اسامی گروه:

ظهیر سلیم علاقه بند (۹۸۰۸۸۰۴)

سیاوش مندگاری (۹۸۱۵۵۴۴)

سروش سعیدی (۹۸۰۸۶۷۴)

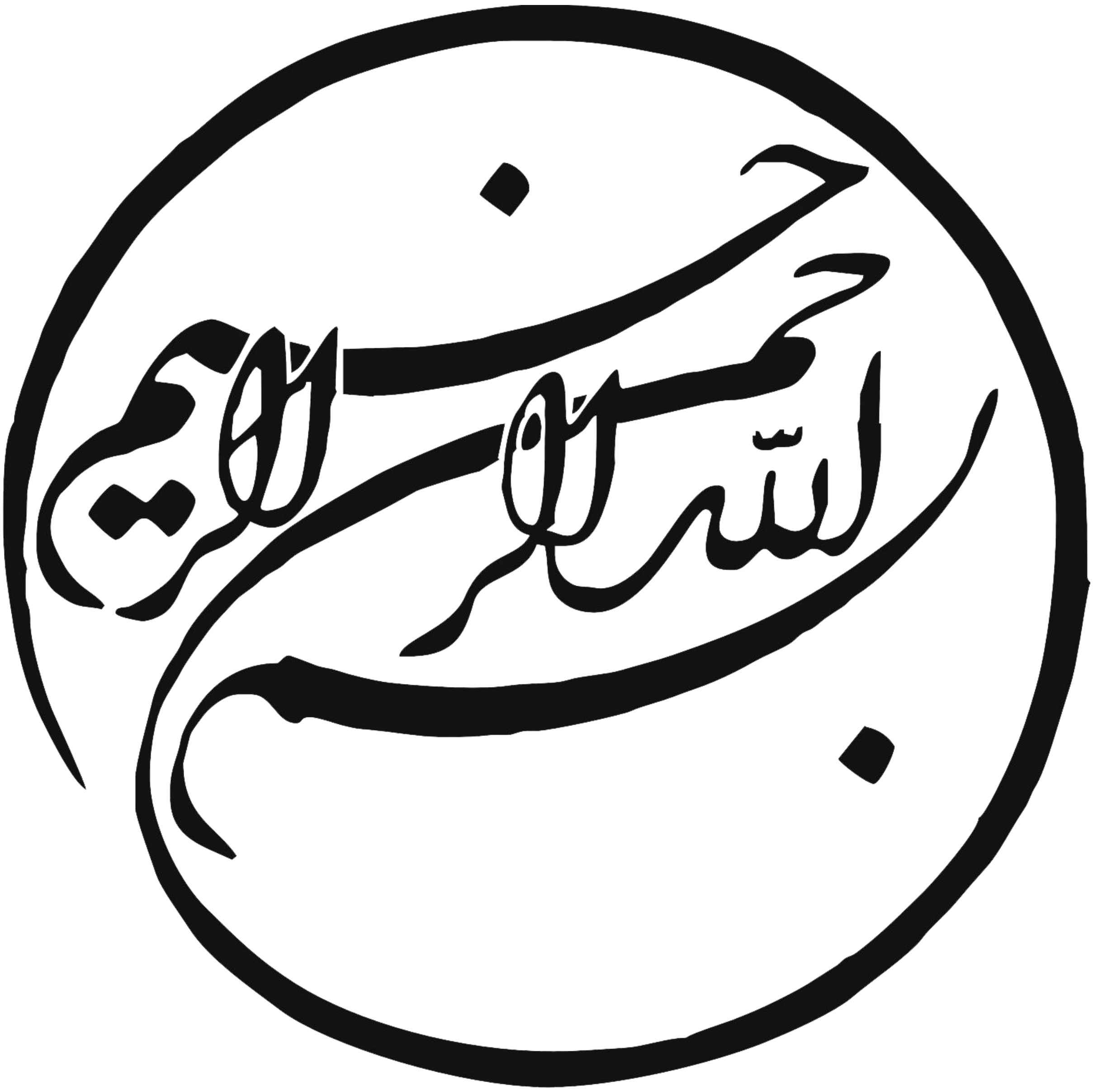
استاد پروژه:

دکتر سید جواد حسینی نژاد

پروژه درس برنامه‌ریزی متغیرهای عدد صحیح

کارشناسی ارشد گرایش بهینه‌سازی سیستم‌ها

بهمن ۱۳۹۹


بسم الله الرحمن الرحیم

۱- کلیات

یکی از اهداف مهم سرویس‌دهی به مشتری‌ها رعایت کردن بحث ارائه خدمت به آن‌ها در موعد مقرر خود است، که اگر این سرویس‌دهی با تأخیر انجام شود، اصطلاحاً با دیرکرد مواجه شده‌ایم و این دیرکرد خود می‌تواند معیاری برای اندازه‌گیری اثربخشی زمان‌بندی جریان کاری ما باشد. زمانی که کارها با تأخیر مواجه می‌شوند، درخواست خریدهایی که به دپارتمان ما وارد می‌شود ازدست‌رفته و در درجه بالاتر ممکن است مشتری‌ها به تأمین‌کننده دیگری روی بیاورند. از طرف دیگر اگر کارها زودتر از موعد مقرر انجام شوند، با زودکرد مواجه خواهیم شد. در این حالت محصول در انبار موجودی نگه‌داشته می‌شود، که نه‌تنها فضای انبار را اشغال می‌کند؛ بلکه باعث می‌شود مقداری از سرمایه در انبار موجودی راکد مانده و محبوس شود. یکی از راه‌هایی که مجموع زمان زودکرد را کاهش می‌دهد، اعمال تأخیر در زمان پردازش کارها و درنتیجه زمان اتمام آن‌هاست. این راه‌حل را در مدل‌سازی مسئله این تحقیق خواهیم دید.

منابعی که در این مقاله به‌وسیله آن‌ها سعی در پردازش کارها با آن‌ها را داریم درواقع دو ماشین است که جریان کاری بر روی هر دو ماشین با توالی یکسانی از آن‌ها عبور می‌کنند. این حالت در مقالات بیشتر مورداستفاده قرارگرفته و اصطلاحاً به آن *Permutation flow shop* می‌گویند. حالت *Permutation* به دو دلیل در تحقیقات بیشتر دیده می‌شود. یکی به دلیل اینکه ازلحاظ مفهومی مطالعه آن ساده‌تر است و دیگری به این خاطر است که اغلب تغییر ترتیب انجام کارها از ماشینی به ماشین دیگر پیچیده است.

چیزی که در این مقاله به دنبال آن هستیم بررسی مسئله زمان‌بندی جریان کار در یک کارگاه با دو ماشین باهدف کمینه‌سازی مجموع زمان دیرکرد و زودکرد کارهاست. همان‌طور که گفته شد تمامی کارها با توالی یکسانی از ترتیب، بر روی هر ماشین پردازش می‌شود و در مدل از دو نوع مدت‌زمان بیکاری استفاده خواهیم کرد که آن‌ها را در بخش بعدی توضیح می‌دهیم.

۲- توصیف مدل

در مدل بررسی‌شده به‌طورکلی n کار را در دست انجام داریم که بایستی در کارگاهی با m ماشین $(m=2)$، آن‌ها را پردازش کنیم. مقادیر d_j زمان‌های مقرر انجام هر کار j است $(j=1,...,n)$. زمان‌های پردازش و اتمام هر کار بر روش ماشین m به ترتیب با p_{jm} و C_{jm} نشان داده می‌شوند. میزان زمان زودکرد هر کار j، E_j بوده و بدین صورت تعریف می‌شود: $E_j = \max\{d_j - C_{j2}, 0\}$. همچنین زمان دیرکرد کار j، T_j بوده و داریم: $T_j = \max\{C_{j2} - d_j, 0\}$.

پس هدف ما در اینجا کمینه‌سازی مجموع این زمان‌هاست؛ یعنی کمینه‌سازی عبارت: $Z = \sum_{j=1}^{n} E_j + T_j$. اما این، تابع هدفی نیست که در مدل به دنبال آن هستیم. در بخش‌های بعدی تابع هدف موردنظر را نیز معرفی

می‌کنیم. در اینجا نیاز است که دو نوع زمان بیکاری را که به کاهش یافتن زمان‌های زودکرد کمک می‌کند معرفی کنیم. یکی از آن‌ها *Unforced idle time* بوده که با اعمال آن به مدل با افزایش زمان اتمام کارهایی که احتمالاً زود به اتمام می‌رسند، به بهبود تابع هدف کمک خواهد کرد. دیگری *Forced idle time* بوده و زمانی مورداستفاده است که یک ماشین جدیداً در دسترس قرارگرفته اما کار بعدی که بایستی بر روی آن ماشین پردازش شود، هنوز آماده نشده است. نوتاسیون به‌کاررفته برای این دو نوع زمان بیکاری به ترتیب $UI_{[j]m}$ و $FI_{[j]m}$ می‌باشند. علامت $[j]$ به این معناست که کاری در پوزیشن j توالی انجام کار، قرارگرفته است. نکته مهم دیگر شیوه تعریف زمان اتمام هر کار در پوزیشن j بر روی هر دو ماشین است. این معادلات در محدودیت‌های مدل نیز به‌وضوح دیده می‌شوند:

$$C_{[0]1} = C_{[0]2} = 0$$
$$C_{[j]1} = C_{[j-1]1} + UI_{[j]1} + p_{[j]1}$$
$$C_{[j]2} = \max\{C_{[j]1} + C_{[j-1]2}\} + UI_{[j]2} + p_{[j]2}$$

به‌طورکلی می‌توان گفت برای یک توالی در نظر گرفته‌شده از کارها، زمان اتمام کاری در پوزیشن $[j]$ از ماشین دوم، دیرکرد یا زودکرد آن را تعیین می‌کند. همچنین توجه به این نکته نیز ضروری است که اضافه کردن زمان بیکاری از نوع *Unforced idle time* به کارهای انجام‌شده بر روی ماشین یک ضروری نیست؛ چراکه در مواقعی باعث بدتر شدن تابع هدف نیز می‌شود.

نوتاسیون کامل مدل را در بخش بعدی خواهیم دید.

۲-۱ نوتاسیون:

۲-۱-۱ اندیس‌ها:

j: اندیس کار $(j = 1, 2, ..., n)$

m: اندیس ماشین $(m = 1 \; or \; 2)$

۲-۱-۲ پارامترها:

n: تعداد کارها

d_j: موعد مقرر اتمام کار j

p_{jm}: مدت زمان پردازش کار j بر روی ماشین m

$[j]$: کاری که در پوزیشن j قرارگرفته است.

$FI_{[j]m}$: زمان بیکاری از نوع *Forced* بر روی ماشین m برای کاری که در پوزیشن jام قرارگرفته است.

$UI_{[j]m}$: زمان بیکاری از نوع *Unforced* بر روی ماشین m برای کاری که در پوزیشن j ام قرارگرفته است.

$I_{[j]m}$: کل زمان بیکاری بر روی ماشین m در کاری که در پوزیشن j ام قرارگرفته $(I_{[j]m} = FI_{[j]m} + UI_{[j]m})$.

۲-۱-۳ متغیرهای حالت:

E_j: میزان زودکرد کار j، $E_j = \max\{d_j - C_{j2}, 0\}$

T_j: میزان دیرکرد کار j، $T_j = \max\{C_{j2} - d_j, 0\}$

۲-۱-۴ متغیر تصمیم:

C_{jm}: زمان اتمام کار j بر روی ماشین m

۲-۲ مدل ریاضی

$$Minimize \;\; Z = \sum_{j=1}^{n} \max\{C_{[j]2} - d_{[j]}, 0\} + \max\{d_{[j]} - C_{[j]2}, 0\} \qquad (1)$$

$$Subject\ to:$$

$$C_{[j]1} = \sum_{k=1}^{j} (p_{[k]1}) \qquad for \;\; j = 1, \ldots, n \qquad (2)$$

$$C_{[j]2} \geq C_{[j-1]2} + UI_{[j]2} + p_{[j]2} \qquad for \;\; j = 1, \ldots, n \qquad (3)$$

$$C_{[j]2} \geq C_{[j]1} + UI_{[j]2} + p_{[j]2} \qquad for \;\; j = 1, \ldots, n \qquad (4)$$

$$C_{[0]2} = 0 \qquad (5)$$

$$UI_{[j]2} \geq 0 \qquad for \;\; j = 1, \ldots, n \qquad (6)$$

۲-۲-۱ توضیح محدودیت‌های مدل

- تابع هدف مجموع زمان‌های دیرکرد و زودکرد را کمینه می‌کند.
- مجموعه محدودیت‌های ۲ و ۳ و ۴ و ۵ زمان اتمام هر کار را بر روی دومین ماشین تنظیم می‌کند. این زمان‌های اتمام با موعدهای مقرر انجام کار مقایسه شده تا به کمک نتیجه محدودیت‌ها بتوانیم میزان زودکرد و دیرکرد را در تابع هدف محاسبه کنیم. در محدودیت ۲ زمان اتمام کاری در پوزیشن j ماشین ۱ را مجموع زمان‌های پردازش تمامی کارهای ماقبل روی همین ماشین در نظر گرفته است که با فرض‌های ارائه‌شده در این مقاله هماهنگ است. اما در مورد زمان اتمام یک کار بر روی ماشین

دوم نیز پیش‌تر صحبت شد که نمود آن را در محدودیت‌های ۳ و ۴ می‌بینیم. در محدودیت ۳ زمان اتمام کار j بر روی ماشین ۲ حداقل به‌اندازه زمان اتمام کار قبلی بر روی همین ماشین است. یعنی از همین لحظه کار j بر روش ماشین ۲ شروع شده و درنهایت زمان‌های پردازش این کار و زمان بیکاری نیز افزوده می‌شود. اما در حالت دیگر که کار قبلی بر روی ماشین ۲ اتمام نیافته باید به سراغ ماشین اول می‌رویم. در محدودیت ۴ این موضوع را می‌بینیم که زمان اتمام کار j روی ماشین ۲ می‌تواند در این حالت حداقل به‌اندازه زمان اتمام همان کار روی ماشین اول باشد و درنهایت مانند حالت قبلی زمان‌های بیکاری و پردازش کار جاری را اضافه می‌کنیم. محدودیت پنجم نیز به این خاطر به مدل اضافه‌شده تا در محدودیت سوم اختلالی ازلحاظ بی‌مقدار بودن اندیس j ایجاد نشود.

- محدودیت آخر نیز بیان می‌کند که زمان‌های *Unforced idle time* بایستی نامنفی باشند.

۳- الگوریتم روش حل پیشنهادی مقاله

در این قسمت می‌بینیم که برای این مدل دو الگوریتم شاخه و کرانی پیشنهاد داده‌شده است که پیش‌تر توسط شالر[1](۲۰۰۷) برای مسئله تک ماشینه پیشنهاد داده‌شده بود. در هر دو الگوریتم موجود، هر گره در درخت شاخه و کران نمایانگر یک توالی *Partial* از کارهاست. برای هر گره در درخت شاخه و کران یک حد بالا و یک حد پایین از تابع هدف بهینه محاسبه می‌شود. یک مقدار از بهترین جواب شدنی موجود[2]که نمایش‌دهنده مقدار مجموع زمان دیرکرد و زودکرد از بهترین توالی موجود می‌باشد، با یک حد پایین که برای یک گره پیداشده است، مقایسه می‌شود. اگر آن جواب شدنی کوچک‌تر یا مساوی حد پایین باشد، آن گره به عمق می‌رسد. توالی *Partial* مرتبط با هر گره به کمک یک روش ابتکاری ساده تکمیل می‌شود و مقدار تابع هدف آن محاسبه می‌گردد تا یک حد بالا را به دست آوریم. اگر مقدار این حد بالا کوچک‌تر از بهترین جواب شدنی موجود باشد، درنتیجه جواب شدنی به‌روز خواهد شد و آن توالی به‌عنوان بهترین توالی موجود تا به الآن نگه‌داشته می‌شود. یک بهترین جواب شدنی موجود آغازین و حل آن، با مرتب کردن کارها به ترتیب زودترین زمان‌های موعد مقرر[3]، واردکردن زمان‌های بیکاری به این توالی و محاسبه تابع هدف کل زمان دیرکرد و زودکرد به دست می‌آید.

تفاوت مابین دو الگوریتم پیشنهادی این است که دریکی از آن‌ها که به آن *Initial partial sequence* می‌گوییم، اگر یکی از گره‌ها چنین حالتی را داشته باشد، توالی ما از ابتدا ساخته می‌شود یا به‌عبارت‌دیگر، با کار اول شروع به پردازش می‌کند و تا انتهای توالی به‌پیش می‌رود تا آخرین کار را نیز پردازش کند. در طرف مقابل حالت *Post partial sequence* را داریم که اگر گرهی چنین وضعیتی را داشت، توالی ما از انتها ساخته

[1] *Schaller*
[2] *Incumbent*
[3] *Earliest due date (EDD)*

می‌شود یا به‌عبارت‌دیگر، از آخرین کار شروع به پردازش می‌کند و به سمت ابتدای توالی به عقب برمی‌گردد تا اولین کار را نیز پردازش کند.

۳-۱ حدود پایین پیشنهادی

همان‌طور که گفته شد، حدود پایین که در این قسمت توسعه داده می‌شوند، مبتنی بر حدود پایین پیشنهادی توسط شالر (۲۰۰۷) برای مسئله تک ماشینه می‌باشد که به‌تناسب شرایط موجود اصلاح‌شده تا حالت دو ماشینه را نیز منعکس کند. برای سادگی کار تنها حالت *Post partial sequence* را در نظرخواهیم گرفت تا حجم مطالب خیلی زیاد نشود.

فرض کنید در حالت *Post partial sequence (PPS)* گرهی داریم که نمایانگر تعداد p کار در یک *PPS* است که این توالی را σ می‌نامیم. همچنین فرض کنید $q=n-p$ و نیز σ' مجموعه دربرگیرنده این q کار باشد که هنوز در توالی قرار نگرفته‌اند. p تعداد کارهایی است که در توالی *Partial* از σ' وجود دارد. زمانی که توالی (شامل کارهای مجموعه σ') را تکمیل می‌کنیم، یک الگوریتم زمان‌بندی جدول ساعات کار برای حل مدل نهایی که بعداً معرفی می‌کنیم برای به دست آوردن مقدار تابع هدف استفاده خواهد شد. با لحاظ نکردن کارهای مجموعه σ' نیز می‌توانیم یک حد پایین با حل مدل برای مجموعه σ به دست آوریم، اما این حد پایین ضعیف خواهد بود. حد پایین می‌تواند با لحاظ نمودن کارهایی که هنوز در توالی قرار نگرفته‌اند (کارهای مجموعه σ') بهبود یابد. برای توسعه حد پایین از نوتاسیون زیر استفاده می‌کنیم:

$P(SPT_{jm})$: مجموع زمان‌های پردازش j کار با کمترین زمان‌های پردازش بر روی ماشین *m*، در مجموعه σ'

$P(LPT_{jm})$: مجموع زمان‌های پردازش j کار با بزرگ‌ترین زمان‌های پردازش بر روی ماشین *m*، در مجموعه σ'.

$d_{EDD[j]}$: موعد مقرر j امین کار زمانی که کارهای مجموعه σ' به ترتیب زودترین موعد مقرر *(EDD)*، مرتب‌شده باشند $(d_{EDD[j]} \le d_{EDD[k]}\ if\ \ j<k)$.

$LBC_{[j]2}$: یک حد پایین برای زمان اتمام بر روی دومین ماشین و برای کاری که در پوزیشن j ام توالی قرارگرفته است؛ زمانی که *Unforced idle time* استفاده‌نشده باشد.

$I_{[j]2}$: یک زمان بیکاری بر روی دومین ماشین برای کار پوزیشن j ام.

شالر (۲۰۰۷) ثابت کرد که اگر موعدهای مقرر به ترتیب *EDD*، جایگزین موعدهای مقرر حقیقی شوند و با زمان‌های اتمام حقیقی مقایسه گردند، یک حد پایین برای کل زمان دیرکرد و زودکرد به دست خواهد آمد:

$$\sum_{j=1}^{n}(\max\{C_{[j]2}-d_{EDD[j]},0\}+\max\{d_{EDD[j]}-C_{[j]2},0\}) \le \sum_{j=1}^{n}(\max\{C_{[j]2}-d_{[j]},0\}+\max\{d_{[j]}-C_{[j]2},0\})$$

که می‌تواند به‌عنوان جایگزینی برای اولین معادله در مدل اصلی استفاده شود. همچنین مطلع هستیم که ما زمان‌های اتمام حقیقی ($C_{[j]2}$) کارها را در یک توالی نمی‌دانیم. اگرچه می‌توانیم حدود بالا و پایین این زمان‌های اتمام را با استفاده از معادلات زیر محاسبه کنیم:

$$P(SPT_{j2})+\sum_{k=1}^{j} I_{[k]2} \le C_{[j]2} \quad for\ \ j=1,...,q$$

$$\sum_{k\in\sigma'} p_{k2}+\sum_{k=q+1}^{j} p_{[k]2}+\sum_{k=1}^{j} I_{[k]2} \le C_{[j]2} \quad for\ \ j=q+1,...,n$$

$$P(LPT_{j2})+\sum_{k=1}^{j} I_{[k]2} \ge C_{[j]2} \quad for\ \ j=1,...,q$$

$$\sum_{k\in\sigma'} p_{k2}+\sum_{k=q+1}^{j} p_{[k]2}+\sum_{k=1}^{j} I_{[k]2} \ge C_{[j]2} \quad for\ \ j=q+1,...,n$$

همچنین می‌دانیم که زمان اتمام کاری در پوزیشن j از یک توالی بایستی بزرگ‌تر و مساوی حد پایین زمان اتمامی باشد که هیچ‌گونه *Unforced idle time* در آن برای کار پوزیشن j ام استفاده‌نشده باشد ($C_{[j]2} \ge LBC_{[j]2}$). این معادلات می‌توانند جایگزین معادلات ۲ و ۳ و ۴ از مدل اصلی شوند. درنتیجه می‌توانیم به کمک مدل زیر با داشتن یک توالی *Post partial* بنام σ یک حد پایین به دست آوریم:

$$Minimize\ \ Z_{LB}=\sum_{j=1}^{q}(\max\{C_{[j]2}-d_{EDD[j]},0\}+\max\{d_{EDD[j]}-C_{[j]2},0\})$$

$$+\sum_{j=q+1}^{n}(\max\{C_{[j]2}-d_{[j]},0\}+\max\{d_{[j]}-C_{[j]2},0\}) \qquad (7)$$

Subject to:

$$P(SPT_{j2})+\sum_{k=q+1}^{j} p_{[k]2}+\sum_{k=1}^{j} I_{[k]2} \le C_{[j]2} \quad for\ \ j=1,...,q \qquad (8)$$

$$\sum_{k\in\sigma'} p_{k2}+\sum_{k=q+1}^{j} p_{[k]2}+\sum_{k=1}^{j} I_{[k]2} \le C_{[j]2} \quad for\ \ j=q+1,...,n \qquad (9)$$

$$P(LPT_{j2})+\sum_{k=1}^{j} I_{[k]2} \ge C_{[j]2} \quad for\ \ j=1,...,q \qquad (10)$$

$$\sum_{k\in\sigma'} p_{k2}+\sum_{k=q+1}^{j} p_{[k]2}+\sum_{k=1}^{j} I_{[k]2} \ge C_{[j]2} \quad for\ \ j=q+1,...,n \qquad (11)$$

$$C_{[j]2} \ge LBC_{[j]2} \quad for\ \ j=1,...,n \qquad (12)$$

$$I_{[j]2} \ge 0 \quad for\ \ j=1,...,n \qquad (13)$$

- معادله هفتم تابع هدف مدل ریاضی ماست. زمان‌های موعد مقرر کارهایی که هنوز در توالی قرار نگرفته‌اند (σ') به ترتیب EDD مرتب می‌شوند. این زمان‌های موعد مقرر برای q پوزیشن اول استفاده می‌شوند و موعدهای مقرر حقیقی برای پوزیشن های دیگر بکار می‌روند. زمان‌های اتمام و زمان‌های بیکاری بایستی برای حل مدل ریاضی مشخص شوند.

- مجموعه محدودیت هشتم یک حد پایین برای زمان‌های اتمام بر روی دومین ماشین برای q کار اول یک توالی کامل، تنظیم می‌کند. این مجموعه محدودیت زمان اتمام کاری که در پوزیشن j ام از یک توالی کامل قرارگرفته است را حداقل به بزرگی مجموع زمان‌های پردازش کارها در مجموعه σ' بر روی ماشین دوم که به ترتیب کوتاه‌ترین زمان‌های پردازش مرتب‌شده است، برای j کار اول را می‌طلبد، به‌علاوه هر زمان بیکاری که استفاده می‌شود.

- مجموعه محدودیت نهم یک حد پایین را برای زمان‌های اتمام بر روی دومین ماشین و برای p کار آخر از یک توالی کامل را تنظیم می‌کند. این مجموعه محدودیت زمان اتمام کاری که در پوزیشن j ام از یک توالی کامل قرارگرفته است را، برای پوزیشن های $q+1$ تا n، حداقل به بزرگی مجموع زمان‌های پردازش کارها در مجموعه σ' بر روی ماشین دوم به‌علاوه مجموع زمان‌های پردازش روی ماشین دوم از کارهایی که در پوزیشن های $q+1$ تا j قرارگرفته است می‌طلبد، به‌علاوه هر زمان بیکاری که استفاده می‌شود.

- مجموعه محدودیت دهم یک حد بالا برای زمان‌های اتمام بر روی دومین ماشین برای q کار اول یک توالی کامل، تنظیم می‌کند. این مجموعه محدودیت زمان اتمام کاری که در پوزیشن j ام از یک توالی کامل قرارگرفته است را به بزرگ‌تر نبودن از مجموع زمان‌های پردازش j کار در مجموعه σ' بر روی ماشین دوم با طولانی‌ترین زمان‌های پردازش بر روی ماشین دوم به‌علاوه مجموع زمان‌های پردازش روی ماشین دوم از کارهایی که در پوزیشن های $q+1$ تا j (اگر j بزرگ‌تر از q باشد) قرارگرفته است می‌طلبد، به‌علاوه هر زمان بیکاری که استفاده می‌شود.

- مجموعه محدودیت یازدهم یک حد بالا برای زمان‌های اتمام بر روی دومین ماشین برای p کار آخر یک توالی کامل، تنظیم می‌کند. این مجموعه محدودیت زمان اتمام کاری که در پوزیشن j ام از یک توالی کامل قرارگرفته است را، برای پوزیشن های $q+1$ تا n، به بزرگ‌تر نبودن از مجموع زمان‌های پردازش j کار در مجموعه σ' بر روی ماشین دوم، به‌علاوه مجموع زمان‌های پردازش روی ماشین دوم از کارهایی که در پوزیشن های $q+1$ تا j قرارگرفته است می‌طلبد، به‌علاوه هر زمان بیکاری که استفاده می‌شود.

- مجموعه محدودیت دوازدهم زمان اتمام کاری که در پوزیشن j از یک توالی کامل قرارگرفته است را حداقل به بزرگی یک حد پایین از زمان اتمام با لحاظ نمودن هر دو ماشین، می‌طلبد. این حد پایین جلوتر تعریف خواهد شد:

$$LBC_{[j]2} = \max\{\sum_{k\in\sigma'} p_{k1} + \sum_{k=q+1}^{j} p_{[k]1}, LBC_{[j-1]2}\} + p_{[j]2}$$

- درنهایت مجموعه محدودیت سیزدهم نیز نامنفی بودن زمان‌های بیکاری واردشده به مدل را نشان می‌دهد.

۳-۲ حدود بالای پیشنهادی

برای هر گره برای امر ارزیابی، در هر یک از الگوریتم‌های شاخه و کران، اگر حد پایین آن کمتر از مقدار بهترین جواب شدنی موجود باشد، یک حد بالایی محاسبه می‌شود. ابتدا توالی *Partial* تکمیل می‌شود. این کار با مرتب‌سازی کارهای برنامه‌ریزی نشده به ترتیب *EDD* انجام می‌شود. برای الگوریتم مخصوص *Post partial sequence* این کارها پس از توالی *Partial* اولیه قرار می‌گیرند، درحالی‌که برای الگوریتم مخصوص *Initial partial sequence* آن‌ها قبل از توالی *Partial* قرار می‌گیرند. سپس از یک الگوریتم زمان‌بندی جدول ساعات کار برای درج زمان بیکاری *Unforced*، با به حداقل رساندن مقدار هدف برای توالی استفاده می‌شود. زودکرد و دیرکرد کلی این برنامه حد بالا است. اگر حد بالا کمتر از مقدار بهترین جواب شدنی موجود باشد، مقدار بهترین جواب شدنی موجود به‌روز می‌شود و این توالی به‌عنوان بهترین راه‌حل پیداشده نگه‌داشته می‌شود.

۴- کد نویسی ساده مدل در نرم‌افزار گمز

در این بخش ابتدا مدل ساده گمز بدون اینکه هیچ روش حل دقیقی بر روی آن اعمال‌شده باشد، همراه با نتایج آن در بخش‌های بعدی آورده می‌شود و پس‌ازآن کد نویسی روش حل دقیق نیز ارائه خواهد شد. داده‌هایی که در این مدل از آن‌ها استفاده کردیم اکثراً به‌صورت تصادفی واردشده‌اند و در سایر بخش‌ها نیز از داده‌های پیشنهادشده خود مقاله استفاده‌شده است. همچنین تعداد ۱۰ کار را به مدل داده‌ایم و زمان‌های بیکاری تمام فعالیت‌ها را روی اولین ماشین طبق پیشنهاد خود مقاله صفر در نظر گرفته‌ایم تا باعث افزایش مقدار تابع هدف نشود. توجه به این نکته ضروری است که حین اجرای مدل با توجه به اینکه تابع هدف مدل غیرخطی است و نوع مدل *MINLP* در گمز در حل این‌گونه مدل‌ها ضعف دارد، ابتدا این معادله را در قالب دو محدودیت که دارای متغیرهایی پیوسته می‌باشند نوشته و سپس از مدل *MIP* گمز برای حل مدل استفاده کرده‌ایم. همچنین این مدل را نیز بر روی حل‌کننده *BARON* تنظیم کرده‌ایم تا گمز مستقیماً از طریق همین حل‌کننده *Solver* اقدام به حل مدل نماید. یکی از ضعف‌های مدل *MINLP* در فضای مدل ما این بود که با تغییر ناچیز بعضی از پارامترها بخصوص اعشاری کردن آن‌ها، مدل ازلحاظ عدد صحیح بودن نشدنی می‌شد، درحالی‌که در پنجره اجرای مدل به این نکته نیز اشاره می‌شد که گمز هیچ تضمینی بابت صحیح نشدن بعضی از جواب‌ها ارائه

نمی‌دهد. به همین خاطر نیز تصمیم به خطی سازی مدل نمودیم تا مدل را از این لحاظ انعطاف‌پذیرتر کنیم. درنهایت با توجه به اینکه نوع متغیرهای مربوط به امر خطی سازی را *Positive* در نظر گرفته‌ایم، مشکلی در چک کردن خروجی عبارت *max* تابع هدف زمانی که مقدار عبارت منفی شود را نخواهیم داشت.

۴-۱ کد ساده در نرم‌افزار گمز

در این قسمت کد ساده گمز را ارائه کرده و در بخش بعدی خروجی نرم‌افزار را می‌آوریم:

```
$Title Permutation Flow Shop Problem (General Solution)

$ontext
Branch-and-bound algorithms for minimizing total earliness and tardiness
in a two-machine permutation flow shop with unforced idle allowed
$offtext

Sets
m index for machines /1,2/
j index for jobs /1*10/
Alias (k,j);

Parameters
d(j) Job due dates /1 39,2 37,3 38,4 36,5 39,6 41,7 44,8 45,9 51,10 53/;

Table p(j,m) Processing time of job j on machine m
         1        2
1        2        2
2        8        4.1
3        5        6
4        5        7.2
5        2        8
6        1        4.6
7        2        3
8        5        1.3
9        4        3
10       6        2.4;

Table UI(j,m) Unforced idle time of job j on machine m
         1        2
```

```
1        0       6

2        0       0

3        0       3

4        0       4

5        0       0

6        0       1

7        0       0

8        0       5

9        0       0

10       0       2;


Variables

Z

Positive Variable X

Positive Variable Y

Integer Variable C(j,m);


Equations

ObjectiveFunction  For minimizing total earliness & tardiness

ConstraintL1(j)    Linearization constraint for 1st maximum part of objective
function

ConstraintL2(j)    Linearization constraint for 2nd maximum part of objective
function

Constraint1(j)     Completion time of job in position j on machine 1

Constraint2(j)     Completion time of job in position j on machine 2 (1st approach)

Constraint3(j)     Completion time of job in position j on machine 2 (2nd
approach);


ObjectiveFunction .. Z =e= sum(j,X(j)+Y(j));

ConstraintL1(j)   .. X(j) =g= C(j,"2")-d(j);

ConstraintL2(j)   .. Y(j) =g= d(j)-C(j,"2");

Constraint1(j)    .. C(j,"1") =e= sum(k$(ord(k)<=ord(j)),p(k,"1"));

Constraint2(j)    .. C(j,"2") =g= C(j-1,"2")+UI(j,"2")+p(j,"2");

Constraint3(j)    .. C(j,"2") =g= C(j,"1")+UI(j,"2")+p(j,"2");


Model PermutationFlowShop /all/;

Option optca = 0, optcr = 0;

Option limrow = 30;
```

```
Option MIP = BARON;
Solve PermutationFlowShop using MIP minimizing Z;
Display Z.l, C.l;
```

۲-۴ خروجی مدل ساده گمز

```
----      68 VARIABLE Z.L                    =        130.000

----      68 VARIABLE C.L

           1           2

1        2.000      10.000
2       10.000      15.000
3       15.000      24.000
4       20.000      36.000
5       22.000      44.000
6       23.000      50.000
7       25.000      53.000
8       30.000      60.000
9       34.000      63.000
10      40.000      68.000
```

همان‌طور که می‌بینیم با داده‌هایی که به مدل نرم‌افزاری وارد کردیم مجموع زمان‌های دیرکرد و زودکرد برای این ۱۰ کار بر روی ۲ ماشین موردنظر عدد ۱۳۰ واحد زمانی به دست آمد. همچنین در پایین نیز زمان اتمام هر کار (شماره ردیف‌ها) بر روی هر ماشین (شماره ستون‌ها) در قالب متغیری بنام متغیر C آورده شده است که ستون دوم این متغیر همراه با موعدهای مقرر انجام کارها معیار اندازه‌گیری و محاسبه مجموع زمان دیرکرد و زودکرد ما را در تابع هدف تشکیل می‌دهند.

۵- کد نویسی روش حل مدل در گمز

در این بخش فرمولاسیون نرم‌افزاری مدل را از دید یک روش حل دقیق بررسی می‌کنیم. روش حل پیشنهادی الگوریتم شاخه و کران *(Branch & Bound)* می‌باشد که کد آن از ابتدا در دسترس بوده و سعی بر این داریم که مدل خود را به این چارچوب وارد کنیم. الگوریتم شاخه و کران برای راه‌اندازی به جواب‌هایی اولیه نیاز دارد که با توجه به اینکه متغیرهای شاخه زنی اعداد صحیح و مثبت هستند حدود بالا و پایین را به ترتیب ۱۰۰ و ۰ برای تمامی متغیرهای زمان اتمام کار لحاظ نموده‌ایم. متغیرهای دیگر ما متغیرهای پیوسته‌ای هستند که

به‌منظور خطی سازی معادله تابع هدف از آن‌ها استفاده کرده‌ایم، اما در اینجا به روند حل الگوریتم برای شاخه زنی ارتباطی نخواهند داشت. کد روش حل را در ادامه می‌بینیم:

۵-۱ کد روش حل در گمز

```
$Title Permutation Flow Shop Problem (B&B)

$ontext
Branch-and-bound algorithms for minimizing total earliness and tardiness
in a two-machine permutation flow shop with unforced idle allowed
$offtext

Sets
m index for machines /1,2/
j index for jobs /1*10/
Alias (k,j);

Parameters
d(j) Job due dates /1 39,2 37,3 38,4 36,5 39,6 41,7 44,8 45,9 51,10 53/;

Table p(j,m) Processing time of job j on machine m
         1        2
1        2        2
2        8        4.1
3        5        6
4        5        7.2
5        2        8
6        1        4.6
7        2        3
8        5        1.3
9        4        3
10       6        2.4;

Table UI(j,m) Unforced idle time of job j on machine m
         1        2
1        0        6
```

```
2        0       0
3        0       3
4        0       4
5        0       0
6        0       1
7        0       0
8        0       5
9        0       0
10       0       2;

scalar
maxC / 100 /
minC / 0 /
;

Variables
Z
Positive variable X
Positive variable Y
Integer Variable C(j,m);

Equations
ObjectiveFunction  For minimizing total earliness & tardiness
ConstraintL1(j)   Linearization constraint for 1st maximum part of objective function
ConstraintL2(j)   Linearization constraint for 2nd maximum part of objective function
Constraint1(j)     Completion time of job in position j on machine 1
Constraint2(j)     Completion time of job in position j on machine 2 (1st approach)
Constraint3(j)     Completion time of job in position j on machine 2 (2nd approach);

ObjectiveFunction .. Z =e= sum(j,X(j)+Y(j));
ConstraintL1(j)   .. X(j) =g= C(j,"2")-d(j);
ConstraintL2(j)   .. Y(j) =g= d(j)-C(j,"2");
Constraint1(j)    .. C(j,"1") =e= sum(k$(ord(k)<=ord(j)),p(k,"1"));
Constraint2(j)    .. C(j,"2") =g= C(j-1,"2")+UI(j,"2")+p(j,"2");
Constraint3(j)    .. C(j,"2") =g= C(j,"1")+UI(j,"2")+p(j,"2");

model PermutationFlowShop /all/
```

```
set node 'maximum size of the node pool' /node1*node1000/;
parameter bound(node) 'node n will have an obj <= bound(n)';
set fixed(node,j) 'variables C(j,m) are fixed to zero in this node';
set lowerbound(node,j) 'variables C(j,m)>=minC in this node';
scalar bestfound 'lowerbound in B&B tree' /-INF/;
scalar bestpossible 'upperbound in B&B tree' /+INF/;
set newnode(node) 'new node (singleton)';
set waiting(node) 'waiting node list';
set current(node) 'current node (singleton except exceptions)';
parameter log(node,*) 'logging information';
scalar done 'terminate' /0/;
scalar first 'controller for loop';
scalar first2 'controller for loop';
scalar obj 'objective of subproblem';
scalar maxC;
set w(node);
parameter nodenumber(node);
nodenumber(node) = ord(node);
fixed(node,j) = no;
lowerbound(node,j) = no;
set h(j,m);
alias (n,node);
waiting('node1') = yes;
current('node1') = yes;
newnode('node1') = yes;
bound('node1') =INF;
loop(node$(not done),
bestpossible = smax(waiting(n), bound(n));
current(n) = no;
current(waiting(n))$(bound(n) = bestpossible) = yes;
first = 1;
loop(current$first,
first = 0;
log(node,'node') = nodenumber(current);
log(node,'ub') = bestpossible;
waiting(current) = no;
C.lo(j,m) = 0;
```

```
C.up(j,m) = maxC;
h(j,m) = lowerbound(current,j);
C.lo(h) = minC;
h(j,m) = fixed(current,j);
C.up(h) = 0;
Option optca = 0, optcr = 0;
Option MIP = BARON;
solve PermutationFlowShop minimizing z using MIP;
log(node,'solvestat') = PermutationFlowShop.solvestat;
log(node,'modelstat') = PermutationFlowShop.modelstat;
abort$(PermutationFlowShop.solvestat <> 1) "Solver did not return ok";
if (PermutationFlowShop.modelstat = 1 or PermutationFlowShop.modelstat = 2,
obj = z.l;
log(node,'obj') = obj;
maxC = smax((j,m), min(C.l(j,m), max(minC-C.l(j,m),0)));
if (maxC = 0,
log(node,'integer') = 1;
if (obj > bestfound,
log(node,'best') = 1;
bestfound = obj;
w(n) = no; w(waiting) = yes;
waiting(w)$(bound(w) < bestfound) = no;
);
else
h(j,m) = no;
h(j,m)$(min(C.l(j,m), max(minC-C.l(j,m),0))=maxC) = yes;
first2 = 1;
loop(j$first2,
first2 = 0;
newnode(n) = newnode(n-1);
fixed(newnode,j) = fixed(current,j);
lowerbound(newnode,j) = lowerbound(newnode,j);
bound(newnode) = obj;
waiting(newnode) = yes;
fixed(newnode,j) = yes;
newnode(n) = newnode(n-1);
fixed(newnode,j) = fixed(current,j);
```

```
lowerbound(newnode,j) = lowerbound(newnode,j);

bound(newnode) = obj;

waiting(newnode) = yes;

lowerbound(newnode,j) = yes;

);

);

else

abort$(PermutationFlowShop.modelstat <> 4 and PermutationFlowShop.modelstat <> 5)
"Solver did not solve subproblem";

);

log(node,'waiting') = card(waiting);

);

done$(card(waiting) = 0) = 1;

display log,C.l,Z.l;

);
```

۵-۲ خروجی مدل روش حل در گمز

در این بخش خروجی مدل روش حل که همان الگوریتم شاخه و کران باشد را می‌بینیم:

```
----     155 VARIABLE C.L

            1           2

1         2.000      10.000
2        10.000      15.000
3        15.000      24.000
4        20.000      36.000
5        22.000      44.000
6        23.000      50.000
7        25.000      53.000
8        30.000      60.000
9        34.000      63.000
10       40.000      68.000


----     155 VARIABLE Z.L                    =      130.000
```

باکمی دقت متوجه می‌شویم که تمامی جواب‌های به‌دست‌آمده برای الگوریتم شاخه و کران با جواب‌های به‌دست‌آمده در بخش‌های قبلی که مربوط به مدل ساده گمز می‌باشند، برابری می‌کند و تابع هدف در اینجا نیز به عدد ۱۳۰ واحد زمانی رسیده است.

همچنین این نکته را نیز اضافه کنیم که در ششمین معادله از مدل اصلی که زمان اتمام کار با اندیس صفر را بر روی ماشین ۲، صفر در نظر می‌گرفت با توجه به اینکه درمجموع نرم‌افزار گمز مقداری برای این متغیر قائل نمی‌شود (درواقع در دامنه تعریف اندیس کارها قرار نمی‌گیرد) و آن را صفر در نظر می‌گیرد، ما نیز از کد نویسی آن صرف‌نظر کردیم. عبارت زیر در خروجی نرم‌افزار گمز و در بخش *Equation* نیز نشان می‌دهد که مقدار $C(0,2)$ وجود نداشته و یا به‌عبارت‌دیگر صفر است:

```
Constraint2(1)..  C(1,2) =G= 8 ; (LHS = 0, INFES = 8 ****)
```

۶- کد نویسی مدل در نرم‌افزار لینگو

در این قسمت به کمک داده‌های قبلی مدل را این بار به نرم‌افزار لینگو وارد می‌کنیم. در این نرم‌افزار با توجه به اینکه لینگو از خطی سازهای داخلی خود برای خطی کردن عبارات غیرخطی استفاده می‌کند ما نیز این کار را به عهده خود نرم‌افزار می‌گذاریم. همان‌طور که در ادامه و در بخش خروجی خواهیم دید نوع مدلی که لینگو آن را تشخیص می‌دهد *MILP* بوده و به‌صورت پیش‌فرض از الگوریتم شاخه و کران برای حل این مدل استفاده می‌کند و درنهایت به جواب بهینه سراسری می‌رسد. در زیر کد را می‌بینیم:

```
Model:

![Article] Branch-and-bound algorithms for minimizing total earliness and
tardiness in a two-machine permutation flow shop with unforced idle
allowed;

Sets:
      Machines /1 2/;
      Jobs /1..10/: Due_Date;
      AliasJobs /1..10/;
      Link(Jobs,Machines): Processing_Time, Idle_Time, C;
Endsets

Data:
      Due_Date = 39 37 38 36 39 41 44 45 51 53;

      Processing_Time = 2 2
                        8 4.1
                        5 6
                        5 7.2
                        2 8
                        1 4.6
                        2 3
                        5 1.3
                        4 3
                        6 2.4;

      Idle_Time = 0 6
```

```
                 0 0
                 0 3
                 0 4
                 0 0
                 0 1
                 0 0
                 0 5
                 0 0
                 0 2;
Enddata;

Title Permutation Flow Shop;

!Objective Function;
      [Objective_Function] Min = @Sum(Jobs(j):@Smax(C(j,2)-Due_Date(j),0) +
      @Smax(Due_Date(j)-C(j,2),0));

!First Constraint;
      @For(Jobs(j):[First_Constraint] C(j,1) =
      @Sum(AliasJobs(k)|k#LE#j:Processing_Time(k,1)));

!Second Constraint;
      @For(Jobs(j)|j#GT#1:[Second_Constraint] C(j,2) >= C(j-1,2) +
      Idle_Time(j,2) + Processing_Time(j,2));

!Third Constraint;
      @For(Jobs(j):[Third_Constraint] C(j,2) >= C(j,1) + Idle_Time(j,2) +
      Processing_Time(j,2));

!Variables Sign Constraints;
      @For(Link:@GIN(C));

End
```

۶-۱ خروجی مدل در لینگو

```
Global optimal solution found.
  Objective value:                              130.0000
  Objective bound:                              130.0000
  Infeasibilities:                              0.000000
  Extended solver steps:                               0
  Total solver iterations:                             8
  Elapsed runtime seconds:                          0.95

  Model Class:                                      MILP

  Total variables:                   70
  Nonlinear variables:                0
  Integer variables:                 50

  Total constraints:                120
  Nonlinear constraints:              0

  Total nonzeros:                   248
  Nonlinear nonzeros:                 0

  Linearization components added:
```

Constraints: 100
Variables: 60
Integers: 40

Model Title: Permutation Flow Shop

تنها بخشی از خروجی‌ها را می‌آوریم که مربوط به متغیرهای تصمیم‌اند:

Variable	Value	Reduced Cost
C(1, 1)	2.000000	0.000000
C(1, 2)	10.00000	-1.000000
C(2, 1)	10.00000	0.000000
C(2, 2)	15.00000	-1.000000
C(3, 1)	15.00000	0.000000
C(3, 2)	24.00000	-1.000000
C(4, 1)	20.00000	0.000000
C(4, 2)	36.00000	0.000000
C(5, 1)	22.00000	0.000000
C(5, 2)	44.00000	1.000000
C(6, 1)	23.00000	0.000000
C(6, 2)	50.00000	1.000000
C(7, 1)	25.00000	0.000000
C(7, 2)	53.00000	1.000000
C(8, 1)	30.00000	0.000000
C(8, 2)	60.00000	1.000000
C(9, 1)	34.00000	0.000000
C(9, 2)	63.00000	1.000000
C(10, 1)	40.00000	0.000000
C(10, 2)	68.00000	1.000000

همان‌طور که در قسمت‌های قبلی نیز توضیح داده شد، در اینجا نیز مقادیر $C(j,m)$ متغیرهای زمان اتمام هر کار بر روی هر ماشین‌اند و مقدار تابع هدف که مجموع زمانی دیرکرد و زودکرد را محاسبه می‌کند، همانند قبل عدد ۱۳۰ است. درمجموع تمامی نتایج با نتایج به‌دست‌آمده از کدهای قبلی برابری می‌کند.