Egyszerű megoldást eredményező elvek

Az első fontos észrevételünk az ütemezési problémákkal kapcsolatban az, hogy reguláris célfüggvények esetén (regulárisnak nevezzük azokat a célfüggvényeket, amelyek monoton növekvőek a befejezési időkben) elegendő azon ütemezéseket vizsgálnunk, amelyekben nincs üres idő, azaz amelyekben a gépek megszakítás nélkül dolgoznak. Ezt a nyilvánvaló állítást sokszor fogjuk használni az ütemezési problémák vizsgálata során.

Legrövidebb végrehajtási idő elve

Első példaként a $1||\sum w_jC_j$ problémát tekintjük (egyetlen gépen ütemezzük a munkákat és célunk a befejezési idők súlyozott összegét minimalizálni). Ezen célfüggvény esetén a w_j/p_j hányados tekinthető a j-edik munka egységenkénti fontosságának, így természetes gondolat azon munkákat ütemezni először, amelyekre ez a hányados nagyobb. Ezt az elvet „súlyozott legrövidebb végrehajtási idő" elvének nevezzük (a WSPT rövidítést használjuk az angol "weighted shortest processing time" kifejezés alapján). Az algoritmust, amely ezen az elven alapul a [2] dolgozatban mutatták be. Miként a következő állítás mutatja ez a megközelítés valóban optimális a vizsgált probléma esetén.

Tétel *A WSPT elv optimális ütemezést ad az* $1||\sum w_jC_j$ *probléma esetén.*

Bizonyítás: Elsőként igazoljuk, hogy csak a WSPT elvet követő algoritmusok adhatnak optimális ütemezést. Ehhez tegyük fel, hogy van egy olyan optimális S ütemezés, amelyre nem teljesül a WSPT elv. Mivel nem teljesül az elv, ezért van az S ütemezésben két egymást követő munka, J_i és közvetlenül utána J_k, amelyekre

$$\frac{w_i}{p_i} < \frac{w_k}{p_k}.$$

Jelöljük t-vel a J_i munka kezdési idejét az S ütemezésben. Tekintsük azt az S' ütemezést, amelyben a J_i és a J_k munkákat felcseréljük. Az S ütemezésben J_i befejezési ideje $t+p_i$, J_k befejezési ideje pedig $t+p_i+p_k$. Az S' ütemezésben J_k befejezési ideje $t+p_k$ és J_i befejezési ideje $t+p_i+p_k$. A többi munka befejezési ideje ugyanaz mindkét ütemezésben. Jelölje a többi munkára számított $\sum w_jC_j$ összeget Q. Ekkor az S ütemezés teljes költsége

$Q + w_i(t + p_i) + w_k(t + p_i + p_k)$, az S' ütemezés teljes költsége $Q + w_k(t + p_k) + w_i(t + p_i + p_k)$.

Másrészt $\frac{w_i}{p_i} < \frac{w_k}{p_k}$, így $w_i p_k < w_k p_i$, amiből azt kapjuk, hogy az S' ütemezés költsége kisebb az S ütemezés költségénél. Ez ellentmond S optimalitásának, amivel az állítását igazoltuk.

Másrészt a fentiekhez teljesen hasonlóan látható, hogy amennyiben két egymást követő munkára egyenlő a w/p hányados, akkor a felcserélésükkel kapott ütemezésre a célfüggvény értéke nem változik. Mivel a feladatnak véges sok lehetséges megoldása van, így létezik optimális megoldás, ezért a fentiek alapján következik a tétel állítása. □

Amennyiben a problémában az egyes munkáknak nincsenek súlyai, akkor a WSPT elv az SPT elvre redukálódik, amely szerint mindig a legkisebb végrehajtási idővel rendelkező munkát ütemezzük először. A fenti tétel alapján következik, hogy az SPT elv optimális az $1||\sum C_j$ probléma esetén. Az alábbiakban adunk egy újabb bizonyítást erre az állításra, amely bizonyítás előnye, hogy kiterjeszthető, a párhuzamos gépek esetére is.

Tétel *Az SPT elv optimális ütemezést ad az* $1||\sum C_j$ *probléma esetén.*

Bizonyítás: Tekintsünk egy gépet, jelölje a gépen ütemezett munkákat növekvő kezdési idő szerint $p_1, p_2, \ldots, p_n$. Ekkor ezen gépre a $\sum C_j$ összeg $np_1 + (n-1)p_2 + \cdots + p_n$. Tehát feladatunk az, hogy az $1, \ldots, n$ együtthatókat hozzárendeljük a $p_1, \ldots, p_n$ értékekhez oly módon, hogy a fenti összeg minimális legyen. Könnyen igazolható, hogy az együtthatók és a p_i értékek azon párosítására lesz minimális az összeg, amelyben az együtthatók és a p_i értékek ellentétesen vannak rendezve (a legnagyobb együtthatóhoz rendeljük a legkisebb értéket, a második legnagyobbhoz a második legkisebb értéket és így folytatjuk). Másrészt ez a hozzárendelés pontosan az SPT ütemezési elvet írja le. □

Most tekintsük a párhuzamos gépek esetét. Ebben az esetben az SPT elv következő kiterjesztése lesz a megfelelő ütemezési stratégia. Tegyük fel, hogy n/m egész. Ez elérhető, ha a munkák sorozatát kiegészítjük fiktív 0 végrehajtási idejű munkákkal. Az MSPT (módosított SPT) elv sorbarendezi a munkákat növekvő végrehajtási idő szerint, ezt követően az első m munka lesz az m darab gépen elsőként végrehajtva, a második m munka másodikként és így folytatva az utolsó m munka n/m-edikként.

Tétel *Az MSPT elv optimális ütemezést ad a* $Pm||\sum C_j$ *probléma esetén.*

Bizonyítás: Vegyünk m gépet. Legyenek az i-edik gépen ($i = 1, \ldots, m$)

ütemezett munkák $p_{i1}, p_{i2}, \ldots, p_{ik_i}$. Ekkor $\sum_{i=1}^{m} k_i = n$ és az ütemezésre a $\sum C_j$ összeg $\sum_{i=1}^{m} \sum_{j=1}^{k_i} (k_i + 1 - j) p_{ij}$. A fenti összegben adott számokhoz (a p_{ij} végrehajtási idők) rendelünk hozzá együtthatókat. Összesen n pozitív egész együtthatónk van, amelyekből minden l pozitív egészre legfeljebb m együttható veheti fel az l értéket. Mivel a végrehajtási idők nemnegatívak, ezért könnyen látható, hogy a teljes befejezési időt megadó összeg minimális, ha az együtthatók között minden $i = 1, 2, \ldots, m/n$ értékre m darab veszi fel az i értéket, és a végrehajtási idők és az együtthatók ellentétesen vannak rendezve. Másrészt ez pontosan az MSPT ütemezési elv által kapott ütemezésnek felel meg. □

Legkorábbi határidő elve

Amennyiben a munkák paraméterei között határidők is szerepelnek, és a cél valamely határidőtől függő függvény minimalizálása, akkor gyakran használt elv, amely azon munkákat ütemezi elsőként amelyeknek a legkorábbi a határideje ezt az elvet EDD (earliest due date) jelöli. Mielőtt példát mutatnánk olyan problémára, amelynél az EDD elv optimális ütemezést ad, egy általános algoritmus mutatunk be, amely speciális esetben az EDD elvre redukálódik. Ezt az általános algoritmust az [1] dolgozatban mutatták be.

Az algoritmus több célfüggvény esetén is használható. Legyen a j-edik munkára h_j egy monoton növekvő függvény, legyen $h_{\max} = \max\{h_1(C_1), h_2(C_2), \ldots, h_n(C_n)\}$. Ezen célfüggvény speciális esetként tartalmazza a $\max L_j$, és $\max T_j$ függvényeket, mivel mind L_j mind T_j monoton növekvő függvényei a befejezési időnek. A vizsgált modellben megengedjük precedencia feltételek megadását is. Tehát a probléma, amit vizsgálunk az $1|\text{prec}|h_{max}$ jelöléssel írható le.

Az alábbiakban bemutatott algoritmus hátulról, az utolsónak vgrehajtott munka felől építi fel az ütemezést. J mindig a már elhelyezett munkák halmaza, J^C azon munkák halmaza, amelyeket még el kell helyeznünk (a J halmaz komplementere) és J' azon J^C-beli munkák halmaza, amelyek az adott lépésben ütemezhetők, azaz azon munkáké, amelyekre az összes precedencia feltételek szerinti rákövetkező munkát már elhelyeztük az ütemezés végén.

Algoritmus

- *1. Lépés:* Legyen $J = \emptyset$, $J^C = \{1, \ldots, n\}$, és J' azon munkák halmaza, amelyekre nincs rákövetkező munka a precedencia gráfban.

- *2. Lépés:* Legyen $j^* \in J'$ olyan munka, amelyre

$$h_{j^*}(\sum_{j \in J^C} p_j) = \min_{i \in J'}(h_i(\sum_{j \in J^C} p_j)).$$

  Egészítsük ki J-t j^*-al, töröljük j^*-ot a J^C halmazból. Legyen J' az új halmaza az ütemezhető munkáknak.

- *3. Lépés:* Ha $J^C = \emptyset$ az algoritmus végetért, egyébként folytassuk a 2. lépéssel.

Tétel *Az algoritmus a munkák egy optimális ütemezését adja az* $1|\text{prec}|h_{\max}$ *problémára.*

Bizonyítás: Az állítást indirekt igazoljuk. Ehhez tegyük fel, hogy a kapott ütemezés nem optimális. Jelölje S az algoritmus által adott ütemezést. Tekintsük azt az optimális ütemezést, amelyhez olyan sorrendje tartozik a munkáknak, amely sorrendnek a legtöbb utolsó közös munkája van S-el. Jelölje ezt OPT. Mivel a tekintett ütemezés nem S, ezért ez az ütemezés, valamely iterációban S-től különböző munkát választ. Tegyük fel, hogy ebben a lépésben OPT a k munkát választja (ekkor a $k \in J'$ feltételnek kell teljesülnie az adott lépésben). Másrészt S egy olyan j^* munkát választ, amely minimalizálja a $h_i(\sum_{j \in J^C} p_j)$ értéket az $i \in J'$ halmazon. Ez azt jelenti, hogy OPT-ban j^* a k munka előtt kerül ütemezésre.

Vegyük azt a sorrendet, amelyet úgy kapunk OPT-ból, hogy j^*-ot a jelenlegi helye helyett közvetlenül k után ütemezzük, a többi munka sorrendjét változatlanul hagyva. Azokat munkákat, amelyek az eredeti ütemezésben j^* és k között voltak továbbá k-t hamarabb fejezzük be. Egyedül a j^* munka befejezési ideje nő, de ezen munka minimalizálta a $h_i(\sum_{j \in J^C} p_j)$ értéket az $i \in J'$ halmazon, így az általa okozott költség nem nagyobb, mint k költsége az eredeti ütemezésben. Következésképp a j^* elemet beillesztve a k elem mögé, egy olyan ütemezési sorrendet kapunk, amelynek a költsége nem nagyobb egy optimális ütemezésnél, így szintén optimális. Másrészt az új ütemezés végén eggyel több munka egyezik meg az S ütemezés utolsó munkáival, amivel ellentmondáshoz jutunk. □

Az $1||L_{\max}$ probléma a legismertebb speciális esete a $1|\text{prec}|h_{\max}$ problémának. Ebben az esetben $h_j = C_j - d_j$, az algoritmus pedig a korábbi határidő, röviditve EDD (earlyest due date) szabályra redukálódik, amely

azt mondja ki, hogy azt a munkát ütemezzük hamarabb, amelynek hamarabb van a határideje.

Hivatkozások

[1] E. L. Lawler, Optimal sequencing of a single machine subject to precedence constraints, *Management Science*, **19**, 1973, 544-546.

[2] W. E. Smith, Various optimizers for single-stage production, *Naval Research Logistics Quarterly*, **3**, 1956, 59-66.