Az optimális megoldást adó algoritmusok shop ütemezés esetén

Ebben a fejezetben olyan modellekkel foglalkozunk, amelyekben a munkák több műveletből állnak. Speciálisan shop ütemezési problémákat vizsgálunk. Az alábbiakban csak néhány példát mutatunk be, további részletek találhatóak a [3] könyvben és az ott található hivatkozások alapján.

Az *általános shop ütemezési problémát* a következőképpen definiálhatjuk. Adott n munka $j_1, \ldots, j_n$ és m gép $M_1, \ldots, M_m$. Az i-edik munka az O_{ij} $(j = 1, \ldots, n_i)$ műveletekből áll, minden O_{ij} műveletre adott a gép, amelyen végre kell hajtani, és adott a p_{ij} végrehajtási idő. Az azonos munkához tartozó műveletek között előfordulhatnak tetszőleges precedencia relációk. Egy munkát egyszerre csak egy gépen hajthatunk végre (azaz az ugyanazon munkához tartozó műveletek nem hajthatók végre ugyanazon időben párhuzamosan különböző gépeken). Az alábbiakban csak olyan modellekkel foglalkozunk, amelyekben a célfüggvény a maximális befejezési idő.

Diszjunktív gráf modell

A diszjunktív gráf modell alkalmas az általános shop ütemezési problémák bizonyos lehetséges megoldásainak leírására. Reguláris célfüggvények (a maximális befejezési időkben monoton növekvő) esetén, ezen lehetséges megoldások között van optimális is, így ezen célfüggvények mellett ez a reprezentáció alkalmas egy optimális ütemezés meghatározására.

Egy adott általános shop ütemezési problémához rendeljük a következő $G = (V, C, D)$ diszjunktív gráfot.

- V a csúcsok halmaza, minden műveletet egy csúcs reprezentál. Továbbá van két speciális csúcs, egy forrás $O \in V$ és egy nyelő $* \in V$. Minden csúcshoz egy súlyt rendelünk hozzá. A kitüntetett O és $*$ csúcsok súlya 0, a műveleteket reprezentáló csúcsok súlya pedig a megfelelő művelet végrehajtási ideje.

- C a konjuktív élek halmaza. Ezek irányított élek, amelyek a műveletek közötti precedencia relációkat adják meg. Továbbá vezet egy konjuktív él a forrásból minden olyan műveletbe, amelyet nem előz meg egyetlen

művelet sem a precedencia gráfban, és vezet egy konjuktív él minden olyan műveletből a nyelőbe, amelyet nem követ egyetlen művelet sem a precedencia gráfban.

- D a diszjunktív élek halmaza. Ezek irányítatlan élek. Diszjunktív élek vannak azon konjuktív éllel nem összekötött műveletek között, amelyek ugyanahhoz a munkához tartoznak, és azon konjuktív éllel nem összekötött műveletpárok között, amelyeket azonos gépen kell végrehajtanunk.

Az ütemezésben a műveletek sorrendjét meghatározza a diszjunktív élekkel összekötött műveletek sorrendje. Ezt a sorrendet úgy kaphatjuk meg, hogy a diszjunktív éleknek is irányítást adunk. Egy S részhalmazát a diszjunktív éleknek egy irányítással együtt kiválasztásnak nevezzük. Amennyiben minden diszjunktív él irányítást kapott és az így eredményezett $G(S) = (V, C \cup S)$ irányított gráf körmentes, akkor az S kiválasztást teljesnek nevezzük. A teljes kiválasztások megfeleltethetők bizonyos lehetséges ütemezéseknek.

Véve egy S teljes kiválasztást, a következő ütemezést konstruálhatjuk meg. Az ütemezést az egyes műveletek kezdési ideje által adjuk meg. Minden útra amely a $G(S)$ gráfban valamely i csúcsból egy j csúcsba vezet legyen az út hossza az útban szereplő csúcsok súlyainak összege j-t nem számítva. Minden műveletre legyen $l(i)$ a leghosszabb út hossza, amely 0-ból a műveletnek megfeleltetett csúcsba vezet. Egyszerűen látható, hogy az i művelet kezdési idejének az $l(i)$ értéket adva egy lehetséges ütemezést kapunk.

Másrészt minden lehetséges ütemezés meghatározza a műveletek sorrendjét az egyes gépeken. Ez a sorrend egy körmentes irányított gráffal irható le, amely egy teljes kiválasztásnak felel meg. Továbbá az is igazolható, hogy a reguláris célfüggvények mellett a teljes kiválasztásból a fentiek alapján képzett ütemezés költsége nem nagyobb, mint az eredeti ütemezésé.

Következésképp reguláris célfüggvények mellett mindig létezik a diszjunktív gráfnak egy olyan teljes kiválasztása, amely egy optimális ütemezést eredményez. Érdemes megjegyeznünk, hogy amennyiben a célfüggvény a maximális befejezési idő, akkor az egyes ütemezések költsége megegyezik a megfelelő teljes kiválasztásban az $l(*)$ (a maximális forrás-nyelő út hossza) értéknek.

Mindenképpen érdemes megjegyeznünk, hogy a diszjunktív gráf reprezentáció kiválóan használható a korlátozás és szétválasztás elvén alapuló

eljárásoknál. Általában a szétválasztás lépésében a lehetséges megoldások osztályozása egyes diszjunktív élek irányításának megadásával történik. A részletek és a korlátozó függvény kiszámítására használható módszerek ismertetése túlmutat jelen jegyzet keretein.

Open shop problémák

Open shopnak nevezzük azokat a shop problémákat, amelyekben az i-edik munka ($i = 1, \ldots, n$) m műveletből az O_{ij} műveletekből ($j = 1, \ldots, m$) áll, ahol O_{ij}-t a j-edik gépen kell végrehajtani, és amely problémában nincs egyéb precedencia feltétel a műveletekre. Tehát egy open shop probléma esetén definiálnunk kell egy sorrendet az ugyanahhoz a munkához tartozó műveletek között és egy sorrendet az ugyanazon gépen végrehajtandó műveletek között. A legtöbb open shop probléma NP-nehéz, ebben a jegyzetben a legismertebb polinomiális időben megoldható esetet vizsgáljuk, azt amelyben csak két gép van ($O2||C_{\max}$ probléma). A következő lineáris időigényű eljárás megoldja ezt a feladatot.

Legyen $I = \{i|p_{i1} \leq p_{i2}\}$ and $J = \{i|p_{i1} > p_{i2}\}$. Tekintsük a következő két esetet.

1. Eset. $p_{r1} = \max\{\max\{p_{i1}|i \in I\}, \max\{p_{i2}|i \in J\}\}$.

Ebben az esetben a következő sorrendek által megadott ütemezést hajtjuk végre.

- Az M_1 gépen először ütemezzük az $I \setminus \{r\}$ halmazhoz tartozó munkák műveleteit növekvő index szerint, aztán a J-hez tartozó munkákat növekvő index szerint, majd a j_r munkát.

- Az M_2 gépen elsőként a j_r munkát, majd az $I \setminus \{r\}$ halmazhoz tartozó munkákat aztán a J-hez tartozó munkákat ütemezzük ugyanabban a sorrendben, mint az M_1 gépen.

- A munkák műveletei közötti sorrend a következő. Az j_r munkára először az O_{r2} műveletet hajtjuk végre és utána O_{r1}-et, a többi munkát először az M_1 gépen hajtjuk végre és utána az M_2-n.

2. Eset. $p_{r2} = \max\{\max\{p_{i1}|i \in I\}, \max\{p_{i2}|i \in J\}\}$.

Ebben az esetben a fenti ütemezésben felcseréljük a két gép szerepét, és az I és J halmazok szerepét.

Tétel *A fentiekben megadott szabályok egy optimális ütemezést határoznak meg.*

Bizonyítás: Az állítást csak az első esetre igazoljuk, teljesen hasonlóan bizonyítható a másik eset is. A formalizmus egyszerűsítése érdekében tegyük fel, hogy $I = \{1, \ldots, |I| - 1\}$, $J = \{|I|, \ldots, n - 1\}$ és $r = n$. Ezt az általánosítás megszorítása nélkül megtehetjük, hiszen a munkákat jelölhetjük ezen feltételeknek megfelelően.

Vizsgáljuk a fentiekben leírt ütemezés költségét. A költség meghatározásához használjuk a fentiekben bemutatott diszjunktív gráf modellt. A megadott szabályok által meghatározott teljes kiválasztáshoz tartozó irányított gráfban kell meghatároznunk a maximális forrás-nyelő utat. Az open shop problémában nincsennek precedencia relációk a műveletek között, így konjuktív élek csak a forrásból vezetnek az összes műveletbe, és a műveletekből a nyelőbe. A diszjunktív élek irányítását az ütemezéshez tartozó teljes kiválasztás a következőképpen definiálja.

- Minden i-re O_{i1}-ből él vezet O_{j1}-be, ahol $j > i$,
- O_{n2} -ből él vezet O_{i2}-be, ahol $i < n$,
- $i \neq n$ esetén az O_{i2}-ből él vezet O_{j2}-be, ahol $i < j < n$,
- $i \neq n$ esetén O_{i1}-ből él vezet O_{i2}-be, és O_{n2}-ből él vezet O_{n1}-be.

Következésképp a leghosszabb utak a következők lehetnek

- $O, O_{n2}, O_{n1}, *$
- $O, O_{n2}, O_{12}, O_{22}, \ldots, O_{n-1,2}, *$
- $O, O_{11}, O_{21}, O_{31}, \ldots, O_{n1}, *$
- $O, O_{11}, O_{21}, \ldots, O_{i1}, O_{i2}, O_{i+1,2} \ldots, O_{n-1,2}, *$

Az első három út költsége rendre $p_{n1} + p_{n2}$, $\sum_{i=1}^{n} p_{i2}$, $\sum_{i=1}^{n} p_{i1}$. A negyedik típusú út költségének becsléséhez különböztessünk meg két esetet.

Amennyiben $i \in I$, akkor az út hosszára a következő korlátot kapjuk:

$$\sum_{l=1}^{i} p_{l1} + \sum_{l=i}^{n-1} p_{l2} \leq \sum_{l=1}^{i-1} p_{l2} + p_{i1} + \sum_{l=i}^{n-1} p_{l2} \leq \sum_{l=1}^{n} p_{l2},$$

a fentiekben használtuk, hogy $p_{l1} \leq p_{l2}$ ha $l \in I$, és azt, hogy $p_{i1} \leq \max\{p_{j1}|j \in I\} = p_{n1} \leq p_{n2}$.

Amennyiben $i \in J$, akkor az út hosszára a következő korlátot kapjuk:

$$\sum_{l=1}^{i} p_{l1} + \sum_{l=i}^{n-1} p_{l2} \leq \sum_{l=1}^{i} p_{l1} + p_{i2} + \sum_{l=i}^{n-1} p_{l1} \leq \sum_{l=1}^{n} p_{l1},$$

a fentiekben használtuk, hogy $p_{l1} > p_{l2}$ ha $l \in J$, és azt, hogy $p_{i2} \leq \max\{p_{j2}|j \in J\} \leq p_{n1}$.

Következésképp azt kaptuk, hogy a leghosszabb forrás nyelő út hossza, azaz az ütemezés költsége $\max\{p_{n1}+p_{n2}, \sum_{i=1}^{n} p_{i2}, \sum_{i=1}^{n} p_{i1}\}$. Mászrészt mivel egy gép csak egy munkát végezhet egyidejűleg és ugyanazon munka műveletei nem hajthatók végre párhuzamosan, ezért a fenti maximumban szereplő kifejezések mindegyike alsó korlátja az optimális maximális befejezési időnek, amivel igazoltuk, hogy az algoritmus valóban optimális ütemezést ad. □

Flow shop problémák

Flow shopnak nevezzük azokat a shop problémákat, amelyekben az i-edik munka ($i = 1, \ldots, n$) m műveletből az O_{ij} műveletekből ($j = 1, \ldots, m$) áll, ahol O_{ij}-t az M_j gépen kell végrehajtani, továbbá az $O_{ij} \rightarrow O_{i,j+1}$ precedencia feltételeink vannak. A precedencia feltételek azt adják meg, hogy a munkák műveleteit adott sorrendben kell végrehajtanunk, először az első gépen aztán a másodikon és így folytatva. Tehát ebben az esetben ellentétben az open shop problémával a munkáknak egy π_j sorrendjét kell meghatároznunk minden M_j-re.

Érdemes külön foglalkoznunk, azon ütemezésekkel, amelyeknél ezek a π_j sorrendek megegyeznek, minden gépre. A problémát ezen megszorítás mellett permutációs flow shop problémának nevezzük. Az alábbiakban ismertetjük Johnson algoritmusát, amely megoldja két gép esetén a permutációs flow shop problémát.

Johnson algoritmusa ([2])

- *1. lépés.* Legyen $k := 1$ és $l := n$, és tekintsük a nem ütemezett munkák $J = \{j_1, \ldots, j_n\}$ halmazát.

- *2. lépés.* Keressük meg a $\{p_{i1}, p_{i2} | j_i \in J\}$ halmaz egy minimális elemét. Jelölje a hozzátartozó indexet i. Ha i nem egyértelműen meghatározott, akkor a lehetséges indexek közül válasszuk a legkisebbet.

- *3. lépés.* Ha a 2. lépésben választott elem p_{i1} akkor helyezzük a j_i munkát a listánk k-adik helyére, töröljük a J halmazból, és növeljük k értékét 1-gyel. Majd lépjünk az 5. lépésre.

- *4. lépés.* Ha a 2. lépésben választott elem p_{i2} akkor helyezzük a j_i munkát a listánk l-adik helyére, töröljük a J halmazból, és csökkentsük l értékét 1-gyel. Majd lépjünk az 5. lépésre.

- *5. lépés.* Amennyiben J üres, akkor vége az eljárásnak. Az optimális ütemezést kapjuk meg, ha az eljárás során meghatározott lista sorrendje szerint hajtjuk végre a munkákat késlekedés nélkül. Ellenkező esetben lépjünk a 2. lépésre.

Az eljárás helyessége egyből következik a következő segédtételből.

Segédtétel. *Ha egy permutációs flow shop probléma egy késleltetést nem tartalmazó S ütemezésében egy j_i munkára és az ütemezésben utána közvetlenül következő j_l munkára*

$$\min\{p_{i1}, p_{i2}, p_{l1}, p_{l2}\} = \min\{p_{l1}, p_{i2}\}$$

teljesül, akkor arra az S' ütemezésre, amelyet úgy kapunk S -ből, hogy felcseréljük az j_i és j_l munkákat a maximális befejezési idő legfeljebb akkora lesz, mint az S ütemezésben.

Bizonyítás. Elsőként tegyük fel, hogy a segédtételben szereplő értékek közül p_{l1} minimális. Az az eset, amelyben p_{i2} minimális teljesen hasonlóan igazolható. Jelölje Q azt az időt, amikor S-ben M_1 elkezdheti a j_i munka ütemezését (a j_i-t megelőző munka első műveletének befejezési ideje), és R pedig a j_i - t megelőző munka második műveletének befejezési idejét.

Vegyük észre, hogy mivel S és S' csak a j_i és j_l munkák sorrendjében különbözik, ezért az állítás igazolásához elegendő belátni, hogy $C_l \geq C'_i$, ahol C_l a j_l munka befejezési ideje az S ütemezésben C'_i pedig a j_i munka befejezési ideje az S' ütemezésben.

Vizsgáljuk elsőként a C_l értéket. Az R és Q időpontok definíciója pontosan azt jelenti, hogy S-ben a j_i munka O_{i2} műveletének végrehajtása nem kezdődhet a $\max\{R, Q + p_{i1}\}$ időpont előtt. Másrészt $p_{l1} \leq p_{i2}$, tehát az O_{l1} műveletet hamarabb befejeződik az M_1 gépen, mint az O_{i2} művelet az M_2 gépen, így

$$C_l = \max\{R, Q + p_{i1}\} + p_{i2} + p_{l2}.$$

Most tekintsük a C'_i értéket. Az S' ütemezésben az O_{l2} művelet M_2-n a $\max\{R, Q+p_{l1}\}$ időpontban kezdődik, és az O_{i2} művelet pedig akkor, amikor O_{l2} és O_{i1} egyaránt befejeződött. Tehát

$$C'_i = \max\{\max\{R, Q + p_{l1}\} + p_{l2}, Q + p_{l1} + p_{i1}\} + p_{i2},$$

azaz

$$C'_i = \max\{\max\{R, Q + p_{l1}\} + p_{l2} + p_{i2}, Q + p_{l1} + p_{i1} + p_{i2}\}.$$

Másrészt $C_l \geq \max\{R, Q + p_{l1}\} + p_{l2} + p_{i2}$, mivel $p_{l1} \leq p_{i1}$. Továbbá $C_l \geq Q + p_{l1} + p_{i1} + p_{i2}$, mivel $p_{l1} \leq p_{l2}$.

Következésképp azt kaptuk, hogy C_l nem kisebb a C'_i kifejezésében szereplő maximum egyik tagjánál sem, így $C'_i \leq C_l$, amivel a segédtételt igazoltuk. □

Mindenképpen érdemes megjegyeznünk, hogy az alábbi segédtétel alapján következik, hogy két gép esetén nem megszorítás, pusztán olyan ütemezéseket vizsgálni, amelyekben a π_j munkasorrendek megegyeznek minden gépre.

Segédtétel *Az $F2||C_{max}$ probléma esetén van olyan optimális ütemezés, amelyben a munkák sorrendje mindkét gépen ugyanaz.*

Bizonyítás: Tekintsük az optimális ütemezéseket. Minden optimális ütemezésre van olyan $0 \leq k \leq n$, hogy az első k munkának megegyezik a sorrendje a két gépen. Vegyünk egy olyan ütemezést, amelyre ez a k maximális. Tegyük fel, hogy erre az ütemezésre $k < n$, legyen i a k-adik munka és j az a munka, amelyet az M_2 gépen közvetlenül az i munka második művelete után hajtunk végre.

Könnyen adódik, hogy amennyiben az első gépen j első műveletét közvetlenül i első művelete után hajtjuk végre, és a többi i után következő művelet kezdési idejét p_{1j}-vel növeljük, akkor egy olyan ütemezést kapunk, amelyben a maximális befejezési idő nem nagyobb mint az eredeti

ütemezésben, így szintén optimális. Másrészt ebben az új ütemezésben, már az első $k+1$ munka sorrendje azonos a két gépen, ami ellentmond k maximalitásának. Következésképp a választott ütemezésben $k = n$, amivel igazoltuk a segédtétel állítását. □

Job shop problémák

A job shop probléma egy olyan shop probléma, amely a flow shop probléma egy általánosítása. Ebben az esetben a j_i munka n_i műveletet tartalmaz, az $O_{i1}, O_{i2}, \ldots, O_{in_i}$. Miként a flow shop probléma esetében is, a műveletek között a $O_{ij} \rightarrow O_{i,j+1}$ precedencia relációk vannak. A műveletekhez meg van adva, hogy mely gépeken kell őket végrehajtani, (az O_{ij} nem feltétlenül az M_j gépen), egy munkához nem tartozik két különböző művelet, amelyeket ugyanazon a gépen kell végrehajtani.

Az alábbiakban bemutatjuk azt az algoritmust, amelyet arra a speciális job shop problémára fejlesztettek ki, amelyben csak két gép van. Az eljárás a Johnson algoritmus általánosítása.

Jackson algoritmus ([1])

- *1. lépés.* Képezzük a következő halmazokat:

  $K_{12} = \{$ azon két műveletből álló munkák, amelyek műveleteiből az elsőt az M_1 a másodikat az M_2 gépen kell végrehajtani $\}$,

  $K_{21} = \{$ azon két műveletből álló munkák, amelyek műveleteiből az elsőt az M_2 a másodikat az M_1 gépen kell végrehajtani $\}$,

  $K_1 = \{$ azon munkák, amelyek egy, az M_1-en végrehajtandó műveletből állnak $\}$,

  $K_2 = \{$ azon munkák, amelyek egy, az M_2-en végrehajtandó műveletből állnak $\}$.

- *2. lépés.* Rendezzük a K_{12} és a K_{21} halmazokat a Johnson eljárás szerint.

- *3. lépés.* Rendezzük a $K_{12} \cup K_1 \cup K_{21}$ halmaz elemeit úgy, hogy K_1-ben tetszőleges legyen a rendezés, és K_{12} legnagyobb eleme kisebb legyen

mint K_1 legkisebb eleme, továbbá K_1 legnagyobb eleme kisebb legyen, mint K_{21} legkisebb eleme. Jelölje ezt a rendezést $(K_1', \prec)$.

- *4. lépés.* Rendezzük a $K_{21} \cup K_2 \cup K_{12}$ halmaz elemeit úgy, hogy K_2-ben tetszőleges legyen a rendezés, és K_{21} legnagyobb eleme kisebb legyen mint K_2 legkisebb eleme, továbbá K_2 legnagyobb eleme kisebb legyen, mint K_{12} legkisebb eleme. Jelölje ezt a rendezést $(K_2', \prec)$.

- *5. lépés.* Az M_1 gépen ütemezzük a munkákat $(K_1', \prec)$ szerint, az M_2-n pedig $(K_2', \prec)$ szerint.

Tétel*Az algoritmus valóban egy optimális megoldását adja meg a job shop problémának két gép esetén.*

Bizonyítás: Elsőként tegyük fel, hogy $\sum_{i \in K_{21}} p_{i2} \leq \sum_{i \in K_{12}} p_{i1} + \sum_{i \in K_1} p_{i1}$. Ebben az esetben az M_1 gépen nincs üres idő. Amennyiben az M_2 gépen sincs üres idő vagy a maximális befejezési idő $\sum_{i \in K_{12} \cup K_1 \cup K_{21}} p_{i1}$, akkor az ütemezés nyilvánvalóan optimális. Ellenkező esetben, a maximális befejezési idő pontosan megegyezik a K_{12} halmazra megszorított probléma optimális ütemezésében a befejezési idővel, azaz ekkor is optimális megoldást kaptunk. Amennyiben $\sum_{i \in K_{21}} p_{i2} > \sum_{i \in K_{12}} p_{i1} + \sum_{i \in K_1} p_{i1}$, akkor az M_2 gépen nincs üres idő, és az algoritmus által kapott ütemezés optimalitása az előző esethez teljesen hasonlóan látható. □

References

[1] Jackson, J. R., An extension of Johnson's result on job lot scheduling, *Naval Research Logistics Quarterely* **3** 1956, 201-203.

[2] Johnson, Optimal two- and three-stage production schedules with setup times included, *Naval Research Logistics Quarterely* **1** 1954, 61-68.

[3] M. Pinedo, *Scheduling, Theory, Algorithms and Systems*, Prentice Hall, New Jersey, 1995.