
بسم الله الرحمن الرحيم

دانشگاه صنعتی اصفهان

دانشکده‌ی مهندسی برق و کامپیوتر

برنامه‌ریزی فصلی، کاربردها و ابزارهای حل

گزارش پروژه‌ی کارشناسی

آرش ماری‌اوریاد

۹۵۳۲۶۰۳

استاد پروژه

دکتر حسین فلسفین

مهر ۱۳۹۹

فهرست مطالب

چکیده

یکی از مهم‌ترین و پرکاربردترین رویکردها در حل مسائل علوم کامپیوتر و دنیای واقعی، مدل‌کردن مسئله در قالب یک مسئله‌ی بهینه‌سازی مقید[۱] و سپس حل آن جهت کمینه یا بیشینه کردن مقدار مقدار تابع هدف[۲] و محاسبه کردن متغیرهای دخیل در مسئله می‌باشد. از این رو امروزه شناخت انواع مختلف روش‌های بهینه‌سازی مقید و مدل‌سازی از یک سو و تسلط بر رویکردهای متنوع حل آن‌ها از سوی دیگر به عنوان ابزاری قدرتمند برای هر مهندس یا محقق مشغول در حوزه‌های مختلف تئوری و عملی ضروری می‌نماید.

در اکثر رویکردهای مرسوم بهینه‌سازی مقید مانند برنامه‌ریزی خطی[۳]، برنامه‌ریزی صحیح[۴] و برنامه‌ریزی چندجمله‌ای[۵]، قیود به صورت ترکیب عطفی[۶] چند گزاره‌ی منطقی یا عبارات جبری ریاضی طراحی و بیان می‌شوند. در این پروژه قصد داریم مدل دیگری از بهینه‌سازی را بررسی کنیم که هسته‌ی اصلی آن برپایه‌ی ترکیب فصلی[۷] قیود است. بررسی‌های گوناگون نشان داده‌است که بهره‌گیری از این ابزار می‌تواند ره‌گشای دسته‌ی مهمی از مشکلاتی باشد که در مدل‌سازی با استفاده از روش‌های بهینه‌سازی مرسوم به وجود می‌آید.

در واقع این رویکرد که برنامه‌یزی فصلی[۸] نام دارد، می‌تواند با بهره‌گیری از ترکیبات فصلی در طراحی قیود، ما را در مدل‌کردن دسته‌ی بزرگی از مسائل یاری برساند. به خصوص که مزیت این روش در مدل‌سازی مسائلی که به نوعی شامل تصمیم‌گیری‌های گسسته می‌باشند، به خوبی نمایان است.

در این گزارش سعی شده‌است ضمن معرفی روش ذکر شده و کاوش در تفاوت‌های آن با دیگر رویکردهای مرسوم، به بررسی تعدادی مسئله‌ی معروف مدل شده در قالب برنامه‌ریزی فصلی، روش‌های حل آن‌ها و ابزارهای برنامه‌نویسی موجود در این زمینه نیز پرداخته‌شود.

کلمات کلیدی: بهینه‌سازی، ترکیب فصلی[۹]، برنامه‌ریزی فصلی، روش پوش محدب[۱۰]، روش Big-M

---

[۱] Constrained Optimization Problem

[۲] Objective Function

[۳] Linear Programming

[۴] Integer Programming

[۵] Quadratic Programming

[۶] Conjunctive

[۷] Disjunctive

[۸] Disjunctive Programming

[۹] Disjunction Sets

[۱۰] Convex Hull

فصل اول: مقدمه

انسان در طول تاریخ همواره با مسائل گوناگونی برآمده از تفکر انتزاعی یا دنیای واقعی مواجه شده‌است که دسته‌ای از این مسائل از نوع بهینه‌سازی می‌باشند یا می‌توان آن‌ها را در قالب یک مسئله‌ی بهینه‌سازی مدل کرد. لذا از دیرباز بررسی انواع مسائل بهینه‌سازی و روش‌های حل آن‌ها از یک‌سو و بررسی رویکردهای مدل‌کردن مسائل گوناگون در قالب یک مسئله‌ی بهینه‌سازی از سوی دیگر همواره مورد توجه بوده‌است.

در یک مسئله‌ی بهینه‌سازی، ما به دنبال کمینه یا بیشینه کردن یک تابع هدف می‌باشیم. اگر حل این مسئله مستلزم در نظر گرفتن تعدادی قید[۱۱] یا محدودیت باشد که در اکثر مواقع نیز چنین است، مسئله بیان‌شده بهینه‌سازی مقید نامیده می‌شود. در واقع در بهینه‌سازی مقید در عین تلاش برای جست‌وجوی مقدار بهینه‌ی یک تابع، قیودی نیز باید ارضا[۱۲] شوند. تابع هدف و قیود به طبیعت و ذات مسئله وابسته می‌باشند. در نهایت می‌توان گفت بهینه‌سازی مقید هنر یافتن بهترین جواب در بین وضعیت‌های ممکن است.

امروزه مسائل بهینه‌سازی در بسیاری از رشته‌های تئوری و مهندسی نظیر علوم کامپیوتر، مهندسی کامپیوتر، مهندسی صنایع، تحقیق در عملیات، اقتصاد و ... مورد استفاده قرار می‌گیرند. همچنین در طول قرن‌های متمادی، توسعه روش‌های تولید جواب و حل مسأله‌ی بهینه‌سازی، یکی از حوزه‌های مهم تحقیقاتی در علم ریاضیات نیز بوده است و اهمیت آن‌ها در طی چند سال گذشته چند برابر شده‌است.

---

[۱۱] Constraint
[۱۲] Satisfy

در این پروژه ما به دنبال بررسی روشی هستیم که هسته‌ی اصلی آن در طراحی قیود به صورت ترکیبات فصلی می‌باشد. با گام نهادن در این مسیر ما مسلط به ابزاری می‌شویم که می‌تواند در مدل‌سازی طیف وسیعی از مسائل دنیای واقعی به ما یاری برساند. لذا در گزارش پیش‌رو ضمن آشنا شدن با ساختار این رویکرد که برنامه‌ریزی فصلی نام دارد، از یک سو با مثال‌هایی روبه‌رو می‌شویم که موجب عمق بخشیدن به درک ما از این رویکرد می‌شوند و از سوی دیگر در روش‌های حل و ابزارهای پیاده‌سازی برنامه‌ریزی فصلی کند و کاو می‌نماییم.

برای ورود به دنیای برنامه‌ریزی فصلی، بهتر است ابتدا با سایر رویکردهای مرسوم در بهینه‌سازی ریاضی آشنا شویم.

مطابق آن‌چه بیان شد یک مسئله‌ی بهینه‌سازی از اجزای اصلی زیر تشکیل شده‌است:

- تابع هدف که سعی بر کمینه یا بیشینه کردن آن داریم
- قیود یا محدودیت‌هایی که از طبیعت مسئله به دست‌آمده‌اند
- قیودی که کران یا مقادیر مجاز متغیرهای دخیل در مسئله را مشخص می‌نمایند

اگر در یک مسئله‌ی بهینه‌سازی، متغیرهای موجود در مسئله پیوسته بوده و تابع هدف و قیود نسبت به آن متغیرها خطی[۱۳] باشند، مسئله مورد نظر برنامه‌ریزی خطی نامیده می‌شود. به همین ترتیب اگر تابع هدف یا حداقل یکی از قیود نسبت به متغیرهای دخیل در مسئله چند جمله‌ای باشند، ما با یک برنامه‌ریزی چندجمله‌ای روبه‌رو هستیم. نوع مهم دیگری از بهینه‌سازی برنامه‌ریزی خطی صحیح[۱۴] نام دارد و در صورتی رخ می‌دهد که حداقل یکی از متغیرهای مسئله فقط مقادیر صحیح دریافت نماید. از انواع دیگر رویکردهای بهینه‌سازی می‌توان برنامه‌ریزی نیمه‌معین[۱۵] و برنامه‌ریزی برداری[۱۶] را نیز نام برد. برای مطالعه‌ی بیشتر در رابطه با رویکردهای مطرح شده می‌توان به [۱] و [۲] مراجعه نمود.

لازم به ذکر است که برنامه‌ریزی خطی در نظر پیچیدگی در کلاس مسائل P قرار دارد بدین معنا که پیچیدگی زمانی مورد برای حل آن نسبت به تعداد متغیرها و قیود از نوع چندجمله‌ای می‌باشد. برای این رویکرد بهینه‌سازی یکی از سه روش زیر مورد استفاده می‌تواند مورد استفاده قرار گیرد.

- روش Simplex
- روش Ellipsoid
- روش Interior-Point

که روش اول دارای بدترین حالت[۱۷] نمایی می‌باشد [۳]و دو روش دیگر برای تمام نمونه‌ها پیچیدگی زمانی چند جمله‌ای را تضمین می‌کنند[٤] و [٥]. اشاره به این موضوع نیز ممکن است خالی از لطف نباشد که برنامه‌ریزی خطی

---

[۱۳] Linear
[۱٤] Integer Linear Programming
[۱٥] Semidefinite Programming
[۱٦] Vector Programming
[۱۷] Worst Case

صحیح را که در کلاس پیچیدگی NP-Hard قرار می‌گیرد، می‌توان با بهره‌گیری از روش شاخه و کران[۱۸] و حل تعدادی (نه لزوماً چندجمله‌ای) نمونه از برنامه‌ریزی خطی حل نمود.

حال با بهره‌گیری از مفهوم بهینه‌سازی و آشنایی ابتدایی با رویکردهای مرسوم آن می‌توانیم به معرفی برنامه‌ریزی فصلی، مدل‌سازی در قالب این بستر و روش‌های حل آن بپردازیم.

ساختار کلی این گزارش بدین صورت است که در فصل دوم گزارش به معرفی و بررسی ساختار برنامه‌ریزی فصلی خواهیم پرداخت. فصل سوم دربردارنده‌ی مثال‌هایی از مدل‌سازی مسائل دنیایی واقعی در بستر یک برنامه‌ریزی فصلی خواهد بود. در فصل چهارم روش‌های حل یک مسئله‌ی برنامه‌ریزی فصلی را بررسی می‌کنیم. ابزارهای برنامه‌نویسی حل و پیاده‌سازی این رویکرد در فصل پنجم مورد توجه و بررسی قرار می‌گیرند. در نهایت نیز فصل ششم به نتیجه‌گیری و معرفی زمینه‌هایی برای تحقیق در آینده می‌پردازد.

---

[۱۸] Branch and Bound

فصل دوم: برنامه‌ریزی فصلی

در بخش قبل به طور مختصر با مفهوم بهینه‌سازی ریاضی آشنا شده و انواع مختلف آن مانند برنامه‌ریزی خطی، برنامه‌ریزی صحیح، برنامه‌ریزی نیمه‌معین و ... را به همراه روش‌های حل مرور کردیم.

حال می‌خواهیم با نوع دیگری از رویکردهای بهینه‌سازی تحت عنوان برنامه‌ریزی فصلی آشنا شویم. در این بخش مفهوم برنامه‌ریزی فصلی بیان شده و در مورد اجزای آن سخن به عمل می‌آید. سپس تعدادی مثال مطرح می‌شود که با بهره‌گیری از روش فوق به دنبال کمینه یا بیشینه کردن تابع هدف می‌باشند. در انتها نیز ضمن معرفی حالت کلی‌تر برنامه‌ریزی فصلی تحت نام برنامه‌ریزی فصلی تعمیم‌یافته، مثال‌هایی در این باب مطرح می‌شود.

ساختار برنامه‌ریزی فصلی

به بیان ساده برنامه‌ریزی فصلی در واقع نوعی رویکرد بهینه‌سازی است که در آن قیود مسئله در قالب ترکیب‌های فصلی ظاهر می‌شوند. به طور کلی یک مسئله‌ی برنامه‌ریزی فصلی شامل اجزای اصلی زیر می‌باشد:

- تابع هدف که قرار است کمینه یا بیشینه شود.
- ترکیب‌های فصلی که بیان‌گر قیود و محدودیت‌های مسئله در فرایند بهینه‌سازی هستند و به طبیعت مسئله بستگی دارند.
- قیودی که کران متغیرهای پیوسته یا مقادیر مجاز برای متغیرهای گسسته در مسئله را مشخص می‌نمایند.

به همین ترتیب منظور از برنامه‌ریزی فصلی خطی، یک برنامه‌ریزی خطی با قیود فصلی می‌باشد. بر همین اساس این رویکرد را می‌توان نوعی بهینه‌سازی روی اجتماعی از پلی‌هدرون‌ها دانست [٦].

یک برنامه‌ریزی فصلی ساده را می‌توان در حالت کلی به صورت آنچه که در فرمول ۲-۱ نمایش داده‌شده، بیان کرد [٦].

$$\begin{gathered} min/max\ \ f(x) \\ subject\ to\colon\ \ \bigcup_{i \in K} [\ g(x)_i \le b_i\ ] \\ LB_i \le x_i \le UB_i \qquad ١ \le i \le n \end{gathered}$$

فرمول ۲-۱

که در این معادله $x \in R^n$ و $\cup$ نماد عملیات or منطقی و LB_i و UB_i به ترتیب بیانگر کران‌های پایین و بالا برای x_i می‌باشند.

به عنوان مثال فرمول ۲-۲ یک برنامه‌ریزی فصلی ساده می‌باشد.

$$\begin{gathered} max\ \ ٢x + ٣y \\ subject\ to\colon\ \ [x + ٢y \le -٢]\ \cup [-٣x - y \ge ١] \\ ٢ \le x \le ٣ \\ y \le ١ \end{gathered}$$

فرمول ۲-۲

دقت شود که روش‌های حل برنامه‌ریزی فصلی در فصل آینده مورد بحث قرار می‌گیرد و آنچه در این بخش اهمیت دارد، تسلط یافتن بر ساختار این رویکرد بهینه‌سازی از یک سو و درک اهمیت آن در مدل کردن پاره‌ای از مسائل از سوی دیگر می‌باشد.

همچنین درخور توجه است که تجربیات مختلف در زمینه‌ی مدل‌سازی نشان داده‌است که برنامه‌ریزی فصلی برای مسائل که شامل تصمیمات گسسته[١٩] می‌باشند، به خوبی عمل می‌نماید [٦]. زیرا می‌توان تصمیم‌گیری‌های گسسته را به راحتی در قالب فرم‌های فصلی بیان کرد. مثال‌هایی از این موارد را در بخش‌های آتی خواهیم دید.

---

[١٩] Discrete Decisions

به طور معمول ما با فرمی تعمیم‌یافته نسبت به فرمول ۲-۱ در ارتباط هستیم که به آن برنامه‌ریزی فصلی تعمیم‌یافته[۲۰] می‌گویند که در بخش بعدی با آن آشنا خواهیم شد.

برنامه‌ریزی فصلی تعمیم‌یافته

در بخش قبل با مفهوم و ساختار برنامه‌ریزی فصلی آشنا شدیم. حال می‌خواهیم با تعمیم مفاهیم قبلی به تعریف برنامه‌ریزی فصلی تعمیم‌یافته پرداخته و مثال‌هایی را در این باب مطرح نماییم. در فصل‌های آینده نیز روش‌های حل این رویکرد بهینه‌سازی را بررسی می‌کنیم و به حل مثال‌های مطرح شده می‌پردازیم.

یک برنامه‌ریزی فصلی عمومیت یافته شامل متغیرهای پیوسته، متغیرهای گسسته، عبارات جبری ریاضی، ترکیبات فصلی و گزاره‌های منطقی می‌باشد که فرم کلی آن در فرمول ۲-۳ آمده‌است [٦].

$$
\begin{aligned}
& Min\ Z = f(x) + \textstyle\sum_{k \in K} c_k \\
& s.t.\ \ g(x) \leq 0 \\
& \bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ r_{ik}(x) \leq 0 \\ c_k = \gamma_{ik} \end{bmatrix} \qquad k \in K \\
& \Omega\ (Y) = True \\
& x^{lo} \leq x \leq x^{up} \\
& x \in R^n\,,\ c_k \in R^1,\ Y_{ik} \in \{True, False\}
\end{aligned}
$$

فرمول ۲-۳

تابع $f: R^n \to R^1$، تابعی از متغیرهای پیوسته‌ی x می‌باشد و تابع هدف نامیده می‌شود. بخش $g(x) \leq 0$ جزء قیود کلی[۲۱] مسئله به حساب می‌آید. ترکیبات فصلی $k \in K$ که هر کدام از D_k عبارت[۲۲] تشکیل شده‌اند و به وسیله‌ی عملگر or به یک‌دیگر متصل می‌باشند. در هر عبارت خود شامل یک متغیر دودویی[۲۳] Y_{ik}، مجموعه‌ای از نامساوی‌های $r_{ik}(x) \leq 0$ و یک متغیر هزینه‌ی[۲۴] C_k می‌باشد. فلسفه‌ی هر عبارت بدین شکل است که اگر Y_{ik} برابر $True$ باشد،

[۲۰] Generalized Disjunctive Programming
[۲۱] Global Constraints
[۲۲] Term
[۲۳] Boolean Variable
[۲۴] Cost Variable

آنگاه قیود $r_{ik}(x) \leq ۰$ و $C_k = \gamma_{ik}$ باید در نظر گرفته[۲۵] و در غیر این صورت کنارگذاشته‌شوند[۲۶]. بدین ترتیب حضور متغیرهایی دودویی می‌تواند در مدل‌سازی مسائلی که شامل تصمیمات گسسته می‌باشند بسیار مفید واقع شود.

همچینن $\Omega(Y) = True$ در واقع مجموعه‌ای از گزاره‌های منطقی هستند که مقداردهی‌های متفاوت به متغیرهای دودویی Y_{ik} را بر اساس شرایط مسئله میسر می‌سازند. به طور معمول $\Omega(Y)$ به صورت فرم نرمال عطفی[۲۷] بیان می‌شوند.

دقت شود که به طور کلی در رابطه با برنامه‌ریزی فصلی عمومیت یافته و توابع f و g و r سه حالت زیر ممکن است رخ دهد:

- خطی باشند.
- غیرخطی اما محدب باشند.
- غیرمحدب باشند.

هر کدام از موارد فوق نیازمند روش‌های مختلفی برای حل می‌باشند. روش‌های ارائه شده در فصول آینده بر روی دو مورد اول تمرکز می‌کنند.

در فرمول ۲-۴ مثالی ساده از یک برنامه‌ریزی فصلی تعمیم‌یافته را می‌توان مشاهده نمود که فاقد قیود جبری می‌باشد. این مثال دو مجموعه ترکیب فصلی[۲۸] دارد که هر کدام شامل دو عبارت می‌باشند و $True$ بودن متغیرهای دودویی منتاظر مشخص می‌نماید که آیا هر عبارت در فرایند حل در نظرگرفته شود یا کنار گذاشته شود. متغیرهای دودویی نیز خود بر اساس گزاره‌های منطقی می‌توانند مقادیر مختلفی بگیرند. دقت شود که گزاره‌های منطقی در این مثال بیان می‌دارند که اگر $Y_۱ = True$ و $Y_۲ = False$ باشند، آنگاه حتما باید $Y_۳ = False$ باشد و همچنین متغیرهایی دودویی $Y_۲$ و $Y_۳$ نمی‌توانند همزمان $True$ شوند.

---

[۲۵] Enforcing
[۲۶] Ignoring
[۲۷] Conjunctive Normal Form
[۲۸] Disjunction Set

$$
\begin{array}{lll}
\text{Minimize} & c + 2x_1 + x_2 & \text{Objective Function} \\
\text{subject to} & & \\
& \begin{bmatrix} Y_1 \\ -x_1 + x_2 + 2 \le 0 \\ c \le 5 \end{bmatrix} \vee \begin{bmatrix} Y_2 \\ 2 - x_2 \le 0 \\ c \le 7 \end{bmatrix} & \text{Disjunctions} \\
& \begin{bmatrix} Y_3 \\ x_1 - x_2 \le 0 \end{bmatrix} \vee \begin{bmatrix} \neg Y_3 \\ x_1 \le 1 \end{bmatrix} & \\
& Y_1 \wedge \neg Y_2 \Rightarrow \neg Y_3 & \text{Logic Propositions} \\
& Y_2 \Rightarrow \neg Y_3 & \\
& Y_3 \Rightarrow \neg Y_2 & \\
& 0 \le x_1 \le 5 & \text{Continuous Variables} \\
& 0 \le x_2 \le 5 & \\
& c \ge 0 & \\
& Y_j \in \{\text{True, False}\},\ j = 1, 2, 3 & \text{Boolean Variables}
\end{array}
$$

فرمول ۲-۴

از این رویکرد می‌توان مثال دیگری را نیز در فرمول ۲-۵ مشاهده کرد که فاقد گزاره‌های منطقی می‌باشد. در این مثال هر مجموعه ترکیب فصلی شامل دو عبارت می‌باشد که متغیرهایی دودویی نظیر عبارات هر مجوعه نقیض یک‌دیگر هستند. لذا از هر مجموعه خودبه‌خود تنها یک عبارت انتخاب می‌شود.

در این فصل با برنامه‌ریزی فصلی و حالت تعمیم‌یافته‌ی آن یعنی برنامه‌ریزی فصلی تعمیم‌یافته آشنا شدیم و از هر کدام مثال‌هایی ساده را مشاهده کردیم. در فصل آینده مثال‌هایی کاربردی از مدل‌سازی مسائل موجود در حوزه‌های گوناگون را در قالب برنامه‌ریزی فصلی تعمیم‌یافته بررسی می‌کنیم.

$$
\begin{array}{lll}
\text{Min} & t & \text{Objective Function} \\
\text{s.t.} & t \geq x_A + 8 & \text{Algebraic Constraints} \\
 & t \geq x_B + 5 & \\
 & t \geq x_C + 6 & \\
 & \begin{bmatrix} Y_1 \\ x_A - x_C + 5 \leq 0 \end{bmatrix} \vee \begin{bmatrix} \neg Y_1 \\ x_C - x_A + 2 \leq 0 \end{bmatrix} & \text{Disjunctions} \\
 & \begin{bmatrix} Y_2 \\ x_B - x_C + 1 \leq 0 \end{bmatrix} \vee \begin{bmatrix} \neg Y_2 \\ x_C - x_B + 6 \leq 0 \end{bmatrix} & \\
 & \begin{bmatrix} Y_3 \\ x_A - x_B + 5 \leq 0 \end{bmatrix} \vee \begin{bmatrix} \neg Y_3 \\ x_B - x_A \leq 0 \end{bmatrix} & \\
 & t, x_A, x_B, x_C \geq 0 & \text{Continuous Variables} \\
 & Y_j \in \{\text{True, False}\},\ j = 1, 2, 3 & \text{Boolean Variables}
\end{array}
$$

فرمول۲-۵

فصل سوم: مدل‌سازی در قالب برنامه‌ریزی فصلی

در فصل قبل با مفهوم برنامه‌ریزی فصلی و برنامه‌ریزی فصلی تعمیم‌یافته آشنا شدیم و مثال‌های ساده را از هر کدام بررسی کردیم. در این فصل قصد داریم تا با بررسی تعدادی از مقالات معتبر در این حوزه، مثال‌هایی کاربردی در باب بهره‌گیری از رویکرد برنامه‌ریزی فصلی در حوزه‌های مختلف مطرح نماییم.

به طور کلی رویکرد برنامه‌ریزی فصلی کاربردهایی فراوانی در مسائل ابعاد باز[۲۹] ، مدارهای الکتریکی و الکترونیکی، برنامه‌ریزی و زمان‌بندی کارها[۳۰]، مدیریت ترافیک و مسیریابی و ... دارد[۷] . باید دقت داشت که هسته‌ی اصلی برنامه‌ریزی فصلی عمومیت‌یافته یعنی مجموعه ترکیبات فصلی و متغیرهایی دودویی نظیر آن‌ها، به عنوان ابزاری قدرتمند به ما امکان مدل‌سازی تصمیم‌گیری‌های گسسته را می‌دهد.

برای بررسی مثال‌های مطرح شده در این فصل لازم است ابتدا با مفهوم مدل‌سازی[۳۱] و کاهش[۳۲] به صورت مختصر آشنا شویم.

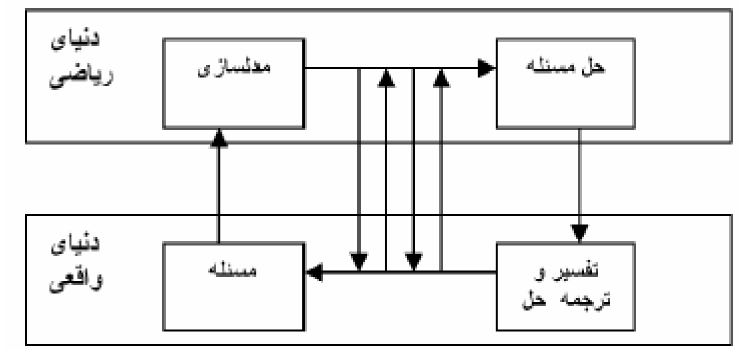
مدل‌سازی و کاهش

مدل‌سازی ریاضی عبارت است از توصیف یک سامانه یا سیستم به کمک زبان و نمادهای ریاضی. مدل‌سازی ریاضی نه تنها در علوم کامپیوتر و سایر علوم کاربردی مانند فیزیک، زیست‌شناسی، زمین‌شناسی، هواشناسی و علوم مهندسی ، هوش

---

[۲۹] Open Dimension Problems
[۳۰] Task Scheduling
[۳۱] Modeling
[۳۲] Reduction

مصنوعی و غیره کاربرد دارد بلکه در علوم اجتماعی مانند علم اقتصاد، روان‌شناسی، جامعه‌شناسی نیز مورد استفاده‌ی گسترده قرار می‌گیرد. در واقع در فرایند مدل‌سازی ما سعی می‌کنیم با فهم شرایط و ویژگی‌های مسئله، آن را در قالب یک مسئله‌ی ریاضی و با کمک متغیرهای پیوسته و گسسته تفسیر کرده و به حل آن بپردازیم. این فرایند را می‌توان به خوبی در تصویر ۳-۱ مشاهده کرد.



تصویر ۳-۱

نموه‌ای خاص از مدل‌سازی که ما در این گزارش به آن توجه ویژه داریم، طراحی یک نمونه‌ی بهینه‌سازی متناظر با مسئله‌ی اولیه به منظور بهره‌گیری از روش‌های تحلیل و حل مسائل بهینه‌سازی می‌باشد. بنابراین ما در این فصل با مسائلی روبه‌رو هستیم که از دنیای واقعی نشأت می‌گیرند و سعی داریم آن‌ها را در قالب یک مسئله‌ی برنامه‌ریزی فصلی تعمیم‌یافته مدل‌سازی کنیم.

مفهوم دیگری که در این فصل با آن سر و کار داریم بحث کاهش ریاضی می‌باشد. در نظریه محاسبه و نظریه پیچیدگی رایانشی، *کاهش* فرایندی است که نمونه‌ای کلی از یک مسأله را به نمونه‌ای از مسأله‌ای دیگر تبدیل می‌کند.

در فرم ریاضی زمانی که مسأله‌ی A را به مسأله‌ی B کاهش می‌دهیم، می‌نویسم $A \leq B$ و این بدان معناست که تابعی یک به یک مانند $f: L_A \to L_B$ چنان وجود دارد وجود دارد که می‌تواند هر نمونه‌ی $x \epsilon L_A$ را به نمونه‌ی $y \epsilon L_B$ تبدیل نماید که در این بیان L_A و L_B زبان‌های نظیر مسائل A و B می‌باشند. اگر این تبدیل در زمان چند جمله‌ای نسبت به اندازه‌ی نمونه‌ی مسئله‌ی A صورت پذیرد، آن کاهش چندجمله‌ای نامیده می‌شود. در این فصل نیز هدف و منظور ما از کاهش، کاهش چند جمله‌ای می‌باشد.

بنابراین ما به کمک مدل‌سازی و کاهش سعی بر آن داریم که یک مسئله‌ی دنیای واقعی را در قالب یک مسأله‌ی بهینه‌سازی به خصوص از نوع برنامه‌ریزی فصلی بیان نماییم و در فصول آینده به حل آن‌ها بپردازیم.

در ادامه فرایند مدل‌سازی سه مسأله‌ی زیر را بررسی می‌نماییم:

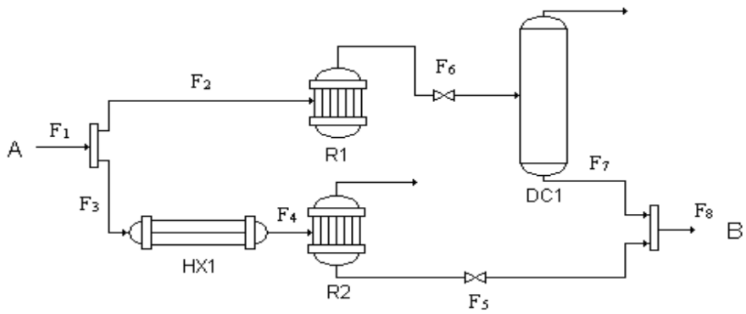
- مسئله‌ی Process Network [۷]

- مسئله‌ی Strip Packing [۸]
- مسئله‌ی مسیریابی بازوی مکانیکی در محیط سه بعدی به همراه موانع [۹]

مدل‌سازی مسئله‌ی Process Network

مسئله‌ی Process Network می‌تواند مثال خوبی برای آشنایی با فرایند مدل‌سازی یک مسئله‌ی دنیای واقعی در قالب یک نمونه‌ی برنامه‌ریزی فصلی عمومیت‌یافته باشد. در این مسئله مواد خام اولیه وارد یک شبکه می‌شوند و پس از عبور از سیستم‌های گوناگون، محصول مورد نظر به دست می‌آید.

نمونه‌ای از این مسئله را می‌توان در شکل ۳-۲ مشاهده کرد. در این مثال ماده‌ی خام A پس از ورود به شبکه، انتخاب مسیرهای مختلف و قرار گرفتن در معرض سیستم‌های گوناگون، در نهایت به محصول یعنی B تبدیل می‌شود.



تصویر ۳-۲

متغیرهای پیوسته‌ی F نشان‌دهنده‌ی جریان مواد گذرنده از بخش‌های مختلف شبکه می‌باشند. در این مثال مسئله‌ی ما مشخص کردن میزان محصول برای تولید یعنی F_8 با قیمت فروش P_1، میزان مواد خام اولیه یعنی F_1 با هزینه‌ی خرید P_2 و انتخاب مجموعه‌ای از واحدها یا سیستم‌ها برای استفاده (یعنی HX۱, R۱, R۲, DC۱) با هزینه‌ی c_k برای $k \in \{HX1, R1, R2, DC1\}$ می‌باشد به گونه‌ای که سود بیشینه شود.

مدل برنامه‌ریزی فصلی تعمیم‌یافته برای این مثال می‌تواند در قالب فرمول ۳-۱ مطرح شود.

$$Max\, Z = P_1 F_8 - P_2 F_1 - \sum_{k \in K} c_k$$

$$\begin{aligned}
&s.t. \\
&F_1 = F_3 + F_2 && (1) \\
&F_8 = F_7 + F_5 && (2)
\end{aligned}$$

$$\begin{bmatrix} Y_{HX1} \\ F_4 = F_3 \\ c_{HX1} = \gamma_{HX1} \end{bmatrix} \vee \begin{bmatrix} \neg Y_{HX1} \\ F_4 = F_3 = 0 \\ c_{HX1} = 0 \end{bmatrix} \qquad (3)$$

$$\begin{bmatrix} Y_{R2} \\ F_5 = \beta_1 F_4 \\ c_{R2} = \gamma_{R2} \end{bmatrix} \vee \begin{bmatrix} \neg Y_{R2} \\ F_5 = F_4 = 0 \\ c_{R2} = 0 \end{bmatrix} \qquad (4)$$

$$\begin{bmatrix} Y_{R1} \\ F_6 = \beta_2 F_2 \\ c_{R1} = \gamma_{R1} \end{bmatrix} \vee \begin{bmatrix} \neg Y_{R1} \\ F_6 = F_2 = 0 \\ c_{R1} = 0 \end{bmatrix} \qquad (5)$$

$$\begin{bmatrix} Y_{DC1} \\ F_7 = \beta_3 F_6 \\ c_{DC1} = \gamma_{DC1} \end{bmatrix} \vee \begin{bmatrix} \neg Y_{DC1} \\ F_7 = F_6 = 0 \\ c_{DC1} = 0 \end{bmatrix} \qquad (6)$$

$$\begin{aligned}
&Y_{R2} \Leftrightarrow Y_{HX1} && (7) \\
&Y_{R1} \Leftrightarrow Y_{DC1} && (8)
\end{aligned}$$

$$\begin{aligned}
&F_i \in R,\, c_k \in R^1,\, Y_k \in \{True, False\} \quad i \in \{1,2,3,4,5,6,7,8\} \\
&k \in \{HX1, R1, R2, DC1\}
\end{aligned}$$

فرمول ۳-۱

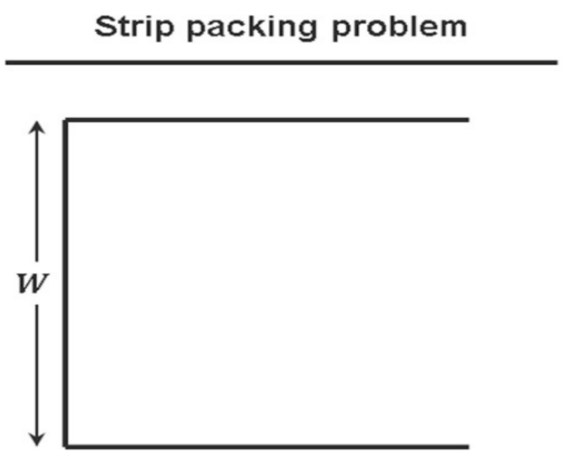
همان‌گونه که مشخص است تابع هدف به گونه‌ای طراحی شده است که با توجه به توضیحات قبلی، سود را بیشینه و هزینه‌ها را کمینه کند. قیود ۱ و ۲ تضمین‌کننده‌ی موازنه‌ی مواد خروجی و مواد ورودی در گره‌های شبکه (سیستم‌ها) می‌باشند. قیود ۳ و ۴ و ۵ و ۶ بیان‌گر تصمیمات گسسته‌ی ما برای انتخاب یا عدم انتخاب سیستم‌های گوناگون می‌باشند و در نهایت قیود ۷ و ۸ مسئله را مجبور می‌سازند که سیستم DC۱ انتخاب شود اگر و تنها اگر سیستم R۱ انتخاب شود و سیستم HX۱ انتخاب شود اگر و تنها اگر سیستم R۲ انتخاب شود که رعایت این قیود با توجه به شکل ۳-۲ و توپولوژی شبکه الزامی است. این فرمول‌بندی در واقع یک برنامه‌ریزی فصلی تعمیم‌یافته با تمرکز بر روی تصمیم‌گیری‌های گسسته برای مسئله‌ی Process Network *می‌باشد*.

لذا در این بخش ما نشان دادیم که چگونه می‌توان با بهره‌گیری از برنامه‌ریزی فصلی عمومیت‌یافته، مسئله‌ی Process Network را به صورت یک مسئله‌ی بهینه‌سازی مدل کرد. در بخش بعدی به فرایند مشابهی می‌پردازیم که سعی بر آن دارد با درک مسئله‌ی Strip Packing، به مدل‌سازی آن در قالب مورد بحث احتمام ورزد.

مدل‌سازی مسئله‌ی Strip Packing

در این بخش قصد داریم با استفاده از رویکرد برنامه‌ریزی فصلی تعمیم‌یافته یک فرمول‌بندی برای مسئله‌ی Strip Packing ارائه دهیم. در فصول آینده نیز با تبدیل این مسئله‌ی برنامه‌ریزی فصلی به یک مسئله‌ی برنامه‌ریزی خطی صحیح ترکیبی[۳۳]، اقدام به حل آن می‌نماییم.

به طور کلی Strip Packing یک نمونه‌ی خاص از دسته‌ی مسائل برش و بسته‌بندی[۳۴] می‌باشد که کاربردهای فراوانی دارد. به طور دقیق‌تر در حالت دو بعدی این مسئله، ما به دنبال قراردادن تعدادی مستطیل با طول و عرض مشخص در یک نوار با عرض مشخص هستیم به گونه‌ای که طول نوار کمینه شود. شکل ۳-۳ یک نوار با عرض ثابت W را نمایش می‌دهد.



تصویر ۳-۳

در این بخش ما تنها حالت دوبعدی مسألهٔ Strip Packing را مورد بررسی قرار می‌دهیم اما رویکردی که در ادامه ذکر می‌شود به راحتی قابلیت تعمیم به ابعاد بالاتر را دارد. به بیان دیگر حالت عمومی این رویکرد می‌تواند برای تمام مسائل ابعاد باز و دسته‌بندی و برش مورد استفاده قرار بگیرد.

همان‌گونه که اشاره شد در حالت کلی در مسئله‌ی Strip Packing تعداد N مستطیل داریم و می‌خواهیم آن‌ها در یک نوار با عرض ثابت W به گونه‌ای قرار دهیم که طول نوار کمینه باشد. طول و ارتفاع مستطیل‌ها را به ترتیب با L_i و H_i و $1 \leq i \leq N$ نمایش می‌دهیم.

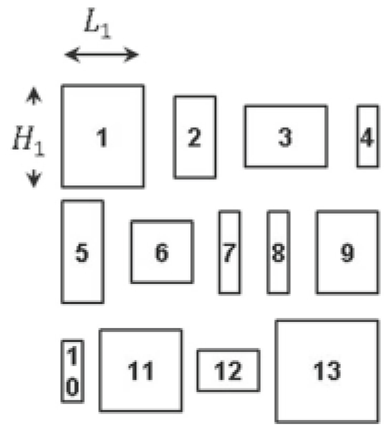
دقت شود که در حالت مورد بحث ما مستطیل‌ها نمی‌توانند چرخانده شوند و لذا ارتفاع مستطیل‌ها عرض نوار و طول مستطیل‌ها نیز طول نوار را پوشش می‌دهند.

---

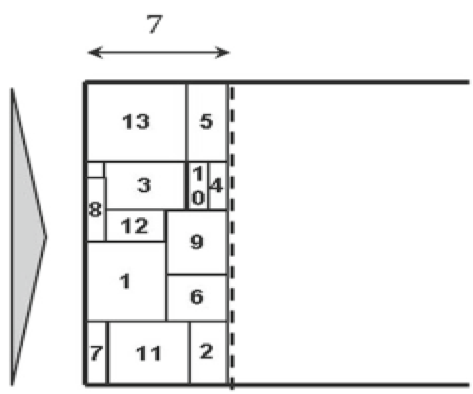
۳۳ Mixed-Integer Linear Programming

۳۴ Cutting and Packing Problems

در شکل ۳-۴ نمونه‌ای از این مسئله متشکل از ۱۳ مستطیل قابل مشاهده است. همچنین تصویر ۳-۵ نیز حالتی بهینه برای قرارگیری این مستطیل‌ها در نواری با عرض ثابت را نشان می‌دهد.



تصویر ۳-۴

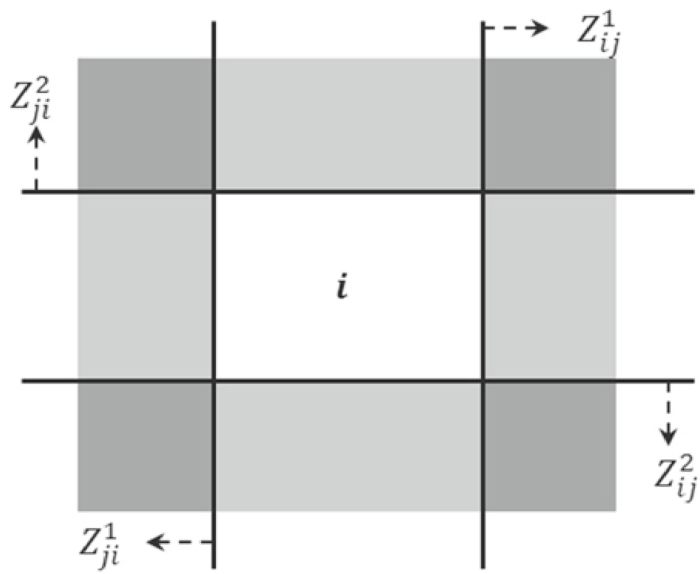


تصویر ۳-۵

برای این مدل‌سازی نیاز است ابتدا تعدادی متغیر کمکی تعریف نماییم. برای هر مستطیل متغیرهای پیوسته‌ی $(x_i\,, y_i)$ را برابر گوشه‌ی بالا سمت چپ آن مستطیل هنگام قرارگیری در نوار در نظر می‌گیریم. هدف ما کمینه کردن طول نوار یا lt می‌باشد.

همچنین برای مدل‌سازی و حل این مسئله نیاز داریم که وضعیت یک مستطیل را نسبت به مستطیلی دیگر بسنجیم. برای این هدف متغیر دودیی $Z_{ij}^{۱}$ را بدین صورت تعریف می‌نماییم که مقدار آن برابر $True$ است اگر و تنها اگر مستطیل iام در سمت چپ مستطیل jام قرار بگیرد. به همین ترتیب متغیر دودیی $Z_{ij}^{۲}$ را در نظر میگیریم به شکلی که مقدار آن برابر $True$ است اگر و تنها اگر مستطیل iام در بالای مستطیل jام قرار بگیرد. دقت شود که مدل برنامه‌ریزی فصلی ما

باید به گونه‌ای باشد که از هم‌پوشانی مستطیل‌ها جلوگیری به عمل آورد. تصویر ۳-۶ نقش متغیر Z را در نحوه‌ی کنار هم قرار گرفتن دو مستطیل در نوار نسبت به هم نشان می‌دهد.



تصویر ۳-۶

با توجه به آنچه که در فصل قبل دیدیم و با استفاده از فرمول ۲-۳ می‌توانیم نمونه‌ی کلی برنامه‌ریزی فصلی تعمیم‌یافته‌ی خود را برای مسئله‌ی Strip Packing در حالت دوبعدی در قالب فرمول ۳-۲ بیان نماییم.

$$
\begin{array}{ll}
min\, lt & \\
s.t. \quad lt \geq x_i + L_i & i \in N \\
\begin{bmatrix} Z_{ij}^1 \\ x_i + L_i \leq x_j \end{bmatrix} \vee \begin{bmatrix} Z_{ji}^1 \\ x_j + L_j \leq x_i \end{bmatrix} & \\
\vee \begin{bmatrix} Z_{ij}^2 \\ y_i - H_i \geq y_j \end{bmatrix} \vee \begin{bmatrix} Z_{ji}^2 \\ y_j - H_j \geq y_i \end{bmatrix} & i, j \in N, i < j \\
Z_{ij}^1 \veebar Z_{ji}^1 \veebar Z_{ij}^2 \veebar Z_{ji}^2 & i, j \in N, i < j \\
0 \leq x_i \leq UB - L_i & i \in N \\
H_i \leq y_i \leq W & i \in N \\
Z_{ij}^1, Z_{ij}^2 \in \{True, False\} & i, j \in N, i \neq j
\end{array}
$$

فرمول ۳-۲

هدف ما کمینه کردن طول نوار یعنی مقدار متغیر lt می‌باشد. قید کلی $lt \geq x_i + L_i$ مسأله را وادار می‌نماید که طول نوار حتما بزرگتر از مختصات راست‌ترین مستطیل باشد. در این مدل‌سازی یک مجموعه ترکیب فصلی داریم که شامل چهار عبارت می‌باشد. این ترکیبات فصلی در واقع هر ۴ حالت از نحوه‌ی قرارگیری دو مستطیل در کنار یک دیگر را نشان می‌دهد.

از سوی دیگر گزاره‌ی منطقی موجود در فرمول ۳-۲ بیان می‌دارد که تنها یکی از ۴ عبارت موجود در ترکیب فصلی در نظر گرفته شود و سه مورد دیگر کنار گذاشته‌شوند. دقت شود همان طور که در فصل قبل اشاره شد، در ساختار یک برنامه‌ریزی فصلی تعمیم‌یافته زمانی که متغیر دودیی متناظر با یک عبارت مقدار $True$ بگیرد به معنای آن است که آن عبارت باید در حل مسئله لحاظ شود و در غیر این صورت کنار گذاشته شود.

در فصول آینده پس از آشنا شدن با روش‌های حل یک برنامه‌ریزی فصلی عمومیت‌یافته، این مثال خاص را بررسی کرده و در مسیر حل آن قدم برمیداریم.

در ادامه به مدل‌سازی یک مسئله‌ی پرکاربرد در دنیای رباتیک با استفاده از رویکرد برنامه‌ریزی فصلی می‌پردازیم.

مسیریابی بازوی مکانیکی در فضای سه بعدی با موانع

در این مثال قصد داریم مسئله‌ی مسیریابی بازوهای رباتی را در فضای چند بعدی و با حضور منابع مدل کنیم.

بازوی‌های مکانیکی برای انجام وظایف خود نیاز دارند یک مسیر مناسب میان مبدا و مقصد خود در فضای سه بعدی به گونه‌ای انتخاب کنند که با موانع موجود برخوردی نداشته‌باشند. در این مسئله فرض ما بر این است که محیط پویا نیست و شرایط محیطی از جمله موقعیت مکانی موانع در طول زمان تغییر نمی‌کند. همچنین فرض دیگر در نظر گرفته‌شده در این مدل‌سازی، آگاهی کامل عامل یعنی بازوی مکانیکی از تمام خصوصیات و ویژگی‌های محیط از جمله محل موانع است.

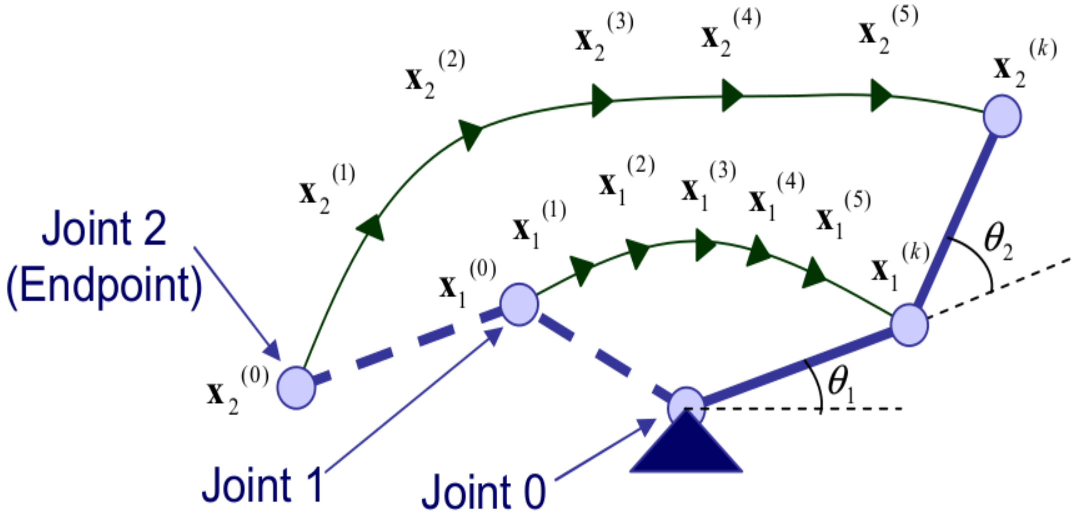
یک بازوی مکانیکی مطابق تصویر ۳-۷ از تعدادی مفصل تشکیل شده‌است. ایده‌ی اصلی برای حل این مسئله و مدل‌سازی، یافتن مسیری ممکن[۳۵] برای مفاصل است به گونه‌ای که مفصل نهایی یعنی دورترین مفصل از تکیه‌گاه بتواند بدون برخورد به موانع از مختصات مبدا به مختصات مقصد برسد.

لذا یک مسیر ممکن تلقی می‌شود اگر شرایط زیر را داشته‌باشد:

- مفاصل و رابط بین آن‌ها در طول مسیر با هیچ مانعی برخورد نکنند.
- مسیر باید از نظر فیزیک سینماتیک ممکن باشد.
- مسیر باید از نظر فیزیک داینامیک ممکن باشد.

[۳۵] Feasible Path

- مفصل انتهایی باید به واسطه‌ی این مسیر از مبدا به مقصد برسد.

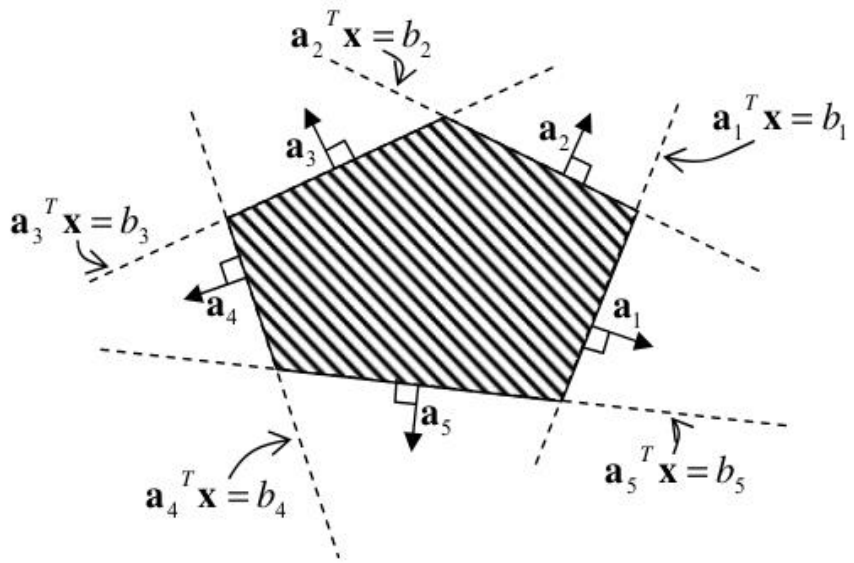


تصویر ۳-۷

مانند مثال قبل ابتدا تعدادی متغیر کمکی برای مدل‌سازی معرفی می‌نماییم. متغیر پیوسته‌ی x_i^t نشان‌دهنده‌ی موقعیت فیزیکی مفصل iام در لحظه‌ی t می‌باشد. همان‌گونه که از تعریف این متغیر مشخص است، مدل‌سازی ما از رویکردی زمان-گسسته استفاده می‌نماید و یک مسیر k مرحله‌ای برای مفصل iام مطابق شکل ۳-۷ به صورت توالی‌ای از موقعیت‌های مکانی $x_i^۱$ تا x_i^k بیان می‌شود. مشخص است که برای سادگی در این مدل‌سازی مسائلی مانند اصطکاک بازوی مکانیکی در حال حرکت با هوا و سایر موارد مشابه در نظر گرفته نمی‌شوند.

در ادامه سعی می‌کنیم با استفاده از متغیرهای تعریف شده و فرم کلی برنامه‌ریزی فصلی تعمیم‌یافته، قیودی را طراحی نماییم که ۴ شاخص ذکر شده برای یک مسیر ممکن را در نظر بگیرند.

برای جلوگیری از برخورد بازوی مکانیکی با موانع ابتدا لازم است یک مانع را به فرم ریاضی تعریف نماییم. فرض می‌کینم یک مانع محدب دو بعدی مطابق آنچه که در شکل ۳-۸ نمایش داده‌شده است، از N خط راست تشکیل باشد و معادله‌ی هر خط مطابق تصویر بیان شود. مشخص است که به راحتی می‌توان این فرمول‌بندی را برای ابعاد بالاتر نیز تعمیم داد.



تصویر ۳-۸

یک شیء در زمان t به یک مانع برخورد کرده است اگر مختصات آن در لحظه‌ی t یعنی x^t درون مانع قرار گرفته‌باشد. بنابراین برای جلوگیری از برخورد، قید ذکر شده در فرمول ۳-۳ باید برقرار باشد.

$$\bigvee_{i=1\ldots N} \mathbf{a}_i^T\mathbf{x}_t > b_i \,.$$

فرمول ۳-۳

لازم به ذکر است که برای سادگی، موانع پلی‌هدرون‌های محدب هستند.

حال اگر فرض کنیم در فضای حرکتی ما تعداد M مانع قرار دارد، قیود متناظر با عدم برخورد هر شی با موانع در لحظه‌ی t با متخصات x_t به صورت ذکر شده در فرمول ۳-۴ خواهد بود.

$$\bigwedge_{j=1,\ldots,M} \bigvee_{i=1,\ldots,N} \mathbf{a}_{ij}^T\mathbf{x}_t > b_{ij} \,.$$

فرمول ۳-۴

در واقع قیود مطرح شده در فرمول‌های ۳-۴ و ۳-۳ تضمین می‌کنند که در لحظه‌ی دلخواه t مختصات شی مورد نظر خارج از فضای داخل و روی ابرصفحه‌های[۳۶] تشکیل دهنده‌ی موانع باشد.

[۳۶] Hyperplane

با بهره‌گیری از فرمول ۳-۴ و متغیرهای کمکی متناظر با مختصات مفاصل بازوی مکانیکی که پیش از این تعریف شده‌است، می‌توان بیان داشت که مفصل lام در لحظه‌ی t با هیچ‌کدام از موانع برخوردی نخواهد داشت اگر و تنها اگر قید مطرح شده در فرمول ۳-۵ برقرار باشد.

$$\bigwedge_{j=1,\ldots,M} \bigvee_{i=1,\ldots,N} \mathbf{a}_{ij}^T \mathbf{x}_l^{(t)} > b_{ij}.$$

فرمول ۳-۵

تا این مرحله توانسته‌ایم با استفاده از ترکیب‌های فصلی قیودی طراحی نماییم که مانع از برخورد مفاصل با موانع می‌شود. موضوع مهم دیگر عدم برخورد رابط مفاصل با موانع می‌باشد. منظور از رابط مفاصل قسمتی از بازوی مکانیکی است که دو مفصل را به یک دیگر متصل می‌نماید و برای سادگی با خطی راست مدل می‌شود. بنابراین مختصات یک نقطه از رابط بین دو مفصل l و $l+۱$ در لحظه‌ی t را می‌توان به ازای $\lambda\epsilon[۰,۱]$ به صورت آنچه که در فرمول ۳-۶ نشان داده شده‌است، در نظر گرفت.

$$\mathbf{x} = \lambda \mathbf{x}_l^{(t)} + (1-\lambda)\mathbf{x}_{l+1}^{(t)}$$

فرمول ۳-۶

با انتخاب تعداد متناهی عدد از بازه‌ی $[۰,۱]$ برای λ، میتوانیم تعداد محدودی نقطه از رابط را در نظر بگیریم. در این مثال با انتخاب $\lambda = \frac{۱}{۲}$، نقطه‌ی وسط رابط را به عنوان نماینده‌ی رابط در مدل‌سازی انتخاب می‌کنیم. به طور مشابه و با استفاده از فرمول ۳-۵ به راحتی قیود مناسب برای عدم برخورد رابط‌ها یا نقاطی از آن‌ها با موانع در قالب فرمول ۳-۷ به دست می‌آید.

$$\bigwedge_{j=1,\ldots,M} \bigvee_{i=1,\ldots,N} \mathbf{a}_{ij}^T \left(\lambda \mathbf{x}_l^{(t)} + (1-\lambda)\mathbf{x}_{l+1}^{(t)}\right) > b_{ij}.$$

فرمول ۳-۷

پس از طراحی ترکیبات فصلی متناظر با محدودیت‌های مربوط به عدم برخورد بازوی مکانیکی با موانع، نوبت به بررسی فیزیک سینماتیک بازو می‌سد. در این بخش ما به طور کلی دو بحث زیر را مورد توجه قرار می‌دهیم:

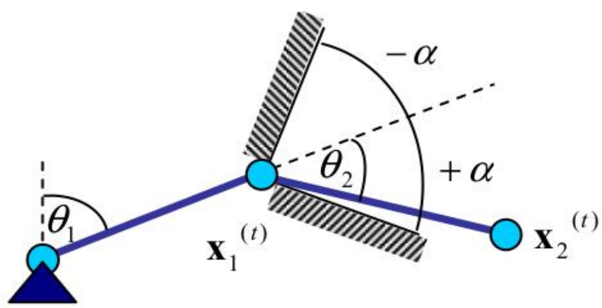
- فاصله‌ی بین دو مفصل متوالی یا همان طول رابط میان دو مفصل باید در طول فرایند حرکت ثابت بماند.
- در اکثر بازوهای مکانیکی برای حرکت مفاصل و رابط‌ها محدودیت‌هایی از نظر میزان تغییر زاویه وجود دارد. لذا لازم است در یک حالت عمومی این دسته از محدودیت‌ها نیز برای هر مفصل کنترل شوند.

فرض کنید طول رابط میان مفصل l و مفصل $l+1$ را به d_l نشان دهیم. با استفاده از خواص ضرب ماتریسی، قید مرتبط با حفظ فاصله‌ی میان دو مفصل متوالی را می‌توان به صورت فرمول ۳-۸ نشان داد.

$$(\mathbf{x}_{l+1}^{(t)} - \mathbf{x}_l^{(t)})^T(\mathbf{x}_{l+1}^{(t)} - \mathbf{x}_l^{(t)}) = d_l$$

فرمول ۳-۸

از سوی دیگر در مورد محدودیت تغییر زاویه‌ی رابط فرض کنید که مطابق شکل ۳-۹، زاویه‌ی مفصل شماره‌ی ۱ یعنی θ_2 میتواند در بازه‌ی $[-\alpha , +\alpha]$ تغییر نماید. با توجه به شکل و با استفاده از خواص ضرب داخلی دو بردار $x_2^t - x_1^t$ و $x_1^t - x_2^t$ رابطه‌ی بیان مطرح شده در فرمول ۳-۹ به دست می‌آید.



تصویر ۳-۹

$$(\mathbf{x}_2^{(t)} - \mathbf{x}_1^{(t)})^T(\mathbf{x}_1^{(t)} - \mathbf{x}_0^{(t)}) = d_1 \cdot d_0 \cdot cos\theta_2$$

فرمول ۳-۹

و لذا قید متناظر با محدودیت میزان تغییر زاویه می‌تواند به صورت ذکر شده در فرمول ۳-۱۰ در نظر گرفته‌شود.

$$(\mathbf{x}_2^{(t)} - \mathbf{x}_1^{(t)})^T(\mathbf{x}_1^{(t)} - \mathbf{x}_0^{(t)}) \geq d_1 \cdot d_0 \cdot cos\alpha$$

فرمول ۳-۱۰

لازم به ذکر است که این قیود خطی نمی‌باشند و از نوع چندجمله‌ای هستند و لذا برنامه‌ریزی فصلی به دست آمده نیز از از نوع چندجمله‌ای خواهد بود.

در رابطه با محدودیت‌های مرتبط با فیزیک دینامیک مسئله، به دلیل پرهیز از پیچیدگی ما تنها محدودیت سرعت را برای حرکت مفاصل در نظر می‌گیریم که قید متناظر با آن را می‌توان به راحتی مطابق فرمول ۳-۱۱ تعریف نمود.

$$\frac{\mathbf{x}_l^{(t+1)} - \mathbf{x}_l^{(t)}}{\Delta t} \leq V_{max},$$

فرمول ۳-۱۱

در نهایت با کنار هم قرار دادن قیود مطرح شده، می‌توانیم حالتی ساده شده از مسئله‌ی یافتن مسیر ممکن برای حرکت بازوی مکانیکی در یک فضای سه بعدی با حضور موانع را به یک مسئله‌ی برنامه‌ریزی فصلی چندجمله‌ای مدل کنیم.

لازم است دقت شود که تمام قیود مطرح شده در قسمت‌های قبل در راستای اطمینان از ممکن بودن مسیر به دست‌آمده طراحی شدند. در جهت ارتقای مسیر و یافتن مسیر بهینه نیز می‌توان با استفاده از ترکیبات فصلی قیودی را طراحی کرد که این موضوع از حوصله‌ی این گزارش خارج است.

در این فصل ابتدا با مفهوم مدل‌سازی و کاهش آشنا شدیم و سپس مثال‌هایی پرکاربرد از مدل‌سازی تعدادی مسئله‌ی معروف در قالب برنامه‌ریزی فصلی را دیدیم. همان‌گونه که مطرح شد این نوع مدل‌سازی در حوزه‌های مختلف از کامپیوتر و برق گرفته تا شیمی و مدیریت و عمران کاربردهای فراوانی دارد که مثال دیگری از آن در [۱۰] قابل مشاهده می‌باشد.

در فصول آینده روش‌های حل برنامه‌ریزی فصلی تعمیم‌یافته را بررسی کرده و مثال‌های مطرح شده در قسمت‌های قبل را با استفاده از آن روش‌ها حل می‌نماییم. سپس با دو ابزار معروف برای برنامه‌نویسی و حل مسائل برنامه‌ریزی فصلی آشنا می‌شویم.

فصل چهارم: حل برنامه‌ریزی فصلی

در دو فصل گذشته با ساختار برنامه‌ریزی فصلی و حالت تعمیم یافته‌ی آن یعنی برنامه‌ریزی فصلی تعمیم‌یافته آشنا شدیم و در ادامه مثال‌هایی از مدل‌سازی مسائل معروف حوزه‌های مختلف در قالب این رویکرد بهینه‌سازی را دیدیم.

در این فصل می‌خواهیم با روش‌های حل یک مسئله‌ی برنامه‌ریزی فصلی آشنا شویم. به طور کلی برای حل این دسته از مسائل بهینه‌سازی دو رویکرد به صورت عمده مورد استفاده قرار می‌گیرد:

- تبدیل یک مسئله‌ی برنامه‌ریزی فصلی تعمیم‌یافته به مسئله‌ی برنامه‌ریزی صحیح ترکیبی[۳۷] و سپس حل آن
- حل مستقیم مسئله‌ی برنامه‌ریزی فصلی تعمیم‌یافته

در اکثر مواقع روش اول به دلیل برتری در پیچیدگی زمانی لازم برای حل و وجود حل‌کننده‌های[۳۸] معروف برای آن، بیشتر از روش دوم مورد توجه قرار می‌گیرد. در این پروژه نیز از همان ابتدا تمرکز بیشتر بر روش اول قرار معطوف و دو رویکرد برای استفاده از این روش بررسی شد. لازم به ذکر است که روش دوم می‌تواند به عنوان کارهای آینده برای این پروژه در نظر گرفته شود. علاقه‌مندان به مطالعه‌ی بیشتر در خصوص روش دوم می‌توانند به [٦] مراجعه نمایند.

همان‌گونه که از فصول پیشین به یاد داریم، هسته‌ی اصلی یک برنامه‌ریزی فصلی تعمیم‌یافته، مجموعه‌ی ترکیبات فصلی آن می‌باشد که برای پیاده‌سازی تصمیم‌گیری‌های گسسته مورد استفاده قرار می‌گیرد. لذا برای تبدیل یک نمونه از

---

[۳۷] Mixed Integer Programming

[۳۸] Solver

مسئله‌ی برنامه‌ریزی فصلی عمومیت یافته به نمونه‌ای از مسئله‌ی برنامه‌ریزی صحیح ترکیبی باید این مجموعه از ترکیبات فصلی را به قیود و ترکیبات عطفی تبدیل نماییم.

برای این تبدیل دو روش مرسوم موجود است:

- روش Big-M [۷]
- روش پوش محدب[۳۹] [۷] و [۱۱]

در ادامه‌ی این فصل ضمن معرفی این دو روش، آن‌ها را با هم مقایسه کرده و مثال‌های مطرح شده در فصول قبل را با این دو روش به مسائل برنامه‌ریزی صحیح ترکیبی تبدیل می‌نماییم.

جهت یادآوری لازم به ذکر است که فرم کلی یک مسئله‌ی کمینه‌سازی در قالب یک نمونه‌ی برنامه‌ریزی فصلی تعمیم‌یافته را مطابق آنچه که در فصول پیشین دیدیم میتوانیم در قالب فرمول ۴-۱ بیان کنیم.

$$
\begin{aligned}
&\min z = f(x) \\
&g(x) \le 0 \\
&\vee_{i \in D_k} \begin{bmatrix} Y_{ki} \\ r_{ki}(x) \le 0 \end{bmatrix} \quad k \in K \\
&\underline{\vee}_{\ i \in D_k} Y_{ki} \qquad\qquad k \in K \\
&\Omega(Y) = \text{True} \\
&x^{lo} \le x \le x^{up} \\
&x \in \mathbb{R}^n \\
&Y_{ki} \in \{\text{True}, \text{False}\} \quad k \in K, i \in D_k
\end{aligned}
$$

فرمول ۴-۱

توضیحات مربوط به متغیرهای موجود در فرمول فوق و تفسیر نقش آن‌ها، در فصل دوم به تفصیل بیان و بررسی شده‌است.

همچنین به طور کلی یک نمونه از مسئله‌ی برنامه‌ریزی خطی صحیح ترکیبی را می‌توان به صورت آنچه که در فرمول ۴-۲ نشان‌داده‌شده است، بیان کرد.

---

[۳۹] Convex Hull

$$\min z = f(x, y)$$
$$g(x, y) \le 0$$
$$x \in X$$
$$y \in Y$$

فرمول ۴-۲

که در این فرم، $f: R^n \to R^1$ و $g: R^n \to R^m$ دو تابع پیوسته و x در بردارنده‌ی متغیرهای پیوسته و y در بردارنده‌ی متغیرهای گسسته می‌باشند. در فرم فوق قیود خطی در قالب در تابع g بیان شده‌اند.

روش Big-M

در روش Big-M سعی می‌کنیم با استفاده از یک مقدار عددی ثابت بزرگ که آن را M می‌نامیم، مجموعه ترکیبات فصلی موجود در فرمول ۴-۱ را از فرم فصلی خارج کنیم. در واقع می‌خواهیم هر عبارت در مجموعه ترکیبات فصلی یعنی متغیر فعال‌کننده و قید نامساوی را به صورت یک عبارت جبری ریاضی بنویسم.

بنابراین یک عبارت دلخواه از مجموعه‌ی ترکیبات فصلی مانند آنچه در فرمول ۴-۳ دیده می‌شود را در نظر بگیرید.

$$\begin{bmatrix} Y_{\mathrm{ki}} \\ r_{\mathrm{ki}}(x) \le 0 \end{bmatrix}$$

فرمول ۴-۳

با استفاده از مقدار به اندازه‌ی کافی بزرگ M می‌توانیم عبارت فوق را به صورت یک عبارت جبری ریاضی در قالب فرمول ۴-۴ بنویسیم.

$$r_{\mathrm{ki}}(x) \le M^{\mathrm{ki}}(1 - y_{\mathrm{ki}})$$

فرمول ۴-۴

نتیجه‌ی به دست‌آمده در فرمول ۴-۴ را بدین شکل می‌توان تفسیر کرد که اگر مقدار y_{ki} برابر با $False$ یا صفر باشد، آنگاه در مسئله‌ی برنامه‌ریزی فصلی عمومیت‌یافته، این ترکیب نباید در نظر گرفته‌شود حال آن‌که مطابق فرمول ۴-۴ با صفر شدن مقدار y_{ki}، تابع r_{ki} با توجه به بزرگ بودن M آزاد می‌شود و محدودیتی روی آن وجود ندارد که این موضوع به نوعی با در نظر نگرفتن قید متناظر آن در برنامه‌ریزی فصلی معادل است.

از سوی دیگر اگر مقدار y_{ki} برابر با $True$ یا یک باشد، آنگاه در مسئله‌ی برنامه‌ریزی فصلی عمومیت‌یافته، این ترکیب و لذا قید متناظر با آن باید در نظر گرفته‌شود حال آنکه در مطابق فرمول ۴-۴ همین اتفاق رخ می‌دهد.

بنابراین فرم کلی مسئله‌ی برنامه‌ریزی صحیح ترکیبی به دست‌آمده را می‌توان در فرمول ۴-۵ مشاهده نمود.

$$
\begin{array}{lll}
 & \min z = f(x) & \\
\text{s.t.} & g(x) \le 0 & \\
 & r_{\mathrm{ki}}(x) \le M^{\mathrm{ki}}(1-\gamma_{\mathrm{ki}}) & k \in K, i \in D_{\mathrm{k}} \\
 & \sum_{\mathrm{i} \in \mathrm{D_k}} \gamma_{\mathrm{ki}} = 1 & k \in K \\
 & Hx \ge h & \\
 & \mathrm{x}^{\mathrm{lo}} \le x \le x^{\mathrm{up}} & \\
 & x \in \mathbb{R}^{\mathrm{n}} & \\
 & \gamma_{\mathrm{ki}} \in \{0,1\} & k \in K, i \in D_{\mathrm{k}}
\end{array}
$$

فرمول ۴-۵

در نهایت با تبدیل مسئله‌ی برنامه‌ریزی فصلی عمومیت‌یافته به مسئله‌ی برنامه‌ریزی صحیح ترکیبی و حل آن مطابق روش‌های مطرح شده در فصل اول می‌توان مسئله‌ی اولیه را حل و مقادیر متغیرهای دخیل در مسئله را به دست آورد.

به عنوان مثال، مسئله‌ی Strip Packing را که در فصل سوم مطرح و مطابق فرمول ۳-۲ در قالب یک مسئله‌ی برنامه‌ریزی تعمیم‌یافته مدل‌سازی شد، در نظر بگیرید. حال با بهره‌گیری از روش Big-M می‌توانیم مطابق فرمول ۴-۶ مدل پیشین را به یک نمونه از مسئله‌ی برنامه‌ریزی صحیح ترکیبی تبدیل نماییم که در آن مقدار ثابت و بزرگ UB نقش همان M را دارد.

$$
\begin{array}{ll}
\min lt & \\
s.t. \quad lt \ge x_i + L_i & i \in N \\
x_i + L_i \le x_j + UB(1 - z_{ij}^1) & i, j \in N, i < j \\
x_j + L_j \le x_i + UB(1 - z_{ji}^1) & i, j \in N, i < j \\
y_i - H_i \ge y_j - W(1 - z_{ij}^2) & i, j \in N, i < j \\
y_j - H_j \ge y_i - W(1 - z_{ji}^2) & i, j \in N, i < j \\
z_{ij}^1 + z_{ji}^1 + z_{ij}^2 + z_{ji}^2 = 1 & i, j \in N, i < j \\
0 \le x_i \le UB - L_i & i \in N
\end{array}
$$

فرمول ۴-۶

روش پوش محدب

در فرمول ۴-۷ فرم تغییر یافته‌ی فرمول ۴-۱ با استفاده از روش پوش محدب نمایش داده‌شده است.

$$
\begin{aligned}
& \min\ z = f(x) \\
\text{s.t.} \quad & g(x) \le 0 \\
& x = \sum_{i \in D_k} \nu^{ki} && k \in K \\
& \gamma_{ki} r_{ki}\left(\nu^{ki}/\gamma_{ki}\right) \le 0 && k \in K, i \in D_k \\
& \sum_{i \in D_k} \gamma_{ki} = 1 && k \in K \\
& Hx \ge h \\
& x^{lo}\gamma_{ki} \le \nu^{ki} \le x^{up}\gamma_{ki} && k \in K, i \in D_k \\
& x^{lo} \le x \le x^{up} \\
& x \in \mathbb{R}^n \\
& \gamma_{ki} \in \{0, 1\} && k \in K, i \in D_k
\end{aligned}
$$

فرمول ۴-۷

در این رویکرد برای هر مجموعه‌ی فصلی $k\epsilon K$ و هر تک عبارت فصلی $i\epsilon D_k$، متغیرهای پیوسته‌ی x به متغیرهای v^{ki} تفکیک[40] می‌شوند. قیود $x^{lo}Y_{ki} \le\ v^{ki} \le x^{up}Y_{ki}$ را می‌توان اینگونه تفسیر کرد که اگر یک عبارت در ترکیبات فصلی فعال[41] باشد ($Y_{ki} =$ ۱)، متغیر تفکیک شده‌ی متناظر با آن دارای کران‌های بالا و پایین متعبر و همانند x باشد. از سوی دیگر اگر آن عبارت غیرفعال باشد ($Y_{ki} =$ ۰)، مقدار متغیر تکفیک شده‌ی متناظر صفر خواهد بود.

قید $x = \sum_{i\epsilon D_k} v^{ki}$ بیان می‌دارد که همواره مقدار متغیر اصلی برابر مقدار متغیرهای تفکیک‌شده‌ی نظیر عبارات فعال در ترکیبات فصلی باشد. نامساوی موجود در عبارات ترکیبات فصلی نیز به وسیله‌ی تابع $Y_{ki}r_{ki}(v^{ki}/Y_{ki})$ بیان می‌شوند بدین صورت که اگر عبارتی فعال باشد، نامساوی متناظر با آن در نظر گرفته می‌شود و در غیر این صورت با یک نامساوی بدیهی روبه‌رو هستیم که به معنای عدم در نظر گرفتن نامساوی اصلی می‌باشد.

سرانجام با تبدیل مسئله‌ی برنامه‌ریزی فصلی تعمیم‌یافته به مسئله‌ی برنامه‌ریزی صحیح ترکیبی و حل آن مطابق روش‌های مطرح شده در فصل اول می‌توان مسئله‌ی اولیه را حل و مقادیر متغیرهای دخیل در مسئله را به دست آورد.

---

[40] disaggregation
[41] active

به عنوان مثال به یاد داریم که در فصل چهارم مسئله‌ی Strip Packing مطرح شد و در قالب یک نمونه‌ی برنامه‌ریزی فصلی تعمیم‌یافته بیان شد. در فرمول ۴-۸ با استفاده از رویکرد پوش محدب، مسئله‌ی اولیه در به فرم یک مسئله‌ی برنامه‌ریزی خطی صحیح ترکیبی در آمده است.

$$
\begin{array}{ll}
\min lt & \\
s.t. \quad lt \geq x_i + L_i & i \in N \\
x_i = \nu_i^{ij} + \nu_i^{ji} & i, j \in N, i \neq j \\
y_i = \mu_i^{ij} + \mu_i^{ji} & i, j \in N, i \neq j \\
\nu_i^{ij} + L_i z_{ij}^1 \leq \nu_j^{ij} & i, j \in N, i < j \\
\nu_j^{ji} + L_j z_{ji}^1 \leq \nu_i^{ji} & i, j \in N, i < j \\
\mu_i^{ij} - H_i z_{ij}^2 \geq \mu_j^{ij} & i, j \in N, i < j \\
\mu_j^{ji} - H_j z_{ji}^2 \geq \mu_i^{ji} & i, j \in N, i < j \\
z_{ij}^1 + z_{ji}^1 + z_{ij}^2 + z_{ji}^2 = 1 & i, j \in N, i < j \\
0 \leq \nu_i^{ij} \leq (UB - L_i) z_{ij}^1 & i, j \in N, i \neq j \\
0 \leq \nu_i^{ji} \leq (UB - L_i) z_{ji}^1 & i, j \in N, i \neq j \\
H_i z_{ij}^2 \leq \mu_i^{ij} \leq (W) z_{ij}^2 & i \in N, i \neq j \\
H_i z_{ji}^2 \leq \mu_i^{ji} \leq (W) z_{ji}^2 & i \in N, i \neq j \\
0 \leq x_i \leq UB - L_i & i \in N \\
H_i \leq y_i \leq W & i \in N \\
z_{ij}^1, z_{ij}^2 \in \{0, 1\} & i, j \in N, i \neq j
\end{array}
$$

فرمول ۴-۸

مقایسه‌ی روش Big-M و روش پوش محدب

تا کنون با دو روش برای تبدیل یک نمونه از مسئله‌ی برنامه‌ریزی فصلی تعمیم‌یافته به نمونه‌ای از برنامه‌ریزی خطی صحیح ترکیبی آشنا شده‌ایم. تجارب گوناگون نشان داده‌است که در روش Big-M نسبت به روش پوش محدب، فرایند تبدیل ساده‌تر و با پیچیدگی زمانی کمتری انجام می‌شود و مدل نهایی کوچک‌تر و دارای متغیرهای کمتر است.

اما همان‌طور که مشخص است در نظر گرفتن مقدار بزرگی برای M و به طور کلی استفاده از این رویکرد در نهایت قیودی را تولید می‌کند که ممکن است به اندازه‌ی کافی محکم[42] نباشند. به بیان دیگر با وجود آنکه روش پوش محدب در پایان کار مدل بزرگ‌تر و با تعداد متغیرهای بیشتری تولید می‌کند، اما قیود حاصل از آن محکم‌تر می‌باشند.

---

[42] Tight

در خور توجه است که برای فایق آمدن بر این نقطه‌ی ضعف روش Big-M، سعی می‌شود مقدار M به گونه‌ای انتخاب شود که قیود حاصل محکم‌تر باشند. یکی از روش‌هایی که رویکرد فوق را در پیش می‌گیرد، Improved Big-M نام دارد که بررسی آن از حوصله‌ی این گزارش خارج است اما به خواننده توصیه می‌شود جهت مطالعه‌ی بیشتر، توضیحات مطرح شده در [۷] مورد بررسی قرار گیرند.

در این فصل با دو روش تبدیل یک نمونه از برنامه‌ریزی فصلی تعمیم‌یافته به نمونه‌ای از برنامه‌ریزی خطی صحیح ترکیبی آشنا شدیم و آن‌ها را با هم مقایسه کردیم. در فصل بعدی هدف ما آشنایی با ابزارهای برنامه‌نویسی برای پیاده‌سازی و حل مسائل برنامه‌ریزی فصلی می‌باشد.

فصل پنجم: ابزارهای پیاده‌سازی و حل برنامه‌ریزی فصلی

در فصول قبلی با ساختار برنامه‌ریزی فصلی و برنامه‌ریزی فصلی تعمیم‌یافته آشنا شدیم و سپس دو روش تبدیل یک نمونه از این مسئله را به نمونه‌ای از مسئله‌ی برنامه‌ریزی صحیح ترکیبی مشاهده کردیم. در این بخش می‌خواهیم با ابزارهای برنامه‌نویسی آشنا شویم که ما را در پیاده‌سازی و حل مسائل برنامه‌ریزی فصلی تعمیم‌یافته یاری می‌رسانند.

هرچند همواره باید این نکته را مد نظر داشت که ما می‌توانیم با استفاده از روش‌های بیان شده در فصل چهارم، ابتدا مسئله‌ی برنامه‌ریزی فصلی تعمیم‌یافته‌ی خود را به یک مسئله‌ی برنامه‌ریزی خطی صحیح ترکیبی تبدیل نماییم و سپس از حل‌کننده‌های موجود و مرسوم برای حل آن استفاده نماییم.

دو مورد از مهم‌ترین و مرسوم‌ترین ابزارهای پیاده‌سازی مسئله‌ی برنامه‌ریزی فصلی تعمیم‌یافته عبارتند از :

- ابزار EMP GAMS
- کتاب‌خانه‌ی Pyomo به زبان پایتون

که در ادامه به بررسی آن‌ها می‌پردازیم.

ابزار EMP GAMS

برای آشنایی با ساختار نحوی و دستور زبان این ابزار، سعی می‌کنیم یک نمونه از مسئله‌ی برنامه‌ریزی فصلی تعمیم‌یافته در فصول پیشین را در نظر گرفته و آن را در فرم مناسب برای این ابزار پیاده‌سازی کرده و حل نماییم.

بدین ترتیب فرمول ۵-۱ در واقع یک نمونه‌ی ساده از مسئله‌ی برنامه‌ریزی فصلی تعمیم‌یافته است که مشابه آن را در فصل دوم دیده‌ایم.

$$
\begin{aligned}
&\text{Minimize} \quad c + 2x_1 + x_2 \\
&\text{subject to} \\
&\qquad \begin{bmatrix} Y_1 \\ -x_1 + x_2 + 2 \le 0 \\ c \le 5 \end{bmatrix} \vee \begin{bmatrix} Y_2 \\ 2 - x_2 \le 0 \\ c \le 7 \end{bmatrix} \\
&\qquad \begin{bmatrix} Y_3 \\ x_1 - x_2 \le 0 \end{bmatrix} \vee \begin{bmatrix} \neg Y_3 \\ x_1 \le 1 \end{bmatrix} \\
&\qquad Y_1 \wedge \neg Y_2 \Rightarrow \neg Y_3 \\
&\qquad Y_2 \Rightarrow \neg Y_3 \\
&\qquad Y_3 \Rightarrow \neg Y_2 \\
&\qquad 0 \le x_1 \le 5 \\
&\qquad 0 \le x_2 \le 5 \\
&\qquad c \ge 0 \\
&\qquad Y_j \in \{\text{True}, \text{False}\},\ j = 1, 2, 3
\end{aligned}
$$

فرمول ۵-۱

مثال فوق را می‌توان در بستر GAMS EMP به صورت زیر در قطعه کد ۵-۱ پیاده‌سازی کرد.

```
Set i / 1*2 /
    j / 1*3 /;

Positive Variables x(i), c;
Variable           z;
Binary Variables   y(j);

x.up(i) = 5;
c.up    = 7;

Equations Obj, Eq1, Eq2, Eq3, Eq4, Eq5, Eq6;

Obj..   z =e= c + 2*x('1') + x('2');
```

```
* Equations for Disjunctions
Eq1..   x('2') - x('1') =l=  - 2;
Eq2..   c =l= 5;
Eq3..   x('2') =g= 2;
Eq4..   c =l= 7;
Eq5..   x('1') - x('2') =l= 1 ;
Eq6..   x('1') =l= 1;

* Equations for Logic Propositions
Logic Equations  LEq1, LEq2, LEq3;

LEq1..   y('1') and not y('2') -> not y('3');
LEq2..   y('2') -> not y('3');
LEq3..   y('3') -> not y('2');

Model small1 / all /;

File emp / '%emp.info%' /;
put emp;
$onput
disjunction y('1') Eq1 Eq2 elseif y('2') Eq3 Eq4
disjunction y('3') Eq5 else Eq6
$offput
putclose;

Option optcr = 0.0;

solve small1 using EMP minimize z;
```

کد ۵-۱

این ابزار یک به صورت خودکار یک نمونه از مسئله‌ی برنامه‌ریزی فصلی تعمیم‌یافته را برای حل ابتدا به یک نمونه از مسئله‌ی برنامه‌ریزی خطی صحیح ترکیبی تبدیل کرده و سپس با استفاده از حل‌کننده‌های موجود در GAMS اقدام به حل آن می‌نماید.

خروجی برنامه‌ی فوق علاوه بر مقدار تابع هدف و مقادیر متغیرها، شامل اطلاعات جانبی دیگری مانند نمونه‌ی زیر می‌باشد.

```
--- EMP Summary
    Logical Constraints = ۳
    Disjunctions        = ۲
...
...
--- Disjunction Summary
    Disjunction ۱ Term ۲ is active
    Disjunction ۲ Term ۲ is active
```

در قسمت EMP Summary خلاصه‌ای از ساختار مسئله‌ی برنامه‌ریزی فصلی عمومیت‌یافته دیده می‌شود و در قسمت Disjunction Summary، مشخص می‌شود که کدام یک از عبارات موجود در ترکیبات فصلی فعال می‌باشند.

کتاب‌خانه‌ی Pyomo به زبان پایتون

ابزار دیگری که می‌توان با استفاده از آن اقدام به پیاده‌سازی و حل نمونه‌های برنامه‌ریزی خطی نمود، کتابخانه‌ی Pyomo و ماژول gdp موجود در آن به زبان پایتون نام دارد.

برای آشنایی با این ابزار، ترکیب فصلی ساده‌ی مطرح شده در فرمول ۵-۲ را در نظر بگیرید.

$$[x = ۰] \quad \vee \quad [y = ۰]$$

فرمول ۵-۲

با استفاده از ماژول gdp موجود در کتاب‌خانه‌ی Pyomo میتوان سه روش زیر را برای پیاده‌سازی فرمول ۵-۲ پیشنهاد داد که در قطعه کد ۵-۲ قابل مشاهده می‌باشند. دقت شود که در این مثال تنها روی توانایی این ابزار در پیاده‌سازی یک ترکیب فصلی تمرکز کرده‌ایم و نمونه‌های بزرگ‌تر در ادامه مورد بررسی قرار می‌گیرند.

Option 1: Rule-based construction

```
>>> from pyomo.environ import *
>>> from pyomo.gdp import *
>>> model = ConcreteModel()

>>> model.x = Var()
>>> model.y = Var()

>>> # Two conditions
>>> def _d(disjunct, flag):
...    model = disjunct.model()
...    if flag:
...       # x == 0
...       disjunct.c = Constraint(expr=model.x == 0)
...    else:
...       # y == 0
...       disjunct.c = Constraint(expr=model.y == 0)
>>> model.d = Disjunct([0,1], rule=_d)

>>> # Define the disjunction
>>> def _c(model):
...    return [model.d[0], model.d[1]]
>>> model.c = Disjunction(rule=_c)
```

Option 2: Explicit disjuncts

```
>>> from pyomo.environ import *
>>> from pyomo.gdp import *
>>> model = ConcreteModel()

>>> model.x = Var()
>>> model.y = Var()

>>> model.fix_x = Disjunct()
>>> model.fix_x.c = Constraint(expr=model.x == 0)

>>> model.fix_y = Disjunct()
>>> model.fix_y.c = Constraint(expr=model.y == 0)

>>> model.c = Disjunction(expr=[model.fix_x, model.fix_y])
```

Option 3: Implicit disjuncts (disjunction rule returns a list of expressions or a list of lists of expressions)

```
>>> from pyomo.environ import *
>>> from pyomo.gdp import *
>>> model = ConcreteModel()
```

```
>>> model.x = Var()
>>> model.y = Var()

>>> model.c = Disjunction(expr=[model.x == 0, model.y == 0])
```

کد ۵-۲

ماژول gdp موجود در ابزار pyomo طیف وسیعی از عملگرهای منطقی[۴۳] را پشتیبانی می‌نماید. تعدادی از این عملگرها را می‌توان در جدول ۵-۱ مشاهده نمود.

Operator	Operator	Method	Function
Conjunction		`Y[1].land(Y[2])`	`land(Y[1],Y[2])`
Disjunction		`Y[1].lor(Y[2])`	`lor(Y[1],Y[2])`
Negation	`~Y[1]`		`lnot(Y[1])`
Exclusive OR		`Y[1].xor(Y[2])`	`xor(Y[1], Y[2])`
Implication		`Y[1].implies(Y[2])`	`implies(Y[1], Y[2])`
Equivalence		`Y[1].equivalent_to(Y[2])`	`equivalent(Y[1], Y[2])`

جدول ۵-۱

به عنوان مثالی دیگر قطعه کد ۵-۳ به ما نشان می‌دهد که چگونه می‌توان عبارات مطرح شده در فرمول ۵-۳ را در بستر این ابزار و با بهره‌گیری از عملگرهای مطرح شده در جدل ۵-۱ پیاده‌سازی نمود.

$$\begin{bmatrix} Y_1 \\ \exp(x_2) - 1 = x_1 \\ x_3 = x_4 = 0 \end{bmatrix} \bigvee \begin{bmatrix} Y_2 \\ \exp\left(\frac{x_4}{1.2}\right) - 1 = x_3 \\ x_1 = x_2 = 0 \end{bmatrix}$$

فرمول ۵-۳

[۴۳] Logical Operators

```
>>> m.unit1 = Disjunct()
>>> m.unit1.inout = Constraint(expr=exp(m.x[2]) - 1 == m.x[1])
>>> m.unit1.no_unit2_flow1 = Constraint(expr=m.x[3] == 0)
>>> m.unit1.no_unit2_flow2 = Constraint(expr=m.x[4] == 0)
>>> m.unit2 = Disjunct()
>>> m.unit2.inout = Constraint(expr=exp(m.x[4] / 1.2) - 1 == m.x[3])
>>> m.unit2.no_unit1_flow1 = Constraint(expr=m.x[1] == 0)
>>> m.unit2.no_unit1_flow2 = Constraint(expr=m.x[2] == 0)
>>> m.use_unit1or2 = Disjunction(expr=[m.unit1, m.unit2])
```

کد ۵-۳

$$Y_{i+1} \Rightarrow Y_i, \quad i \in \{1, 2, \ldots, n-1\}$$

فرمول ۵-۴

از دیگر قابلیت‌های این ابزار می‌توان به توانایی کار با متغیرهای اندیس‌دار اشاره کرد. این قابلیت با مثالی که در فرمول ۵-۴ ذکر و در قطعه کد ۵-۴ نیز پیاده‌سازی شده به خوبی قابل مشاهده است.

```
>>> m = ConcreteModel()
>>> n = 5
>>> m.I = RangeSet(n)
>>> m.Y = BooleanVar(m.I)

>>> @m.LogicalConstraint(m.I)
... def p(m, i):
...     return m.Y[i+1].implies(m.Y[i]) if i < n else Constraint.Skip

>>> m.p.pprint()
p : Size=4, Index=I, Active=True
    Key : Body           : Active
      1 : Y[2] --> Y[1] :   True
      2 : Y[3] --> Y[2] :   True
      3 : Y[4] --> Y[3] :   True
      4 : Y[5] --> Y[4] :   True
```

کد ۵-۴

$$
\begin{aligned}
\min\ & x \\
\text{s.t.}\ & \begin{bmatrix} Y_1 \\ x \geq 2 \end{bmatrix} \vee \begin{bmatrix} Y_2 \\ x \geq 3 \end{bmatrix} \\
& \begin{bmatrix} Y_3 \\ x \leq 8 \end{bmatrix} \vee \begin{bmatrix} Y_4 \\ x = 2.5 \end{bmatrix} \\
& Y_1 \veebar Y_2 \\
& Y_3 \veebar Y_4 \\
& Y_1 \Rightarrow Y_4
\end{aligned}
$$

فرمول ۵-۵

حال با توجه به متدهای مطرح شده برای پیاده‌سازی یک ترکیب فصلی ساده با استفاده از ابزار gdp در Pyomo، می‌توان برنامه‌ریزی فصلی تعمیم‌یافته در فرمول ۵-۵ را به صورت زیر در قطعه کد ۵-۵ در قالب این ابزار پیاده‌سازی کرد.

```
>>> m = ConcreteModel()
>>> m.s = RangeSet(4)
>>> m.ds = RangeSet(2)
>>> m.d = Disjunct(m.s)
>>> m.djn = Disjunction(m.ds)
>>> m.djn[1] = [m.d[1], m.d[2]]
>>> m.djn[2] = [m.d[3], m.d[4]]
>>> m.x = Var(bounds=(-2, 10))
>>> m.d[1].c = Constraint(expr=m.x >= 2)
>>> m.d[2].c = Constraint(expr=m.x >= 3)
>>> m.d[3].c = Constraint(expr=m.x <= 8)
>>> m.d[4].c = Constraint(expr=m.x == 2.5)
>>> m.o = Objective(expr=m.x)

>>> # Create Boolean variables associated with the disjuncts.
>>> m.Y = BooleanVar(m.s)
>>> for idx in m.Y:
...     m.Y[idx].associate_binary_var(m.d[idx].indicator_var)

>>> # Add the logical proposition
>>> m.p = LogicalConstraint(expr=m.Y[1].implies(m.Y[4]))
>>> # Note: the implicit XOR enforced by m.djn[1] and m.djn[2] still apply

>>> # Convert logical propositions to linear algebraic constraints
>>> # and apply the Big-M reformulation.
>>> TransformationFactory('core.logical_to_linear').apply_to(m)
>>> TransformationFactory('gdp.bigm').apply_to(m)
```

```
>>> m.Y.display()  # Before solve, Boolean vars have no value
Y : Size=4, Index=s
    Key : Value : Fixed : Stale
      1 :  None : False :  True
      2 :  None : False :  True
      3 :  None : False :  True
      4 :  None : False :  True

>>> # Solve the reformulated model and update the Boolean variables
>>> # based on the algebraic model results
>>> run_data = SolverFactory('glpk').solve(m)
>>> update_boolean_vars_from_binary(m)
>>> m.Y.display()
Y : Size=4, Index=s
    Key : Value : Fixed : Stale
      1 :  True : False : False
      2 : False : False : False
      3 : False : False : False
      4 :  True : False : False
```

کد ۵-۵

که در این کد بالا خروجی‌ها نیز نمایش داده‌شده‌اند. دقت شود که نماد به کار رفته در خطوط ۴ و ۵ از فرمول ۵-۵ نشان‌دهنده‌ی عملگر xor می‌باشند.

برای مطالعه‌ی بیشتر در مورد این ابزار و کاربردهای آن در حل مسائل برنامه‌ریزی تعمیم‌یافته، به خواننده توصیه می‌شود که به مستندات رسمی این ماژول مراجعه نماید.

معرفی benckmark برای مسئله‌ی برنامه‌ریزی فصلی

در بخش دیگری از فرایند تحقیق در مورد برنامه‌ریزی فصلی در طول این پروژه سعی شد تعدادی benckmark معتبر یافت و معرفی شود. بررسی‌های مختلف نشان داد که benckmark مختص مسئله‌ی مذکور تا زمان تحقیق وجود نداشته‌است. اما در مقالات [۱۲] و [۱۳] دو benckmark از مسائل مسیریابی[۴۴] و شبکه‌های تولید محصول[۴۵] معرفی شده‌است که نمونه‌های آن‌ها در بستر برنامه‌ریزی فصلی بررسی و حل شده‌اند و مطالعه‌ی این دو منبع ممکن است خالی از لطف نباشد.

---

[۴۴] Routing
[۴۵] Network Process

همان‌گونه که مشاهده شد در این فصل دو ابزار برنامه‌نویسی برای پیاده‌سازی و حل یک نمونه‌ی برنامه‌ریزی فصلی تعمیم‌یافته معرفی شده و مورد بررسی قرار گرفت. در فصل آینده، پیرامون مطالب بیان شده در این گزارش بحث و نتیجه‌گیری خواهیم داشت.

فصل ششم: نتیجه‌گیری

در این گزارش برای حل آن دسته از مسائل دنیای واقعی که از جنس بهینه‌سازی می‌باشند، رویکردی تحت نام برنامه‌ریزی فصلی معرفی شد که هسته‌ی اصلی آن قیودی به فرم ترکیبات فصلی بودند. پس از بررسی ساختار مسئله‌ی برنامه‌ریزی فصلی و حالت تعمیم‌یافته‌ی آن یعنی مسئله‌ی برنامه‌ریزی فصلی تعمیم‌یافته، مثال‌هایی از مدل‌سازی مسائل موجود در حوزه‌های گوناگون در قالب رویکرد ذکر شده مطرح گردید. در میان این مثال‌ها دریافتیم که طبیعت برنامه‌ریزی فصلی بیشتر منطبق بر مسائلی است که در طی حل آن‌ها با تصمیم‌گیری‌های گسسته مواجه هستیم. در این بخش به یقین مطالعه‌ی مدل‌سازی‌های بیشتر می‌تواند به درک عمیق‌تر خواننده از قدرت این رویکرد منجر شود.

در باب حل یک برنامه‌ریزی فصلی تعمیم‌یافته با دو روش آشنا شدیم که به واسطه‌ی استفاده از آن‌ها می‌توان یک نمونه از برنامه‌ریزی فصلی را به یک نمونه از برنامه‌ریزی فصلی صحیح ترکیبی تبدیل نمود و سپس اقدام به حل آن کرد. بررسی سایر روش‌های حل روش مورد بحث در این گزارش می‌تواند به عنوان کارهای آینده برای علاقه‌مندان در نظر گرفته‌شود. در انتها نیز دریافتیم شناخت ابزارهای برنامه‌نویسی برای پیاده‌سازی و حل نمونه‌های برنامه‌ریزی فصلی عمومیت یافته تا چه میزان می‌تواند مفید و موثر واقع گردد.

از دیگر مواردی که می‌تواند به عنوان موضوع خوبی برای تحقیقات آینده در نظر گرفته‌شود، بحث دوگان در برنامه‌ریزی فصلی است که به در این پروژه مورد بررسی قرار نگرفت. محاسبه‌ی دوگان یک برنامه‌ریزی فصلی از یک سو و استفاده از قضایای مربوطه از سوی دیگر می‌تواند ما را به درک عمیق‌تر و بهتری از این رویکرد برساند.

در انتها می‌توان گفت که برنامه‌ریزی فصلی ابزاری قدرتمند برای مدل‌سازی دسته‌ی بزرگی از مسائل به شمار می‌رود به گونه‌ای که امروزه به نظر می‌رسد مسلط بودن به این ابزار برای هر مهندسی که کار او مرتبط با مسائل بهینه‌سازی است، لازم می‌باشد.

مراجع

[١] E. M. L. Beale, “Survey of Integer Programming ”,*Journal of the Operational Research Society,*

[٢] L. V. a. S. Boyd, “Semidefinite Programming ”,*SIAM Review* .

[٣] R. M. J. A. Nelder, “A Simplex Method for Function Minimization ”,*The Computer Journal ,*

[٤] L. L .& .A. S. M. Grötschel, “The ellipsoid method and its consequences in combinatorial optimization ”,*Combinatorica* .

[٥] F. R. R. J. V. a. H. W. Christoph Helmberg, “An Interior-Point Method for Semidefinite Programming ”,*SIAM Journal on Optimization ,*

[٦] E. Balas, “Disjunctive Programming ”,*Annals of Discrete Mathematics ,*

[٧] F. T. P. I. Grossmann, “Review of Mixed-Integer Nonlinear and Generalized Disjunctive Programming Methods ”,*Chemie Ingenuier Technic.,*

[٨] I. E. G. Francisco Trespalacios, “Symmetry breaking for generalized disjunctive programming formulation of the strip packing problem ”,*Annals of Operations Research ,*

[٩] L. Blackmore, “Optimal manipulator path planning with obstacles using disjunctive programming ”,*Proceedings of the American Control Conference* .

[١٠] I. M. Pedro M. Castro, “Operating room scheduling with generalized ”,*Computers & Operations Research,*

[١١] I. E. Grossmann, “Generalized Disjunctive Programming: A Framework for Formulataion and Alternative Algorithms for MILP Optimization ”,*Encyclopedia of Optimization* .

[١٢] Q. H. Ricardo Fukasawa, “A disjunctive convex programming approach to the pollution-routing problem ”,*Transportation Research Part B: Methodological ,*

[١٣] M. T. Ali Fattahi, “ϵ-OA for the solution of bi-objective generalized disjunctive programming problems in the synthesis of nonlinear process networks ”,*Computers & Chemical Engineering ,*