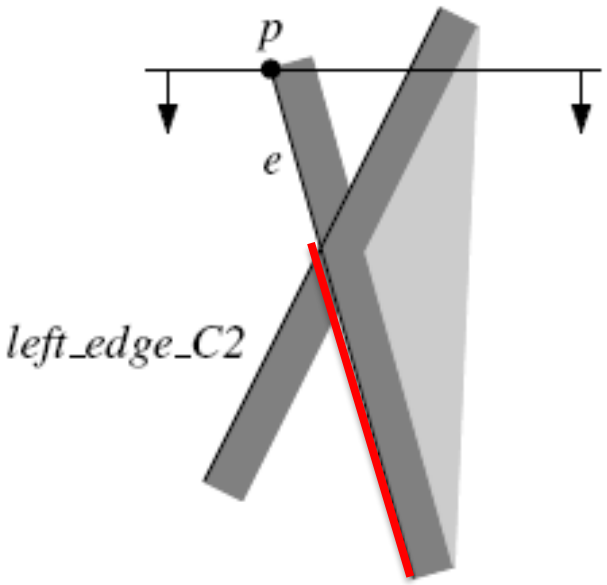
در این حالت کاری انجام نمی شود زیرا این یال ها قبلاً پیدا شده اند .



اگر e با $left\_edge\_C_2$ برخورد داشته باشد آنگاه نقطه برخورد راسی از C است که یال C با آن شروع می شود که این یال بخشی از e و یا بخشی از $left\_edge\_C_2$.

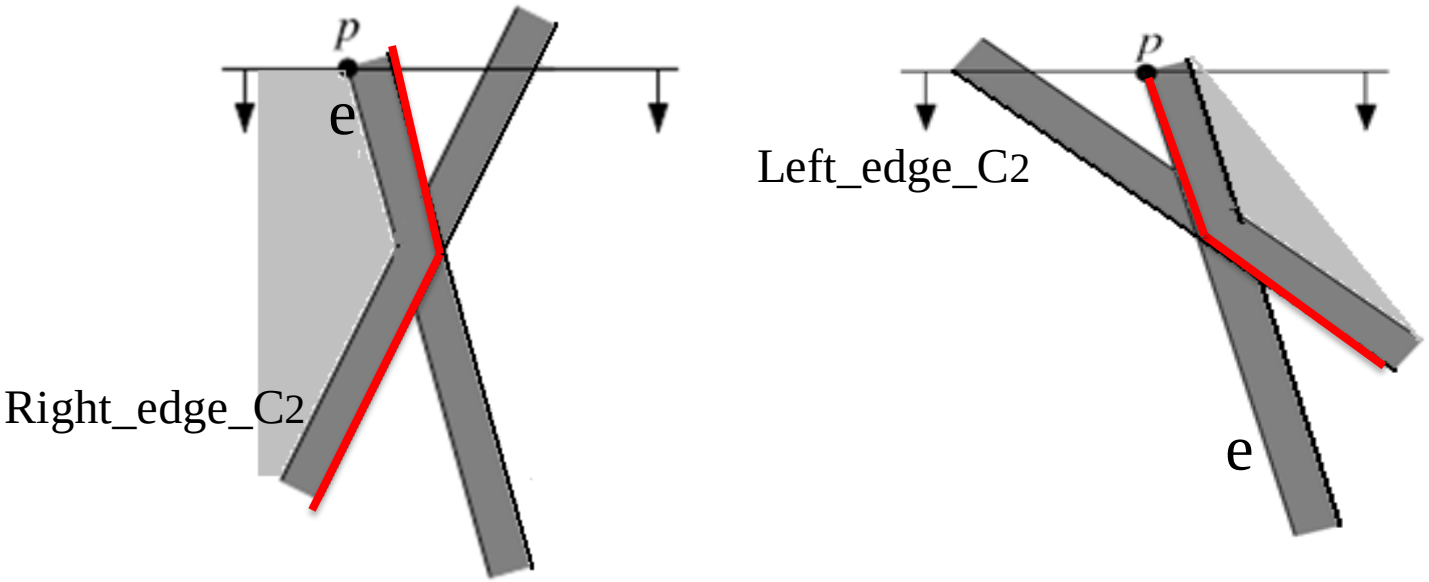
✓ می توان در زمان ثابت یکی از دو حالت فوق را انتخاب کرد :

اگر p در سمت چپ left_edge_C2 قرار داشته باشد آنگاه یال مورد نظر بخشی از e خواهد بود در غیر این صورت بخشی از left_edge_C2 است .



سپس باید نیم صفحه انتخاب شده را به $\mathscr{L}_{left}(C)$ اضافه کرد .

آیا زمانی که باید دو نیم صفحه به $\mathscr{L}_{left}$ یا $\mathscr{L}_{right}$ اضافه شود ترتیب نیم صفحه ها درست است ؟



اگر حالات مشابه اول و دوم یا اول و سوم با یک دیگر رخ دهد،در هر کدام باید دو نیم صفحه به $\mathscr{L}_{left}$ یا $\mathscr{L}_{right}$ اضافه شود.با توجه به اینکه باید نیم صفحه ها به صورت مرتب و از بالا به پایین در لیست مناسب ذخیره شوند باید در الگوریتم نیز با ترتیب درست اضافه شوند.پس با توجه به شکل ها اگر ترتیب اضافه کردن نیم صفحه ها با همان ترتیب بررسی که در قبل توصیف شد انجام شود همان ترتیب درست لحاظ خواهد شد.

بررسی صحت الگوریتم :

❖باید نشان داد که نیم صفحه های مربوط به یال های C با ترتیب درستی اضافه می شوند .

- ✓یک یال از C را در نظر می گیریم ،فرض کنید p نقطه ی بالای آن باشد آنگاه p یک نقطه ی بالایی از یک یال در C_1 یا C_2 است و یا به عنوان نقطه ی برخورد یال های e و e’ از C_1 و C_2 است .
- ✓در حالت اول : زمانی که به نقطه ی p میرسیم یک یال از C پیدا می شود .
- ✓حالت دوم : زمانی که به نقطه ی پایین تر از بین نقاط بالایی e و e’ برسیم یک یال از C پیدا می شود .
- ✓بنابراین همه نیم صفحه ها که یال های C را تعریف می کنند اضافه خواهد شد و چون برای پیدا کردن یال ها از بالاترین نقطه شروع می کنیم بنابراین با ترتیب صحیح پیدا می شوند .

قضیه ۴.۳:

اشتراک دو چند ضلعی محدب در صفحه در زمان O(n) محاسبه می شود .

زمان اجرای الگوریتم :

$$T(n) = \begin{cases} O(1) & if\ n = 1 \\ O(n) + 2T(n/2) & if\ n > 1 \end{cases}$$

نتیجه ۴.۴ :

اشتراک یک مجموعه از نیم صفحه در زمان O(nlogn) و ذخیره سازی خطی محاسبه می شود .

4.3 Incremental Linear Programming

اهداف :

- ✓تبدیل مساله قالب گیری به مساله برنامه ریزی خطی
- ✓حل مساله برنامه ریزی خطی با کمک روش های هندسه محاسباتی
- ✓ارایه یک الگوریتم تصادفی برای بهبود زمان حل مساله برنامه ریزی خطی

تا به اینجا نشان دادیم که چگونه اشتراک نیم صفحه ها را بدست آوریم یا به عبارت دیگر محاسبه همه جواب های یک مجموعه از nمحدودیت که زمان اجرای آن O(nlogn) است .
در مسئله قالب گیری نیاز نیست که همه جواب های مجموعه محدودیت های خطی را به دست آوریم ، فقط یک جواب که یک نقطه در اشتراک نیم صفحه ها است ، کافی می باشد .پس می توان الگوریتم بهتری داشته باشیم .

❖در عمل پیدا کردن یک نقطه در اشتراک نیم صفحه ها کافی است .

مساله ی پیدا کردن یک جواب از یک مجموعه محدودیت های خطی ، **بهینه سازی خطی** یا **برنامه ریزی خطی** نامیده می شود .

❖هدف پیدا کردن یک جواب خاص برای مجموعه محدودیت هاست که یک تابع خطی از متغیرها ماکزیمم کند که این تابع ، **تابع هدف** نامیده می شود .

❖به عبارتی مسئله بهینه سازی خطی به صورت زیر است :

$$\max \quad c_1x_1 + c_2x_2 + \cdots + c_dx_d$$

$$s.t. \quad a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,d}x_d \le b_1$$

$$a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,d}x_d \le b_2$$

$$\vdots$$

$$a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,d}x_d \le b_n$$

تابع هدف

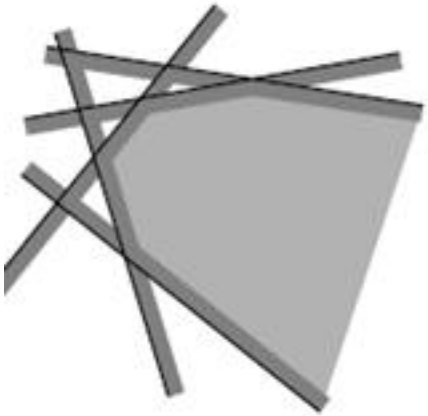
محدودیت ها

❖که c_i ، a_{ij} و b_i اعدادحقیقی هستند و d بعد مسئله است .

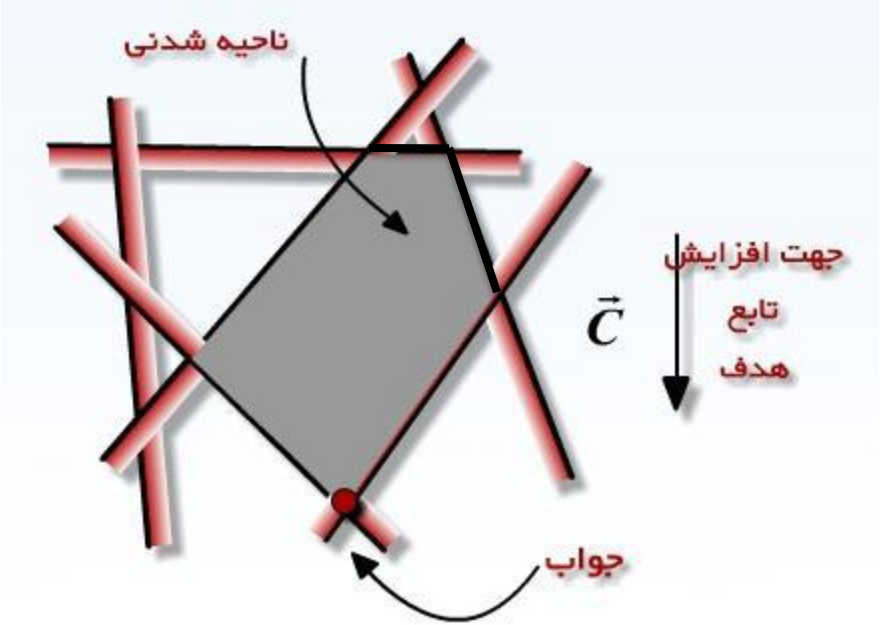
❑در مسئله مورد بحث ما d=2 یعنی مسئله دو بعدی است .

اشتراک نیم صفحه ها مجموعه نقاطی است که در همه محدودیت ها صدق می کند که **ناحیه** **feasible** نامیده می شود و نقاط این ناحیه ، نقاط feasible نامیده می شود .

✓نقاط خارج از ناحیه را نقاط infeasible گوییم .



✓تابع هدف را می توان به عنوان یک بردارجهت در نظر گرفت $f\vec{c}$ تابع هدف تعریف شده توسط بردار c را نشان می دهد .. و برای ماکزیمم کردن تابع هدف ،باید دورترین نقطه در جهت بردار C که در ناحیه feasible قرار دارد را پیدا کرد



❖برای حل این مساله در تحقیق در عملیات روش های مختلفی وجود دارد. مهم ترین این روش ها که در عمل نیز بسیار مورد استفاده قرار می گیرد، روش **سیمپلکس** است.

مراحل کار :

✓تعیین یک تابع هدف دلخواه

✓حل مسئله خطی با توجه به آن تابع هدف و محدودیت ها

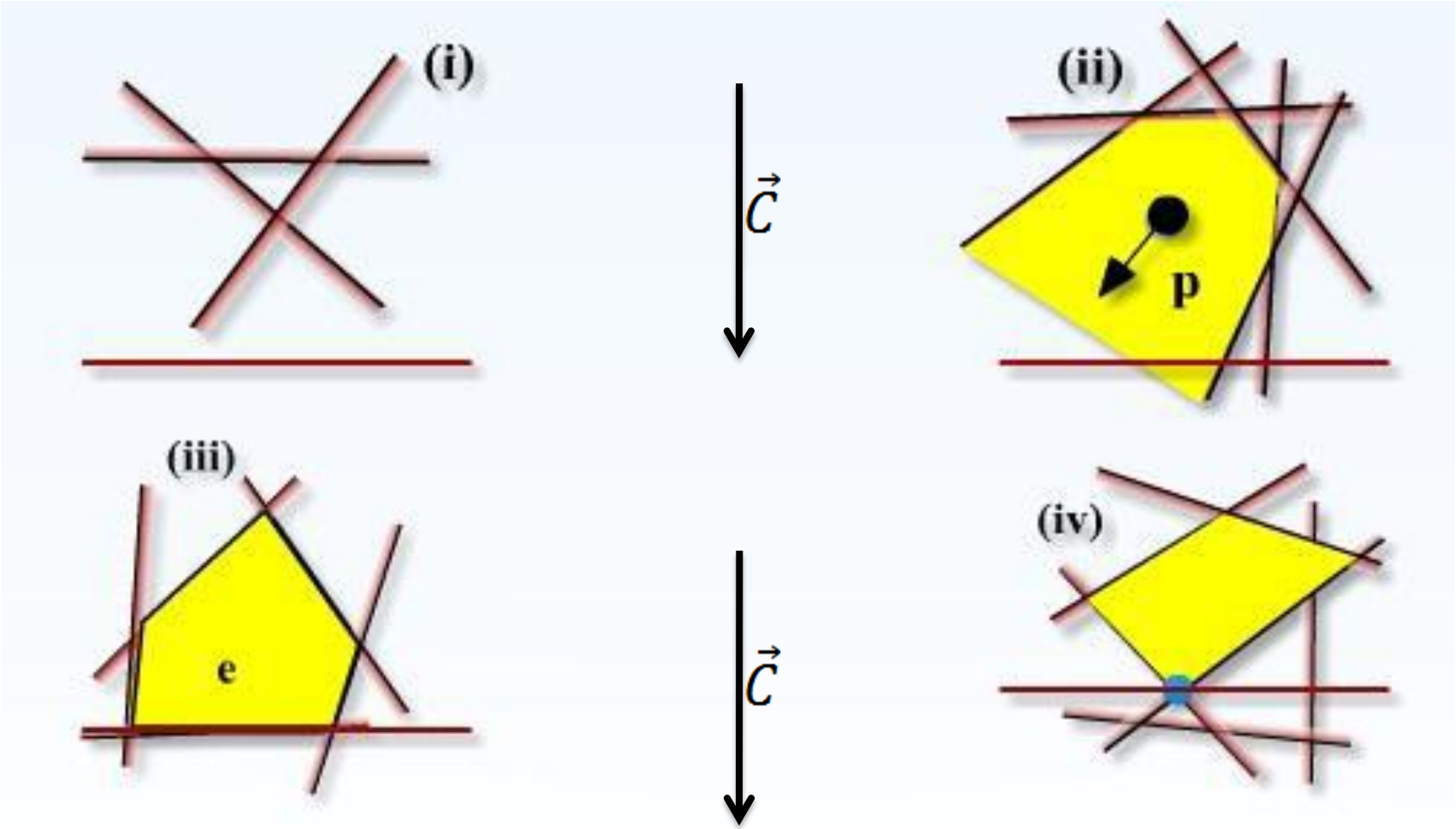
مجموعه n محدودیت خطی در مسئله خطی دو متغییره را با $H=\{h_1,h_2,\ldots,h_n\}$ نشان می دهیم .

بردار $\vec{c}=(c_x,c_y)$ تابع هدف را تعریف می کند و $f_{\vec{c}}(p)=c_x p_x+c_y p_y$ است .

هدف پیدا کردن یک نقطه مانند $p\in\mathbb{R}^2$ است به طوریکه $p\in\bigcap H$ و $f_{\vec{c}}(p)$ ماکزیمم باشد .

✓مسئله خطی را با $(H,\vec{c})$ و ناحیه feasible را با C نشان می دهیم .

۴ حالت کلی برای جواب مسئله وجود دارد :



i. مسئله خطی غیر شدنی است که جوابی برای مجموعه محدودیت ها وجود ندارد .

ii. ناحیه شدنی بیکران است که در این مورد پرتوی ρ وجود دارد که کاملا در ناحیه feasible قرار دارد ،به طوریکه تابع هدف در جهت ρ ماکزیمم خواهد شد .

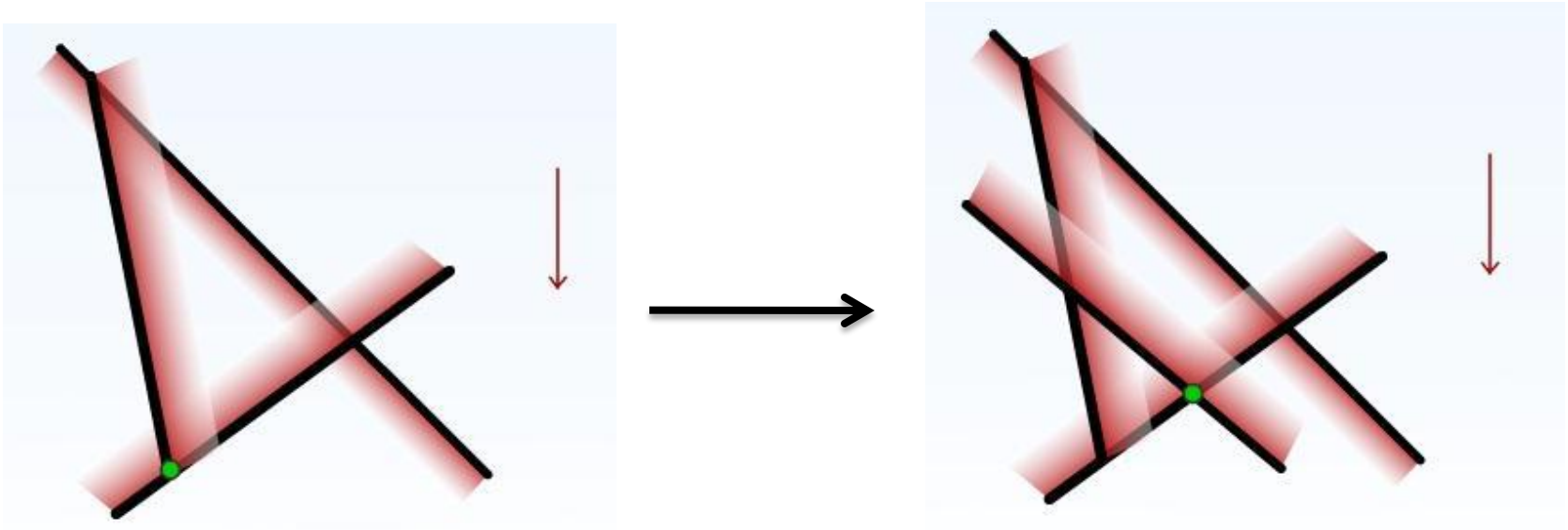
بنابراین در این مورد جوابی که مورد نظر است توصیف ρ می باشد .

iii. ناحیه feasible یک یال e دارد که جواب برای مسئله خطی وجود دارد اما یکتا نیست .

Iv اگر هیچ یک از سه حالت زیر رخ ندهد یک جواب یکتا وجود دارد که یک راس از C می باشد .

- ایده راه حل هندسی مساله استفاده از یک الگوریتم افزایشی است.

- در هر مرحله یک محدودیت اضافه می شود و برای مساله جدید یک جواب بهینه بدست می آید.بنابراین نیاز است که جواب هریک از مسائل میانی خوب و یکتا باشد . به عبارت دیگر فرض می شود که هر ناحیهfeasible میانی ، یک راس بهینه یکتا دارد مثل شکل(iv) . اما ممکن همیشه این شرط برقرار نشود.



❑بنابراین برای تضمین اینکه مسئله کراندار باشد دو محدودیت جدید به صورت زیر اضافه می کنیم :

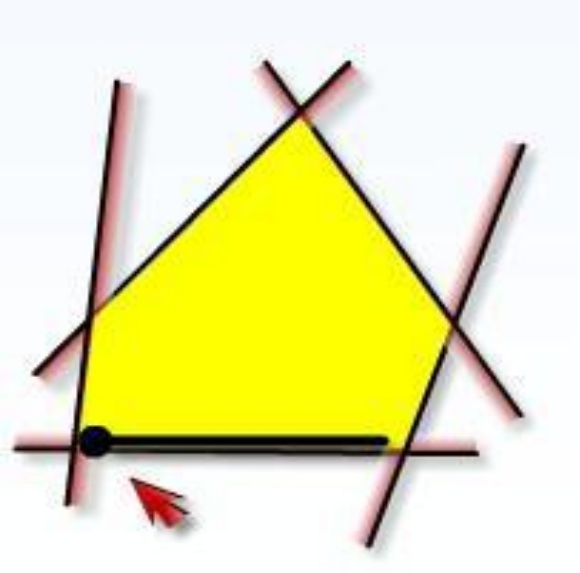
$$m_1 := \begin{cases} p_x \leqslant M & \text{if } c_x > 0 \\ -p_x \leqslant M & \text{otherwise} \end{cases}$$

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

Mباید به اندازه کافی بزرگ باشد که محدودیت های اضافه شده روی جواب بهینه تاثیر نگذارد .

با انتخاب m1 وm2 که به نیم صفحه های مجموعه H بستگی ندارند ناحیه C0=m1∩m2 یک ناحیه گوشه دار است .

✓ اگر مساله بی نهایت جواب داشته باشد، در این حالت کوچکترین نقطه در ترتیب قاموسی خواهد بود .



با این دو قرار داد هرمسئله خطی که feasible بشد یک جواب یکتا دارد که راس ناحیه feasibleاست که این راس ، راس بهینه نامیده می شود .

فرض کنید $(H, \vec{c})$ یک مسئله خطی باشد و نیم صفحه های $h_1,h_2,...,h_n$ را داشته باشیم و H_i مجموعه ای از i محدودیت اول همراه با دو محدودیت m_1 وm_2 می باشد و C_i ناحیه feasible تعریف شده توسط این محدودیت ها می باشد .

با انتخاب C0 هر ناحیه شدنی Ci یک راس بهینه یکتا دارد که با Vi نشان داده می شود.

فرض کنید :

$$H_i := \{m_1, m_2, h_1, h_2 \ldots, h_i\},$$
$$C_i := m_1 \cap m_2 \cap h_1 \cap h_2 \cap \ldots \cap h_i.$$

- جواب یکتای مرحله i را با vi نشان می دهیم .

و داریم :

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

- در نتیجه اگر برای یک i، $Ci=\emptyset$ آنگاه برای هر $j \geq i$، $C_j=\emptyset$

لم ۴.۵ :

فرض کنید $1 \leq i \leq n$ و Ci وvi به صورت قبل تعریف شده باشند ، آنگاه داریم :

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

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

اثبات :

i. فرض کنید $v_{i-1} \in h_i$ ، چون $C_i = C_{i-1} \cap h_i$ و $v_{i-1} \in C_{i-1}$ در نتیجه $v_{i-1} \in C_i$. بعلاوه نقطه بهینه در C_i نمی تواند بهتر از نقطه بهینه در C_{i-1} چون $C_i \subseteq C_{i-1}$ ، بنابراین v_{i-1} راس بهینه در C_i می باشد .

ii) فرض کنید $v_{i-1} \notin h_i$ ، برای رسیدن به تناقض فرض کنید C_i تهی نباشد و v_i نیز روی خط l_i قرار نگیرد و پاره خط $\overline{v_{i-1}v_i}$ را در نظر بگیرید .

$v_{i-1} \in C_{i-1}$ و چون $C_i \subseteq C_{i-1}$ است و همچنین داریم $v_i \in C_{i-1}$.

چون C_{i-1} محدب است ،پاره خط $\overline{v_{i-1}v_i}$ داخل C_{i-1} قرار دارد و نقطه ی v_{i-1} نقطه بهینه C_{i-1} است و تابع هدف $f_{\vec{c}}$ در جهت پاره خط $\overline{v_{i-1}v_i}$ افزیش می یابد .

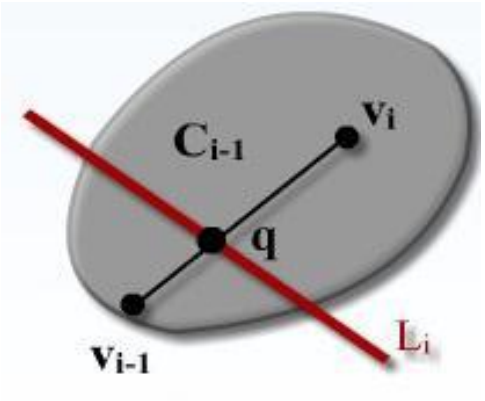
(حرکت از v_{i-1} به v_i) .

قطعاً پاره خط $\overline{v_{i-1}v_i}$ و l_i نقطه برخوردی مثل q دارند زیرا $v_{i-1} \notin h_i$ و $v_i \in C_i$

چون پاره خط $\overline{v_{i-1}v_i}$ در داخل C_{i-1} قرار دارد نقطه q نیز باید در C_i قرار داشته باشد ،

اما مقدار تابع هدف در طول پاره خط $\overline{v_{i-1}v_i}$ افزایش می یابد

بنابراین $f_{\vec{c}}(q) > f_{\vec{c}}(v_i)$ در تناقض با تعریف v_i است .

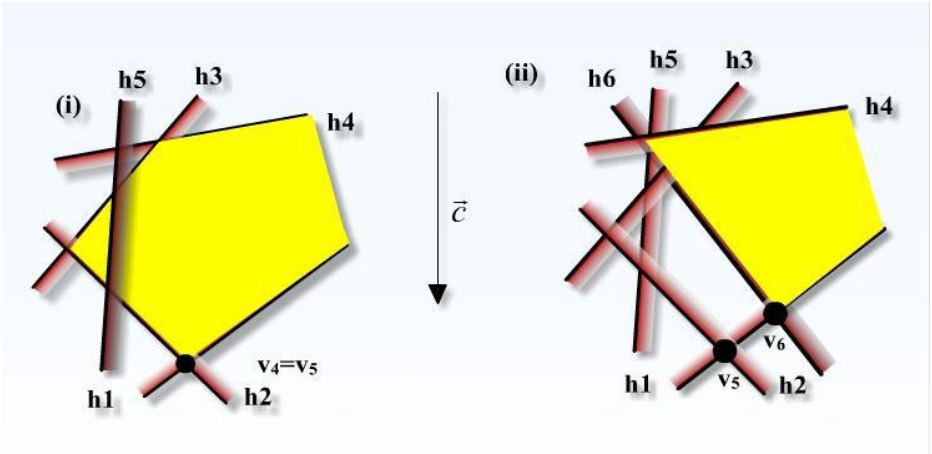


❖دو حالتی که پس از اضافه کردن نیم صفحه جدید رخ می دهد.

راس بهینه v_4 بعد از اضافه کردن ۴ نیم صفحه اول بدست می آید و با اضافه کردن نیم صفحه h_5 راس بهینه همان v_4 باقی می ماند .

راس بهینه بعد از اضافه کردن نیم صفحه h_6 در h_6 قرار نمیگیرد ، بنابراین باید یک راس بهینه جدید پیدا شود .

برطبق لم ۴.۵ راس v_6 روی خط مرزی h_6 قرار دارد ، که در شکل نشان داده شده است .



پیدا کردن راس بهینه :

فرض کنید راس بهینه جاری v_{i-1} باشد که در نیم صفحه بعدی (h_i) قرار ندارد ،مسئله به صورت زیر حل می شود :

هدف پیدا کردن نقطه p روی l_i است بطوریکه $f_{\vec{c}}(p)$ ماکزیمم شود و $p \in h$ برای هر $h \in H_{i-1}$.

برای سادگی مسئله فرض می شود که l_i عمودی نباشد بنابراین می توانیم آنرا با یک مختص x نشان میدهیم و تابع $\overline{f_{\vec{c}}}: R \longrightarrow R$ را تعریف می کنیم به طوری که:

$$f_{\vec{c}}(p) = \overline{f_{\vec{c}}}(p_x) \quad \text{برای} \quad p \in l_i$$

❖مختص x نقطه برخورد نیم صفحه h و l_i را $\sigma(h, l_i)$ می نامیم .

❖اگر برخوردی وجود نداشته باشد آنگاه می توان مسئله خطی را infeasible گزارش کرد .

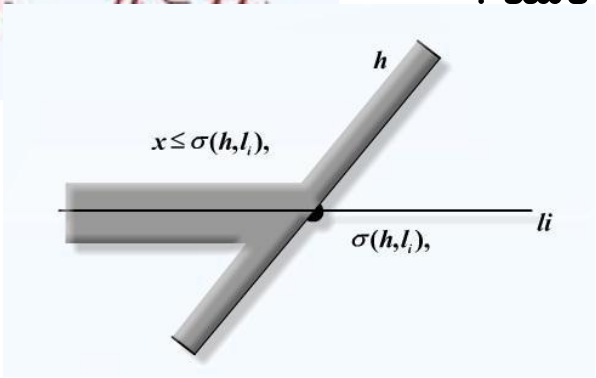
❖اگر محدودیت h برای همه نقاط روی l_i برقرار باشد میتوان آن محدودیت را نادیده گرفت .

مساله به صورت یک متغیره زیر در می آید:

$$\begin{aligned} &\text{Maximize} && \overline{\overline{fc}}(x) \\ &\text{subject to} && x \geq \sigma(h,l_i), \quad h \in H_i \\ & && x \leq \sigma(h,l_i), \quad h \in H \end{aligned}$$

اگر $l_i \cap h$ از چپ کران دار باشد :

اگر $l_i \cap h$ از راست کران دار باشد :



$$X_{left} = \max_{h \in H_{i-1}} \{ \sigma(h, l_i) : \text{از چپ کراندار باشد } l_i \cap h \}$$

$$X_{right} = \min_{h \in H_{i-1}} \{ \sigma(h, l_i) : \text{از راست کراندار باشد } l_i \cap h \}$$

✓ناحیه feasible مسئله خطی یک متغیره بازه [Xleft :Xright] بنابراین اگر Xleft>Xright آنگاه مسئله infeasible است ، در غیر اینصورت نقطه بهینه روی l_i ، نقطه Xleft یا Xright است که با توجه به تابع هدف تعیین می شود .

لم ۴.۶ :

یک مسئله خطی یک بعدی در زمان خطی حل می شود ،بنابراین اگر مورد دوم از لم ۴.۵ رخ دهد آنگاه در زمان O(i)می توان راس بهینه جدید vi را محاسبه کرد یا تصمیم گرفت که مسئله infeasible است.

Algorithm 2DBOUNDEDLP($H, \vec{c}, m_1, m_2$)

Input. A linear program $(H \cup \{m_1, m_2\}, \vec{c})$, where H is a set of n half-planes, $\vec{c} \in \mathbb{R}^2$, and m_1, m_2 bound the solution.

Output. If $(H \cup \{m_1, m_2\}, \vec{c})$ is infeasible, then this fact is reported. Otherwise, the lexicographically smallest point p that maximizes $f_{\vec{c}}(p)$ is reported.

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

لم ۴.۷:

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

اثبات :

برای اینکه نشان داد الگوریتم به درستی جواب را پیدا می کند باید نشان داد که بعد از هر مرحله (بعد از اضافه کردن نیم صفحه جدید h_i) نقطه v_i بدست آمده نقطه بهینه برای C_i است که از لم ۴.۵ نتیجه می شود .

اگر مسئله خطی یک متغیره رو L_i، infeasible باشد آنگاه C_i تهی است و در نتیجه $C = C_n \subseteq C_i$ تهی است که به این معناست که مسئله خطی infeasible است .

✓باتوجه به اینکه نیم صفحه ها یکی یکی اضافه می شوند n مرحله داریم و زمانی که در مرحله i ام صرف می شود برابر است با حل مسئله خطی یک متغیره که O(i) است .

بنابراین زمان اجرای الگوریتم برابر است با :

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

آیا می توان زمان اجرای الگوریتم را کمتر کرد ؟

در واقع ما هزینه هر مرحله i را توسط O(i) کراندار کردیم ،اما زمان مرحله iام همیشه به این سقف نمیرسد .

زمانی که $v_{i-1} \notin h_i$ مرحله iام به زمان $\theta(i)$ نیاز دارد ولی زمانی که $v_{i-1} \in h_i$ باشد مرحله iام در زمان ثابت حل می شود .

بنابراین اگر بتوان زمان تغییر راس بهینه را محدود کرد ممکن است بتوان زمان اجرای بهتری ارائه کرد .

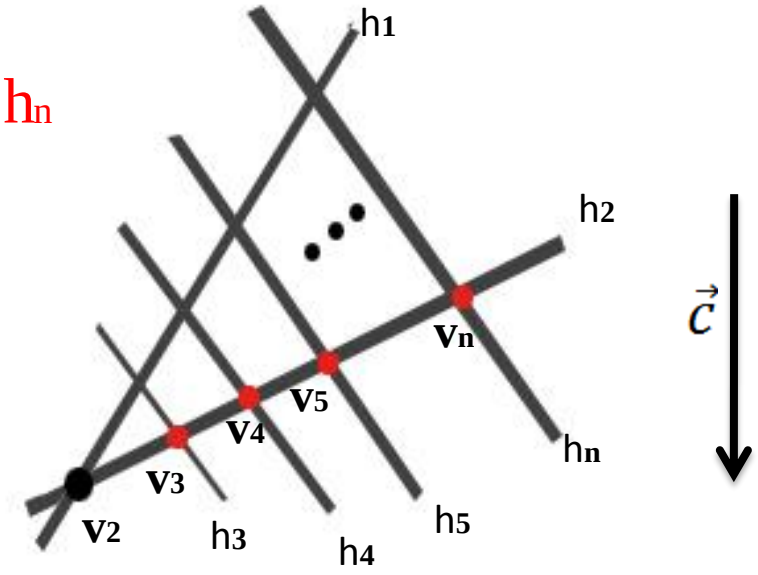
4.4 Randomized Linear Programming

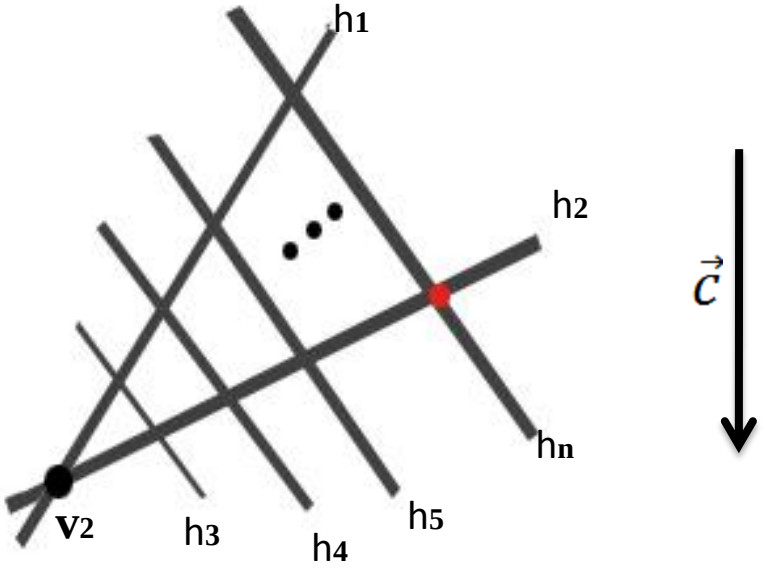
✓ آیا ترتیب اضافه کردن نیم صفحه ها در زمان اجرای الگوریتم موثر است ؟

در تصویراول زمانی که نیم صفحه ها را باترتیب $h_1,h_2,h_3...,h_n$ اضافه میکنیم ، با اضافه کردن هر نیم صفحه راس بهینه تغییر می کند اما در تصویر دوم اگر به ترتیب $h_1,h_2,h_n,h_{n-1},...h_3$ نیم صفحه ها رو اضافه کنیم راس بهینه عوض نخواهد شد.

$h_1 , h_2, h_n ..., h_3$

$h_1 , ..., h_n$





❖ پس برای کاهش زمان اجرای الگوریتم باید ترتیب نیم صفحه ها اصلاح شود.

❖ یافتن ترتیبی که گفتیم ساده نیست. اما می توان از یک ترتیب تصادفی استفاده کرد .

الگوریتم تصادفی :

Algorithm 2DRANDOMIZEDBOUNDEDLP(H, $\vec{c}$, m_1, m_2)

Input. A linear program $(H \cup \{m_1, m_2\}, \vec{c})$, where H is a set of n half-planes, $\vec{c} \in R_2$, and m_1, m_2 bound the solution.

Output. If $(H \cup \{m_1, m_2\}, \vec{c})$ is infeasible, then this fact is reported. Otherwise, the lexicographically smallest point p that maximizes $f_{\vec{c}}(p)$ is reported.

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

Algorithm RANDOMPERMUTATION(A)

Input. An array A[1···n].

Output. The array A[1···n] with the same elements, but rearranged into a random permutation.

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

❖ Random(k) یک تولید کننده عدد تصادفی است که عدد صحیح k را به عنوان ورودی میگیرد و یک عدد صحیح تصادفی بین ۱ تا k در زمان ثابت تولید می کند .

زمان اجرای الگوریتم تصادفی :

- ✓زمان اجرا به ترتیب تصادفی که محاسبه می شود بستگی دارد . چون n نیم صفحه داریم ، n! جایگشت وجود دارد .به عبارتی n! حالت برای اجرای الگوریتم وجود دارد که هر کدام زمان اجرای مربوط به خود را دارند .
- ✓چون انتخاب جایگشت تصادفی است احتمال هر کدام از زمان های اجرا برابر است .
- ✓بنابراین برای بدست آوردن زمان اجرای الگوریتم ، میانگین زمان اجرا روی n! ترتیب مختلف را بدست می آوریم و آنرا زمان مورد انتظار اجرای الگوریتم گوییم .

لم ۴.۸ :

برای مسئله خطی دو متغیره با n محدودیت ،زمان مورد انتظار اجرای الگوریتم $O(n)$ است و به حافظه خطی نیاز دارد .

✓یادآوری امید ریاضی و یا میانگین متغیر تصادفی :

$f(x)$: تابع چگالی احتمال

$$E(x)=\begin{cases}\sum xf(x)\\ \int xf(x)\end{cases}$$

اثبات :

❖ زمان اجرای الگوریتم RANDOMPERMUTATION ، O(n) است .

بنابراین تنها مسئله تحلیل زمان مورد نیاز برای اضافه کردن نیم صفحه های h_1 تا h_n است .

❖ اضافه کردن یک نیم صفحه زمانی که راس بهینه تغییر نکند به زمان ثابت نیاز دارد .

زمانی که راس بهینه تغییر کند نیاز به حل یک مسئله خطی یک متغیره است .

اکنون کرانی برای زمان مورد نیاز مسائل خطی یک متغیره باید بدست آورد .

✓ برای اضافه کردن صفحات، متغیر تصادفی X_i را بدین صورت تعریف می کنیم:

$$Xi = \begin{cases} 1 & v_{i-1} \notin h_i \\ 0 & otherwise \end{cases}$$

چون زمان حل مسئله خطی یک متغیره روی i محدودیت در زمان O(i) حل می شود زمان ، اجرای الگوریتم بعد از اضافه کردن همه صفحات h_1 تا h_n برابر است با :

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

✓با توجه بودن به خطی بودن امید ریاضی:

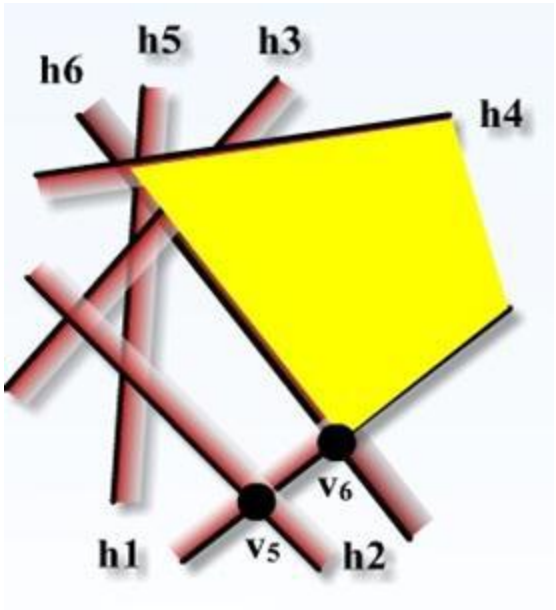
$$E(\sum_{i=1}^{n} O(i)\, X_i) = \sum_{i=1}^{n} O(i)\,.\,E(X_i)$$

E(Xi) چیست؟

احتمال اینکه $v_{i-1} \notin h_i$.

❖برای تحلیل E(Xi) از روشی به نام BackWards Analysis استفاده می شود .

❖یعنی فرض می شود که الگوریتم انجام گرفته و راس بهینه v_nبدست آمده، حال صفحه h_nرا حذف می کنیم .



❖**با چه احتمالی نقطه بهینه تغییر می کند؟**

✓وقتی v_n تغییر می کند که راسی از C_{n-1} در جهت افزایش بردار $\vec{c}$ نباشد، و این زمانی امکان پذیر است که h_n یکی از نیم صفحه هایی باشد که v_n را تعریف می کند .

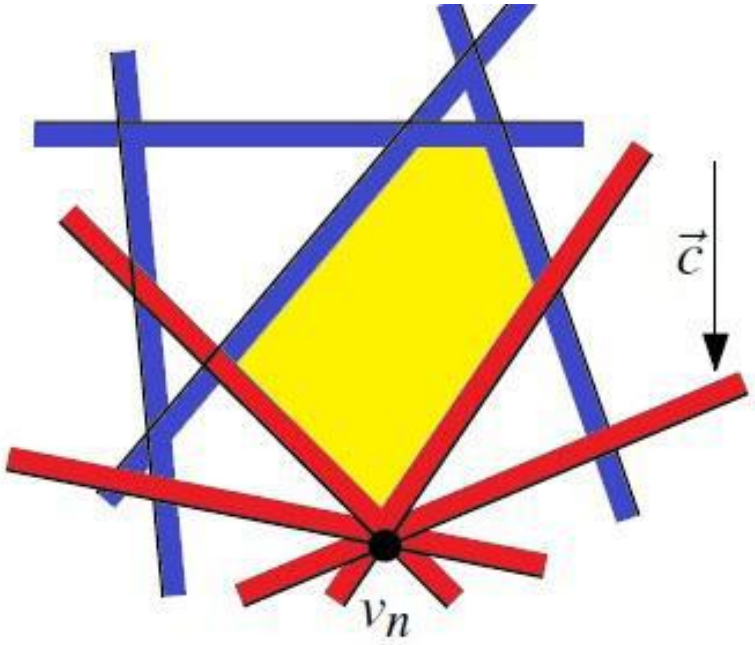
✓ v_n راس C_n می باشد،حداقل توسط دو تا از نیم صفحه ها تعریف می شوند .

✓از آنجا که نیم صفحه ها به طورتصادفی اضافه شده اند، احتمال اینکه h_n یکی از نیم صفحه های تعریف کننده v_n باشد حداکثر $2/n$ می باشد.

چرا حداکثر 2/n؟

✓ اولاً اگر v_n توسط بیش از دو نیم صفحه تعریف شده باشد، آنگاه حذف یکی از نیم صفحه ها باعث تغییر v_n نمیشود .

✓ دوماً ممکن است v_n توسط m_1 و m_2 تعیین شده باشد که در انتخاب تصادفی h_n قرار ندارند .



نتیجه : احتمال E(xi) حداکثر 2/n است .

حال اگر همین روند Backwardsبرای i معادله اول Hدرنظر گرفته شود، باز هم داریم:

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

در نتیجه زمان کل مورد انتظار برای مسئله خطی یک بعدی برابر است با:

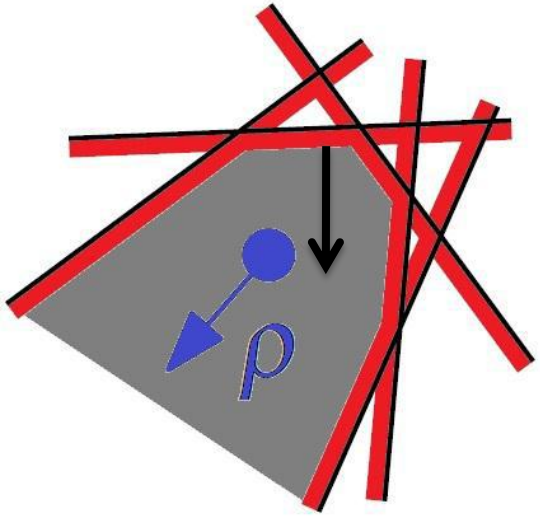
$$\sum_{i=1}^{n} O(i).E(X_i) \longrightarrow \sum_{i=1}^{n} O(i)\ 2/i = O(n)$$

4.5 Unbounded Linear Programs

❖در قسمت قبل برنامه خطی نامحدود را با اعمال دو محدودیت اضافی کراندار کردیم، اما این همیشه ممکن نیست و حتی گاهی برنامه خطی محدود می شود ولی کران یا همان مرز ناحیه ممکن بسیار بزرگ می شود به گونه ای که قابل شناخت نیست.

❖چگونه کراندار بودن یا نبودن برنامه خطی را $(H, \vec{c})$ تشخیص بدهیم ؟

$\vec{\eta}(h)$: بردار نرمال نیم صفحه h و عمود بر کران h رو به ناحیه feasible



لم ۴.۹:

یک مسئله خطی $(H, \vec{c})$ غیرکراندار است اگر و فقط اگریک بردار $\vec{d}$ وجود داشته باشد بطوریکه $\vec{d}.\vec{c} > 0$ و $\vec{d}.\vec{\eta}(h) \geq 0$ برای هر $h \in H$ و مسئله خطی $(H', \vec{c})$ feasible است که :

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

اثبات :

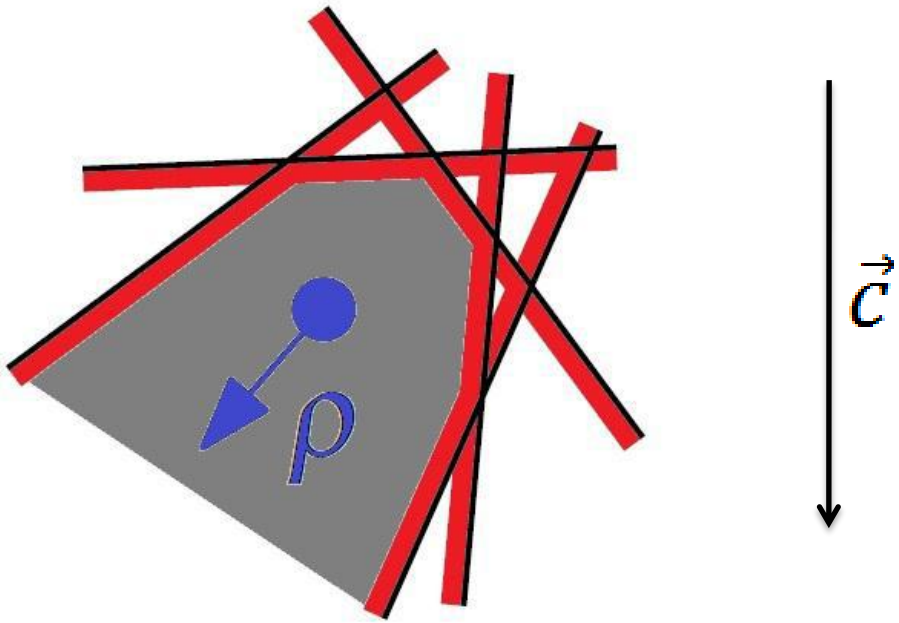
طرف رفت: قبلا گفته شد که بیکران بودن C به معنی وجود اشعه ρ به گونه ایست که کامل در ناحیه feasible قرار بگیرد و تابع هدف در طول ρ مقادیر بزرگ دلخواه بگیرد.

✓P نقطه شروع ρ است و $\vec{d}$ بردار جهت ρ می باشد .

بنابراین می توان ρ را به صورت رو برو نمایش داد : $\rho = \{p + \lambda\vec{d} : \lambda > 0\}$

تابع هدف f بشترین مقدار را می گیرد اگر و تنها اگر $\vec{d}.\vec{c} > 0$ به عبارت دیگر $\vec{d}.\overrightarrow{\eta}(h) \geq 0$

.



طرف برگشت : چون $(H', \vec{c})$ یک مسئله feasible است یک نقطه $p_0 \in \bigcap_{h \in H'} h$ وجود دارد .اشعه $\rho_0 = \{p_0 + \lambda \vec{d} : \lambda > 0\}$ را فرض می کنیم . چون $\vec{d}.\vec{\eta}(h) = 0$ برای $h \in H'$ اشعه ρ_0 کاملاً در هر $h \in H'$ قرار می گیرد بعلاوه چون تابع هدف $f_{\vec{c}}$ مقادیر بزرگ دلخواه را در طول ρ_0 خواهد داشت.

برای یک نیم صفحه $h \in H \backslash H'$ داریم $\vec{d}.\vec{\eta}(h) > 0$ که ایجاب میکند که یک پارامتر λ_h وجود داشته باشد به طوریکه $p_0 + \lambda \vec{d} \in h$ برای هر $\lambda \geq \lambda_h$.

فرض کنید $\lambda' = \max_{h \in H \backslash H'} \lambda_h$ و $p = p_0 + \lambda' \vec{d}$

در نتیجه $\rho = \{p + \lambda \vec{d} : \lambda > 0\}$ کاملاً در هر نیم صفحه ی $h \in H$ قرار می گیرد ، بنابراین $(H, \vec{c})$ بدون کران است .

اگر مختصات را طوری دوران دهیم که بردار $\vec{c}$ به صورت عمودی به سمت بالا باشدداریم $\vec{c}$=(0,1)و هر بردار جهت d=(dx,dy) که را می توان به صورت d=(dx,1) نوشت که با dxروی خط $y=1$نشان داده میشود.

$$\vec{\eta}(h) = (\eta_x, \eta_y)$$

$$\vec{d} \cdot \vec{\eta}(h) = d_x \eta_x + \eta_y \geqslant 0$$

$$d_x \eta_x \geqslant -\eta_y.$$

بنابراین یک دستگاه از نامعادله خطی یا به عبارت دیگر یک مسئله خطی یک متغیره $\overline{H}$ داریم .