

TDK-DOLGOZAT

Szoldatics András, Ősz Olivér



CIKLIKUS ÜTEMEZÉS S-GRÁFFAL

SZOLDATICS ANDRÁS
Programtervező informatikus
BSc, 6. félév

Pannon Egyetem
Műszaki Informatikai Kar

ÓSZ OLIVÉR
Programtervező informatikus
BSc, 3. félév

Pannon Egyetem
Műszaki Informatikai Kar

Témavezető:
Hegyháti Máté
PhD hallgató, PE MIK

Tartalomjegyzék

1. Bevezetés	3
2. Szakirodalmi áttekintés	5
2.1. Ütemezési feladatok	5
2.2. Matematikai programozási módszerek	8
2.2.1. Gyártórendszerek ütemezési modellei	8
2.2.2. Ciklikus ütemezési modellek	10
2.3. S-gráf módszertan	11
3. Ciklikus ütemezés S-gráffal	14
3.1. Ciklusidő minimalizálása	14
3.2. Profit maximalizálása	15
3.3. Időegységenkénti profit maximalizálása	16
3.3.1. Első gyorsító módszer	18
3.3.2. Második gyorsító módszer	19
4. Teszteredmények	21
4.1. Ciklusidő minimalizálás	21
4.2. Optimális profit/ciklusidő meghatározása	22
5. Összegzés	26
A. Részletes teszteredmények	30

1. fejezet

Bevezetés

Az informatika rohamos fejlődése egyre nagyobb és bonyolultabb feladatok megoldását tette lehetővé az ipari gyártórendszerek optimalizálásában és egyéb területeken. Amikor egyes folyamatok azonos, korlátozott mennyiségű erőforrásokért versengenek, az optimális működéshez elengedhetetlen azok megfelelő ütemezése.

Az ütemezési feladatok egyik osztályozási szempontja az ütemezendő időtartam hossza. Leggyakrabban rövidtávú (legfeljebb néhány nap) ütemezésekkel foglalkozik a szakirodalom. A középtávú ütemezés 1-2 hónapra tervez, végül a hosszútávú akár egy egész évre is kiterjedhet. Ilyen nagy időtartamra csak stabil piaci igények mellett célszerű tervezni, ahol nem változik a termékpaletta és a termelési stratégia vagy a változások előre jól becsülhetőek.

Rövidtávú ütemezésre számos módszert publikáltak a szakirodalomban. Az implementált modellek elméletileg használhatóak hosszabb időtartamra is, viszont a sok döntési lehetőség miatt mai számítási sebességgel nem oldhatóak meg elfogadható idő alatt. A probléma kezelhető méretűvé redukálásának egy lehetősége új korlátozások alkalmazása. Egy elterjedt módszer a Shah és társai[1] által kidolgozott periodikus vagy másnéven ciklikus ütemezés. Az eredeti problémát úgy módosították, hogy egy rövidtávú ütemezési feladat optimális megoldását egy ismétlődő ciklusnak tekintik. Így néhány korlátozás megváltoztatásával használhatóak az ismert rövidtávú ütemezési módszerek. A megoldáshoz kulcsfontosságú a ciklus hosszának megfelelő megválasztása. Abban az esetben, ha a ciklusidő a teljes ütemezendő időintervallum hossza, a feladat megegyezik az eredeti hosszútávú problémával.

Az ütemezési feladatokat célfüggvény szerint is megkülönböztetjük. Rövidtávú tervezés esetén a leggyakoribb az adott mennyiségű termék gyártási idejének, más néven makespan-jének minimalizálása. Ezt ciklikus ütemezéssel nehezebb lenne megoldani, de hosszútávú ütemezésnél általában nem is ez a cél. A feladat az egységnyi időre jutó profit maximalizálása.

A vegyipari gyártórendszerek tervezése során számos különböző ütemezési feladat merül fel. Az előforduló esetek széles körét a 2.1 alfejezetben mutatjuk be. Ezt követően a szakirodalomban fellelhető megoldási módszereket ismertetjük. Köztük kevert egészértékű lineáris programozási (Mixed Integer Linear Programming - MILP) módszereket, ciklikus ütemezési modelleket és a munkánk alapját képező S-gráf módszertant. A harmadik fejezetben bemutatjuk az általunk kifejlesztett algoritmus működését. Empirikus vizsgálataink eredményeit a negyedik fejezet tartalmazza, melyben szakirodalmi példák felhasználásával teszteltük módszerünket.

2. fejezet

Szakirodalmi áttekintés

2.1. Ütemezési feladatok

A vegyiparban használatos gyártórendszerek között kétféle működési elvet különböztetünk meg. Munkánk során szakaszos üzemű rendszerekkel foglalkozunk, melyekben a gyártási folyamatokat végző berendezések három lépésben működnek. Az első lépésben előre meghatározott mennyiségű és összetételű anyagokkal töltik meg. A másodikban lejátszódik a gyártási lépés, úgynevezett taszk, amely lehet egy reakció, hűtés, keverés vagy más folyamat. Végül a keletkezett anyagot eltávolítják a berendezésből, melyet később felhasználhat egy másik taszk vagy végtermékként kikerülhet a rendszerből. Léteznek folyamatos üzemű rendszerek is, ahol az anyagok egy meghatározott mértékű sebességgel folyamatosan áramlanak be a berendezésekbe, a folyamat végén pedig a termék folyamatosan áramlik ki. Egyes kevert rendszerekben mindkét működési elvű berendezés megtalálható.

A gyártórendszerek felépítésük szerint is csoportosíthatók:

Szekvenciális felépítésű A taszkok sorban követik egymást egy termék gyártási folyamatában

Egylépcsős Minden termék legyártása egy taszkból áll

Többlépcsős A több lépésből álló gyártási folyamatokat a szakirodalomban két csoportba sorolják:

Multiproduct Minden termék ugyanazon a lépéseken megy keresztül

Multipurpose Az egyes termékek taszkjainak halmaza és sorrendje eltérő lehet

Hálózat topológiájú A folyamatok közti kapcsolatok egy bonyolultabb hálózatot hoznak létre

Az egyes rendszerek pontos felépítését és a termékek legyártásának folyamatát a **recept** írja le. Az ütemezés szempontjából a legfontosabb információk a berendezések, taszkok valamint a köztük áramló anyagok kapcsolatát. A receptnek tartalmaznia kell az elvégzendő taszkok sorrendjét, és azt hogy mely berendezésekben milyen időtartam alatt hajthatók végre. Ha az ütemezési feladat során további tényezőket is figyelembe kell venni, mint például az anyagok berendezések közti mozgatásához szükséges időt (transfer time), akkor ezek az információk is a recept részeit képezik.

A hálózat topológiájú gyártórendszerek működését az úgynevezett általános célú recept írja le. A szekvenciális rendszerekkel szemben a folyamatoknak több bemenete és kimenete is megengedett. Az ilyen bonyolult rendszerek ábrázolására használható a Resource-Task Network (RTN)[2] és a State-Task Network (STN)[3] reprezentáció. Mindkét módszer irányított páros gráfon szemlélteti a gyártási folyamat működését. Az STN grájában körökkel jelölik a különböző állapotú anyagokat és téglalapokkal a taszkokat, a berendezések nincsenek feltüntetve. Az élek egy taszk során felhasznált anyagokból a taszkra mutatnak, a taszkból pedig a legyártott anyagokra. RTN-ben a berendezéseket is ugyanolyan erőforrásnak tekintik, mint a köztes anyagokat, így azokat is körökkel vagy ellipsziszekkel jelölik a gráfban és a bennük elvégezhető taszkokkal kétirányú élek kötik össze. Minden taszkból annyi példányt kell létrehozni, amennyi berendezésben elvégezhető, különben a több bemenet együttes berendezés-használatot jelentene.

Gyártórendszerek ütemezése során gyakran a köztes anyagok tárolását is figyelembe kell venni. Két taszk elvégzése között hosszú idő telhet el, mialatt valahol tárolni kell az anyagot. Tárolókapacitás szerint a következő eseteket különböztetjük meg:

Unlimited Intermediate Storage (UIS)

Az ütemezés szempontjából végtelen kapacitásúnak tekinthető

Finite Intermediate Storage (FIS)

Az adott anyag tárolására véges méretű tároló áll rendelkezésre

Common Intermediate Storage (CIS)

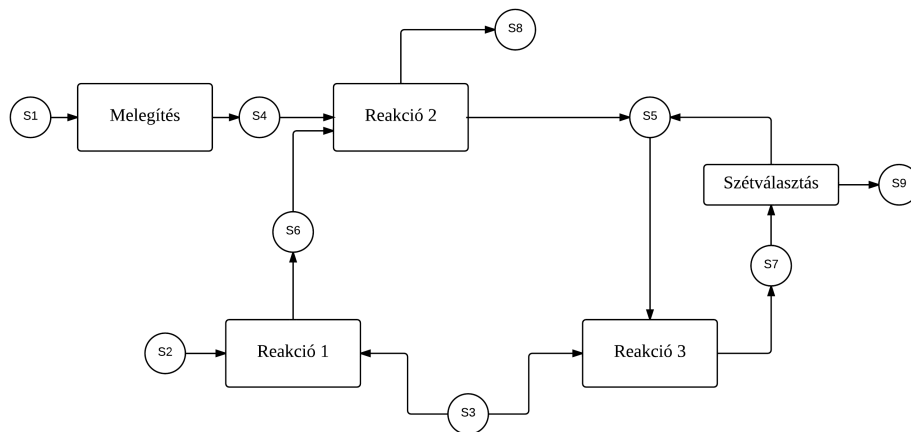
Egy tárolóegység többféle anyag tárolására használható, feladattól függetlenül egymás után vagy akár párhuzamosan

No Intermediate Storage (NIS)

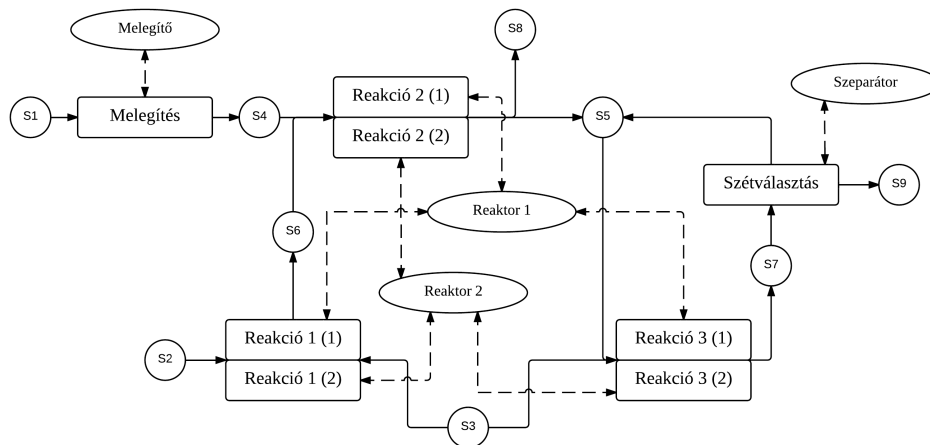
Nincs köztes tároló, egy taszk véget érése után a berendezés szolgál tartályként, amíg az anyag át nem kerül egy másikba

Mixed Intermediate Storage (MIS)

2.1. ábra. Egy példa STN reprezentációra



2.2. ábra. Az előbbi rendszer RTN reprezentációja



A különböző anyagokra eltérő tárolási korlátozások vonatkoznak az előbbiekből

A felsorolt kapacitási korlátozások időbeli megkötésekkel[4] párosíthatók:

Unlimited Wait (UW)

Nincs megszabva, hogy mennyi ideig tárolható az anyag (pl. szilárd, kémiaiilag stabil anyagok))

Limited Wait (LW)

Az anyag egy adott, véges időtartamig tárolható (pl. felmelegítés utáni állapotban)

Zero Wait (ZW)

Az anyag olyan instabil, hogy elkészülte után azonnal el kell kezdeni a következő taszkját

Az ütemezésnek számos célja lehet, amelyeket a célfüggvény ad meg. A makespan minimalizálásakor az termékek előírt mennyiségének legyártásához szükséges időt kell csökkenteni. A gyártás az utolsó termék elkészültekor ér véget, ezért itt nem számít, hogy a többi termék mennyivel előbb készül el, ellentétben olyan feladatokkal, ahol a befejezési idők összegét kell minimalizálni. A termékeknek határidőt is meg lehet adni, ami további lehetőségeket nyújt a célfüggvény megválasztására, pl. késések és korai befejezések miatti eltérések minimalizálása. Főleg hosszútávú feladatoknál szokásos árakat rendelni a termékekhez, majd annyit legyártani, olyan összeállításban, ami adott idő alatt maximális bevételt eredményez.

2.2. Matematikai programozási módszerek

A szakirodalomban az egyik legelterjedtebb kutatási irány az ütemezési feladatok MILP-pel történő megfogalmazása. A felírt modellek megoldására általános célú megoldókat használnak. Az elmúlt években számos ilyen módszer publikáltak makespan minimalizálásra és ciklikus ütemezésre egyaránt.

2.2.1. Gyártórendszerek ütemezési modellei

A szakirodalomban publikált ütemezési módszerek különböző időábrázolást alkalmaznak modelleikben. Általában valamilyen időpontokat határoznak meg, ahol a taszkok elkezdődhetnek vagy befejeződhetnek. Ezeket diszkrét időpontoknak[3] nevezzük, ha azok egyenlő távolságra vannak egymástól, és

folytonosnak, ha változó távolságra[5, 6]. Az időábrázolás lehet globális[7], azaz minden berendezésre egyaránt vonatkozó, vagy berendezés-specifikus[8]. Az időpontok, taszkok és berendezések hozzárendelése bináris változókkal történik. A modellek különféle korlátozásokat adnak a változók értékeire. Egy gyakori korlátozás egy lehetséges megfogalmazása a 2.1, mely azt biztosítja hogy egy időpontban egy berendezés csak egy taszkot kezdhet el. I a taszkok, J a berendezések, T az időpontok halmaza és $x_{i,j,t}$ a taszk elkezdését jelölő bináris változó.

$$\sum_{i \in I} x_{i,j,t} \leq 1 \quad \forall j \in J, t \in T \quad (2.1)$$

Léteznek olyan időábrázolási módszerek, amelyek nem időpontokon, hanem a taszkok közötti precedencián alapulnak. Ilyenkor a taszkok kezdeti ideje egy folytonos változó, a sorrendet pedig a bináris precedencia-változók határozzák meg. A modellek között megkülönböztetünk általános[9], ill. közvetlen precedenciát[10] is a bináris változók jelentése szerint.

$$x_{i,i'} = \begin{cases} 1 & \text{ha } i \text{ megelőzi } i' \text{-t} \\ 0 & \text{különben} \end{cases} \quad \forall i, i' \in I \quad (2.2)$$

Közvetlen precedencia esetén minden taszknak legfeljebb egy rákövetkező taszkja van:

$$\sum_{i' \in I} x_{i,i'} \leq 1 \quad \forall i \quad (2.3)$$

Az általános precedencia leírásában minden később kezdődő taszkot rákövetkezőnek tekintenek, ezért $x_{i,i'} + x_{i',i} = 1$. Az utóbbi formuláció előnye, hogy kevesebb változót eredményez, és egyszerűbben megfogalmazhatóak az erőforrásokra vonatkozó korlátozások.

Az időpontos modellek képesek általános topológiájú rendszerek ütemezésének megoldására, valamint olyan kiegészítő korlátozások megfogalmazására, mint az emberi erőforrások, energiafogyasztás, stb. kezelése. A precedencia alapú módszerek csak szekvenciális folyamatoknál használhatóak és nehezen bővíthetők, de nagyobb időtartamra vagy legyártandó mennyiségre ütemezve is jól alkalmazhatók gyorsaságuk miatt.

Az STN és RTN reprezentációk közti különbség a MILP modellekben is megmutatkozik. RTN esetén a berendezések ugyanolyan erőforrásnak számítanak, mint a nyersanyagok vagy az emberi munkaerő. A szakirodalom megújuló erőforrásoknak nevezi az olyan erőforrások csoportját, amelyeket a taszkok felhasználnak kezdetükkor, majd újra legyártják őket a folyamat végén. A berendezések is megújuló erőforrások, így nem kell új változókat felvenni, ha valamelyikből több egyforma áll rendelkezésre. Az STN-modellek

általánosan használhatók, míg az RTN alapúakban az egyes berendezésekre vonatkozó korlátozások, pl. tisztítási idők nehezen kezelhetők.

A folyamatok leírásához használt változók és paraméterek segítségével a célfüggvény is felírható. Például a teljes gyártási idő minimalizálására használható a 2.4 korlátozás, ahol MK a minimalizálandó makespan változó, T_t a t folytonos időpont idejét jelölő változó és a $pt_{i,j}$ paraméter adja meg i taszk j berendezésben történő végrehajtási idejét.

$$MK \geq x_{i,j,t}(T_t + pt_{i,j}) \quad \forall i \in I, j \in J, t \in T \quad (2.4)$$

A MILP alapú módszerek széles körben való használhatóságuk miatt váltak elterjedtté a gyártórendszerek optimalizálásában. A modellek felírása legtöbbször intuitív módon történik, ezért előfordulhatnak modellezési hibák, melyekre az utóbbi időben néhány szakirodalmi cikk[11] is rámutatott.

2.2.2. Ciklikus ütemezési modellek

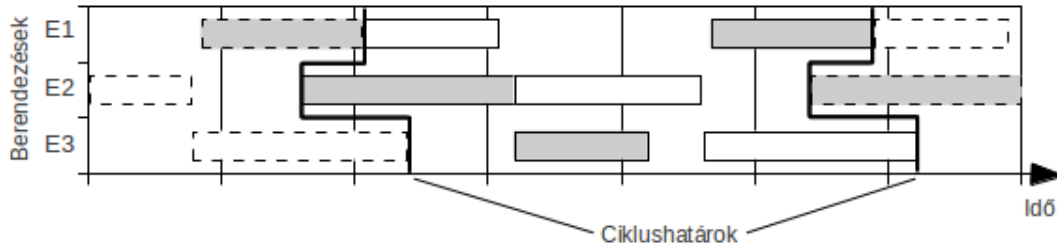
A ciklikus ütemezésre Shah és társai 1993-ban publikáltak egy módszert[1], amely több későbbi munkának is alapjául szolgált. Megfogalmazták a feladatot, miszerint az ismétlődő ciklusokat úgy kell ütemezni, hogy a szomszédosak illeszkedjenek egymáshoz. Ennek garantálására bevezették az $\Omega(t)$ wrap-around operátort, amely megfelelteti a ciklusidő utáni időpontokat a ciklus eleji időpontokkal. Így a taszkok annyival a ciklusidő után is véget érhetnek, amennyivel később kezd az első taszkját az adott berendezés.

$$\Omega(t) = \begin{cases} t & \text{ha } t = 1, \dots, |T| \\ \Omega(t - |T|) & \text{ha } t > |T| \\ \Omega(t + |T|) & \text{ha } t < |T| \end{cases} \quad (2.5)$$

A ciklusok illeszkedését jól szemlélteti a 2.3 ábrán látható Gantt-diagram. A szakirodalomban is szokásos ábrázolásmód szerint a téglalapok jelölik a taszkokat, eltérő színezéssel megkülönböztetve az egyes termékeket. Minden berendezés időbeosztása külön sorban helyezkedik el.

Megfigyelhető, hogy ez a formuláció megkülönböztet olyan ütemezéseket, amelyek csak a ciklushatárok helyében különböznek, de a taszkok ugyanúgy követik egymást. Ez a redundancia kiküszöbölhető, ha valamelyik taszk egy időponthoz van rögzítve, azaz a hozzárendelési változója előre 1-re van beállítva. Egy másik lehetőség egy olyan korlátozás hozzáadása, hogy az az időpont legyen az első, amelyikben az elkezdődő taszkok száma a legnagyobb. Az első módszer gyorsabb, és a manapság használatos megoldók előfeldolgozó technikáinak segítségével további redundanciát is megszüntethet.

2.3. ábra. Egy ciklikus ütemezés Gantt-diagramja



Schilling és Pantelides módszerükben[12, 13] továbbfejlesztették a modellt. RTN reprezentációt használtak folytonos, globális időábrázolással. Mivel az optimális ciklusidő előre nem ismert, ez is a modell egyik változója. Ez pedig nemlinearitásokhoz vezet a célfüggvényben és a korlátozásokban is. Az így kapott kevert egészértékű nemlineáris programozási feladatot, vagyis MINLP-t a korlátozás és szétválasztás módszerével oldották meg. A korlátozásokat különböző technikákkal linearizálták, a ciklusidőt pedig minden iterációban rögzítve, MILP-ek sorozatát oldották meg. A ciklusidő megválasztása része az általuk kifejlesztett háromlépcsős hibrid brach-and-bound algoritmusnak. Elsőként az időpontok közti távolságon, majd a ciklusidőn, végül a bináris változókon végezték a szétválasztást.

Castro és társai esettanulmányukban[14] egy diszkrét időintervallumos megközelítést is bemutattak a gyorsított folytonos modell mellett. Az időpontok diszkrétizálása megkönnyítette az optimális ciklusidő keresését. Az időpontok számának növelésekor az előző érték alapján gyorsabban ki lehet zárni a szuboptimális megoldásokat. A folytonos módszernél a legyártott termék mennyiségét rögzítették és a ciklusidőt minimalizálták, elkerülve a nemlinearitást.

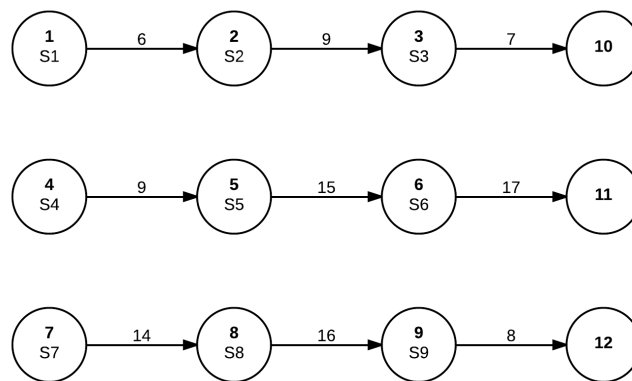
Wu és Ierapetritou[15] egy STN-alapú modellt fejlesztettek ki szakaszos üzemű gyárak ciklikus ütemezésére. A formuláció változó méretű munkák ütemezésére is lehetővé tette. Iteratív módszer helyett MINLP-megoldókat használtak. Pochet és Warichet[16] tovább erősítették a folytonos formulációt, az iteratív megoldást pedig többféle heurisztika alkalmazásával gyorsították.

2.3. S-gráf módszertan

Az S-gráf[17] egy kombinatorikus módszer szakaszos üzemű gyártórendszerek ütemezésére. A matematikai programozás technikáival ellentétben itt nem változókból és paraméterekből készül a modell, hanem csúcsokból és súlyo-

zott, irányított élekből. A gyártási receptből egy úgynevezett receptgráf készül, ahol a csúcsok a taszkokat, valamint a végtermékeket jelölik, az élek pedig a taszkok sorrendjét. A taszkokhoz hozzá van rendelve a végrehajtásukra alkalmas berendezések halmaza (A 2.4 ábrán S1-S9 halmazok). Az élek súlya a taszkok végrehajtási idejének felelnek meg. Mivel a különböző berendezések végrehajtási idejei nem feltétlenül egyeznek meg, a receptgráfban a súlyok a minimális időszükségletet jelölik.

2.4. ábra. Példa egy receptgráfra

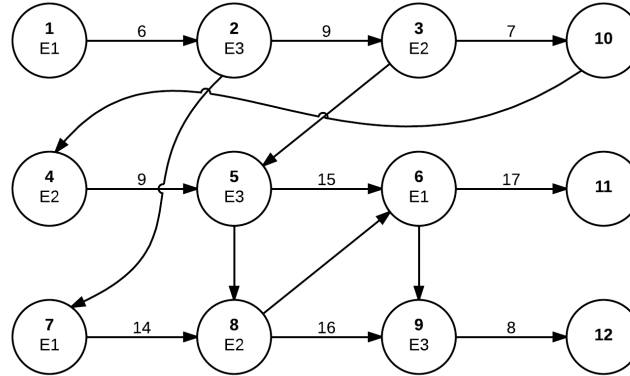


Az ütemezés során ütemezési S-gráfok generálódnak, melyek a taszkok különböző ütemezéseinek felelnek meg. Minden taszkhoz pontosan egy berendezés kerül hozzárendelésre a megfelelő halmazból. Ezután az azonos berendezéshez tartozó taszkok sorrendjét kell meghatározni a megfelelő ütemezési élek behúzásával. Az S-gráfban az élek súlyozása a taszkok kezdete közt eltelt idő alsó korlátjának felel meg. Egy taszk akkor kezdődhet el, ha a csúcsba mutató élek elején lévő taszk kezdete után legalább annyi idő telt el, amennyi az él súlya.

Az ütemezési élek behúzásának módja a tárolási stratégiától függ. Egy UIS kimenetű taszk véget érése után azonnal elérhetővé válik az adott berendezés. Tehát a berendezés következő taszkjához egy olyan élt kell húzni, aminek a súlya a végrehajtási idő lesz. NIS taszkoknál viszont csak akkor tud a berendezés egy másik taszkot elkezdni, ha az elkészült anyag már egy másik berendezésbe került. Az S-gráfban ez úgy valósítható meg, hogy egy 0 súlyú ütemezési él vezet a recept szerinti következő taszkból a berendezés következő taszkjára. Egy ily módon ütemezett S-gráf látható a 2.5 ábrán.

Az ütemezési döntések részhalmazokra bontják az ütemezéseket. Egyével haladva, az eddig meghozott döntések alapján alsó korlátot lehet számolni egy-egy részhalmazra. A makespan alsó korlátja az S-gráf leghosszabb

2.5. ábra. A példa egy lehetséges S-gráfja



útjának hossza. Ez lehetővé teszi egy korlátozás és szétválasztás-algoritmus (Branch-and-Bound) bevezetését. Az algoritmus egy keresési fát készít, melyben minden csúcs leszármazottjában eggyel több ütemezési döntés van rögzítve. A kört tartalmazó S-gráfok megvalósíthatatlan ütemezést eredményeznek, ezért a leszármazottjait sem vizsgálja meg az algoritmus, akárcsak a már megtalált megoldásnál rosszabb alsó korláttal rendelkezőket. Így a keresési tér kis részének bejárásával is garantáltan megtalálja az optimális ütemezést. Az algoritmus egy előzetes körkeresést használva gyorsan felismeri a kör kialakulásához vezető részproblémákat[18].

Ha egy termékből több adagot (batch-et) is szeretnénk gyártani, akkor a receptgráfban minden taszkját többször kell felvenni. Hogy ne generáljon olyan redundáns ütemezéseket, amelyek csak a batch-ek sorrendjében különböznek, az algoritmus felgyorsítható a sorrend rögzítésével. Ehhez a batch-ek taszkjai közé segédéleket kell behúzni. A módszer nem veszélyezteti az optimalitást, csak az ekvivalens ütemezésekből szűri ki a redundanciát.[18]

3. fejezet

Ciklikus ütemezés S-gráffal

Dolgozatunk megoldandó feladata egy optimális ciklikus ütemezés meghatározása ismeretlen ciklusidő mellett. A megoldó algoritmus 3.3 fejezetben kerül bemutatásra, miután ismertettük azokat az általunk fejlesztett szubrutinokat, melyeket működése során használ. Az megoldás felgyorsítására két módszer is implementálva lett, melyeket a 3.3.1 és 3.3.2 alfejezetekben mutatunk be.

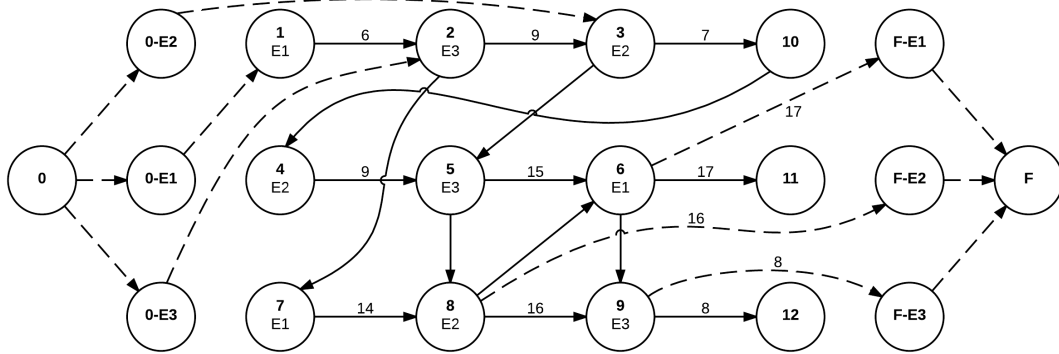
3.1. Ciklusidő minimalizálása

Míg egy ütemezett S-gráf esetén a leghosszabb útjának megkeresésével lehet megkapni a makespan-t, a ciklusidőt más módon kell kiszámolni. A ciklus hosszát pontosan meghatározza, hogy a leghosszabb ideig dolgozó berendezés mikor végez. Ez az intervallum lesz a ciklus, a többi berendezést ennek az ütemezéséhez kell igazítani.

Az S-gráfban a ciklusidő a berendezések első taszkjától az utolsó taszkját követőig vezető leghosszabb utak hosszai közül a legnagyobb. A receptgráfot ki lehet egészíteni olyan segédcsúcsokkal, melyekből minden berendezéshez tartozik egy kezdő és egy utolsó. A kezdő a berendezés minden taszkja elé van ütemezve, az utolsó pedig minden taszkja után. Így a leghosszabb utakat mindig ezen csúcspárok között kell kiszámolni. Az ezek közötti leghosszabb megkeresése tovább egyszerűsíthető két további segédcsúccsal, az egyikből az első csúcsokhoz húzva éleket, a másikba az utolsókból húzva. A segédcsúcsok alkalmazásával az eredeti makespan minimalizáló algoritmus annyiban módosul minimális ciklusidő kiszámítása esetén, hogy csak a két szélső csúcs közötti leghosszabb utat kell megkeresni, nem a gráf bármely két pontja közöttit. A segédcsúcsok látszólag hosszabb utakat hoznak létre az S-gráfban, miközben az optimális ciklusidő nem lehet nagyobb a makespan-nél. Valójá-

ban az eredeti csúcsok között nem jön létre hosszabb út, a keresés pedig nem terjed ki a teljes gráfra, ezért az így kapott ciklusidő valóban nem haladja meg a makespan-t.

3.1. ábra. S-gráf segédcsúcsokkal kiegészítve



A 3.1 ábrán az előző fejezetben szereplő S-gráf látható segédcsúcsokkal kiegészítve. A berendezésekhez tartozó csúcsokra elnevezésük is utal. A leghosszabb út 0-tól F-ig 63 súlyú, és a 0-E1 és F-E1 csúcsokon vezet. Ez lesz tehát a ciklusidő értéke, a másik két berendezés útja 32 és 49 súlyú, ezért beleférnek az intervallumba. Megfigyelhető, hogy az útnak nem kell a berendezés minden taszkját tartalmaznia. A példában a 7-es taszk nem része az útnak, mert az 5-ös taszk befejezésére többet kell várni.

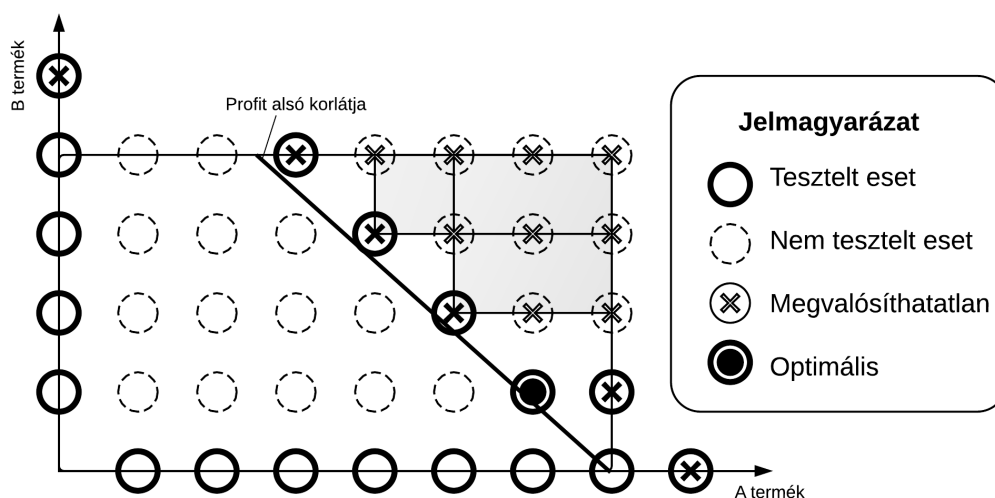
Az implementált megoldó segédcsúcsok használata helyett egy vektorban tárolja a berendezések vonatkozó leghosszabb utakat. A megoldás során gyakran van szükség ezekre az értékekre, viszont az algoritmus során lépésenként csak néhányuk értéke változik meg. Az alsó korlát számolásakor csak azok az értékek kerülnek újbóli kiszámításra, melyek megváltozhattak az aktuális ütemezési döntés miatt.

3.2. Profit maximalizálása

Az egy ciklus alatt legyártandó batch-ek számát adott ciklusidőre hasonlóképp lehet meghatározni, mint a nem periodikus throughput maximalizálás[19] esetén. Az algoritmus a lehetséges batch-kombinációkat bejárva, minden esetre meghív egy szubrutint, ami egy logikai értékkel tér vissza. Az előző alfejezetben bemutatott ciklusidő minimalizáló módosított változata vizsgálja meg, hogy az adott termékmennyiség legyártható-e a rögzített ciklusidőn be-

lül. A kezdeti felső korlát a maximális megengedett ciklusidő lesz, és nem kell megkeresni az optimális ütemezést, elég egy lehetséges megoldást találni. A tapasztalatok azt mutatják, hogy ha létezik ilyen megoldás, azt gyorsabban megtalálja az algoritmus, mint ahogy a megadott idő alatt nem legyártható mennyiségekre belátja, hogy teljesíthetetlenek.

3.2. ábra. Profit maximalizálása korlátolt ciklusidőre



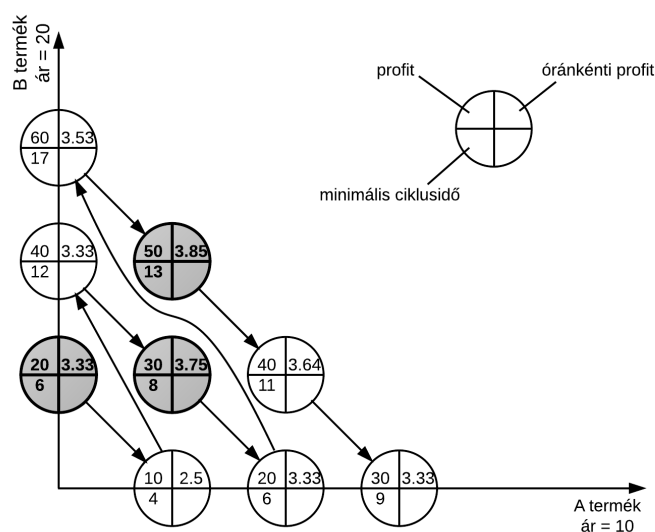
Az algoritmus először a véges keresési teret határozza meg. Minden termékre egyesével növelve a batch-ek számát, meg kell hívni a szubrutint, amíg az talál megoldást. A termékek maximális mennyisége határozza meg a keresési tér határait. Az esetek vizsgálata során tovább csökken a keresési tér. Ha új megvalósítható ütemezést talált az algoritmus, az annál kisebb profittal rendelkező batch-számok szuboptimális tulajdonságuk miatt kizárásra kerülnek. Ha a vizsgált ütemezés nem megvalósítható a ciklusidőn belül, akkor a nála több batch-et tartalmazók sem lesznek, így azok is kizárhatók a keresési térből.

3.3. Időegységenkénti profit maximalizálása

Jelen feladatban a célfüggvény a profit és a ciklusidő hányadosa. Ahhoz, hogy a nemlineáris feladatot lineáris módszerekkel meg lehessen oldani, a számlálót vagy a nevezőt különböző értékeken kell rögzíteni, és így kiszámolni a hányadost. A matematikai programozási módszerek általában a ciklusidőt rögzítik

iterációnként. S-gráf esetében ez nem lenne hatékony, mert így szinte ugyanazt a keresési teret járná be újra és újra. A batch-ek számának, és ezáltal a profitnak az előzetes meghatározásával viszont más recept-gráfok és keresési fák generálódnak. Az implementált algoritmus több szálon, párhuzamosan képes vizsgálni a részproblémákat.

3.3. ábra. A keresési tér bejárása



A bejárás a throughput maximalizáláshoz hasonlóan működik, de a ciklusidőre nincs felső korlát, ezért az egy ciklus alatt legyártható mennyiségre sem. A részproblémáknál minimális ciklusidőt kell keresni a célfüggvény értékének meghatározásához. Ahogy a 3.3 ábrán is látszik, a keresés a legkisebb feladatokkal kezdődik, mikor csak egy termékből kell egy batch-et ütemezni. A mennyiség növelésével lehet újabb részproblémákat generálni, és így egy szélességi kereséssel alulról bejárni a keresési teret. Az ábrán szürkével vannak kiemelve az aktuális célfüggvény-értéket javító ütemezések.

Az optimális megoldás gyakran nagyon távol van a keresés kezdőpontjától. Könnyen belátható, hogy a batch-számok egész számú többszörösére növelésével az időegységenkénti profit nem csökkenhet, hiszen a kisebb részprobléma ciklusának ismétlésével egy ugyanolyan célfüggvény-értékű ciklust lehet létrehozni. Az algoritmus nem a garantáltan optimális ütemezést keresi, hanem a számítási időkorláton belül megtalált legjobb megoldásokat adja vissza.

Több gyakorlati előnye is van ennek a keresésnek, ahol a kisebb termék-mennyiségű feladatok kerülnek először ütemezésre. Az először vizsgált ese-

tekben várhatóan a ciklusidő is alacsonyabb lesz, amit ipari szempontból könnyebb kivitelezni, jobban átlátható és kevesebb karbantartást igényel. A megoldó szoftver képes a legjobb ütemezésekről listát készíteni, így a gyártásért felelős személyek több lehetőség közül is választhatnak. Egy kis mértékben kevésbé jövedelmező ütemezés más szempontokból (pl. környezetvédelem, biztonság) előnyösebb lehet.

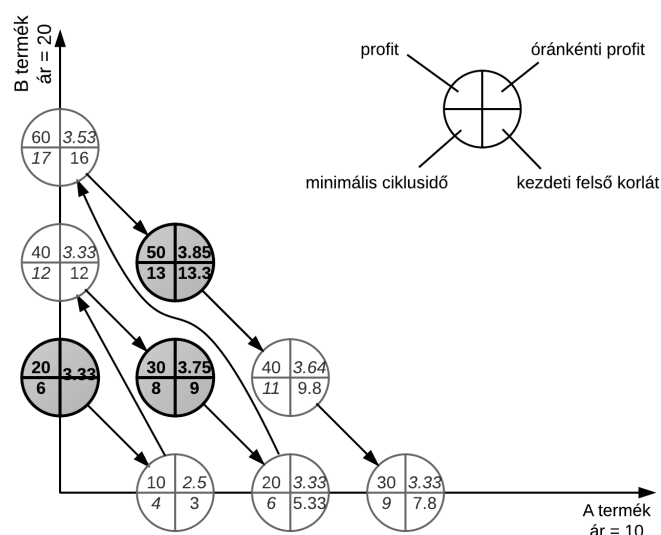
További előnye a keresési módszernek, hogy tovább gyorsítható különböző technikákkal, melyek a keresési teret szuboptimális esetek kiszűrésével csökkentik, segítik a ciklusidő kiszámítását vagy az optimális irányba terelik a keresést.

3.3.1. Első gyorsító módszer

Az első technika a részproblémák megoldását gyorsítja egy kezdeti felső korlát megadásával. Az eddig megtalált legjobb megoldásból minden batch-számra ki lehet számolni, hogy legfeljebb mekkora ciklusidő tarthat hozzá, hogy jobb legyen az időegységenkénti profit értéke. A kiszámolt maximális ciklusidőt az aktuálisan vizsgált részproblémánál kezdeti felső korlátként alkalmazva, a szuboptimális S-gráfokat kiszűri az algoritmus.

$$\text{felső korlát} = \frac{\text{batch-ek profit értéke}}{\text{aktuális legjobb megoldás}}$$

3.4. ábra. Az első gyorsítás működése



A 3.4 ábra egy példán szemlélteti a módszer működését. A nyilak jelölik a bejárési sorrendet. Azoknál az eseteknél, ahol a minimális ciklusidő a felső korlátnál nagyobb, az algoritmus nem számolja ki a tényleges értéket, az ábrán csak szemléltetés céljából vannak feltüntetve.

Ez a gyorsítási technika nem a külső algoritmus keresési terét csökkenti, hanem a generált S-gráfok számát. A ciklusidő-minimalizáló ugyanúgy minden rácspontban meghívódik, de a kezdeti felső korlát által kizárt szub-optimális ütemezések miatt gyorsabban lefut. Azoknál az eseteknél is gyorsít a módszer, ha az optimális ütemezés javít a célfüggvény értékén.

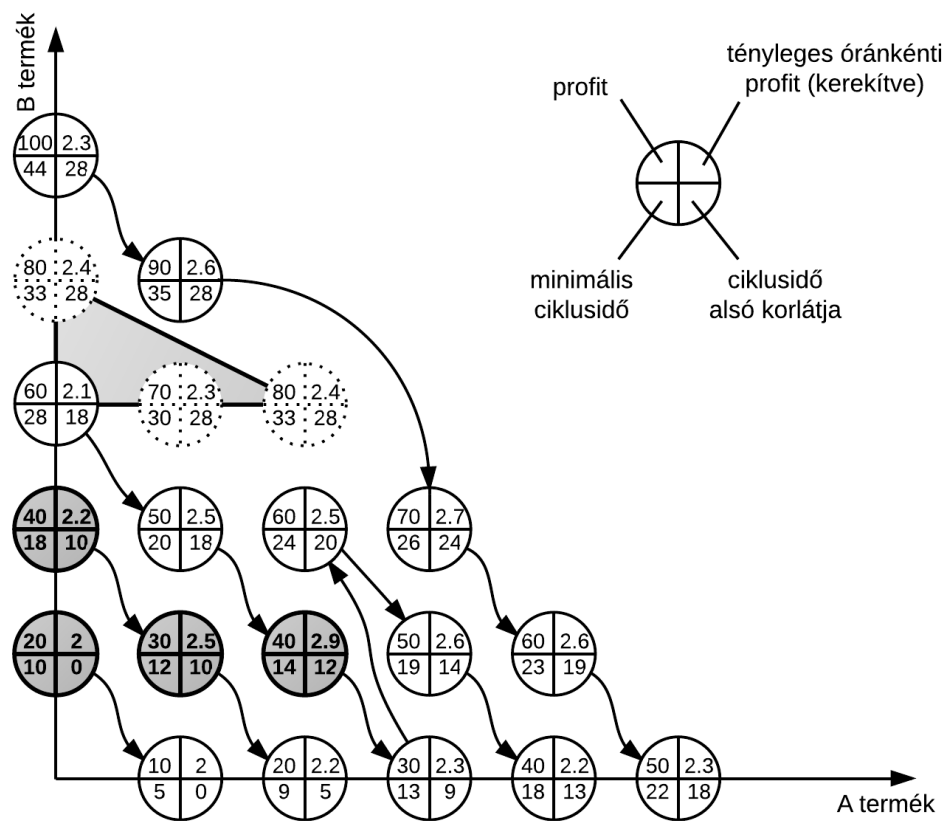
3.3.2. Második gyorsító módszer

A következő gyorsítási módszer a külső keresési teret csökkenti. Magától értődően minden részproblémára teljesül az állítás, hogy a nála nagyobb batch-számokkal rendelkező esetek minimális ciklusideje legalább ugyanakkora. A kapott alsó becslést felhasználva az algoritmus kizárja a keresésből azokat az eseteket, melyek profitjának és a kiszámolt ciklusidőnek a hányadosa kisebb az aktuális legjobb megoldásnál.

A 3.5 szemléltető ábrán pontozott körvonallal vannak jelölve a kizárt esetek. A ciklusidőre vonatkozó alsó korlátok felső korlátot adnak az óránkénti profitra. Megfigyelhető, hogy amikor a 3 batch B termék ütemezése megtörténik, akkor a leszármazott részproblémákra még nagyobb a felső korlát ($80/28 \approx 2.86$), mint az eddigi legjobb megoldás értéke. Mire az algoritmus eléri őket, a korlát fölé javul a célfüggvény értéke, ezért ezeket az eseteket kizárja a vizsgálatból. A részproblémák kizárása akár több mélységig is terjedhet, amit a háromszög kiterjedése is jelöl.

A második gyorsítási módszerben ki kell számolni a minimális ciklusidőt azokban az esetekben is, amikor ez nem javítja a célfüggvény értékét. Tehát a két technika csak úgy használható párhuzamosan, ha minden eset megvizsgálása előtt az algoritmus eldönti, hogy melyiket alkalmazza.

3.5. ábra. A második gyorsítás működése



4. fejezet

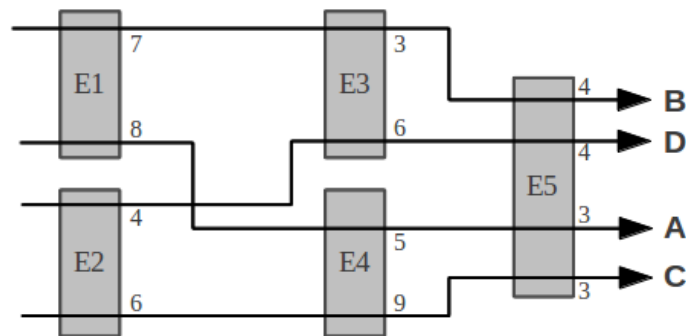
Teszteredmények

Több tesztet végeztünk a megoldó és a gyorsítási módszerek hatékonyságának vizsgálatára. A megoldó a tanszéki S-gráf megoldó[20] javított, kibővített változatára épül.

4.1. Ciklusidő minimalizálás

Az első tesztpéldában használt szekvenciális multiproduct feladatot Vouduouris és Grossmann[21] mutatta be. A rendszer felépítése a 4.1 ábrán látható a termékek gyártási útvonalával és a lépések végrehajtási idejével.

4.1. ábra. Az első példa blokkdiagramja



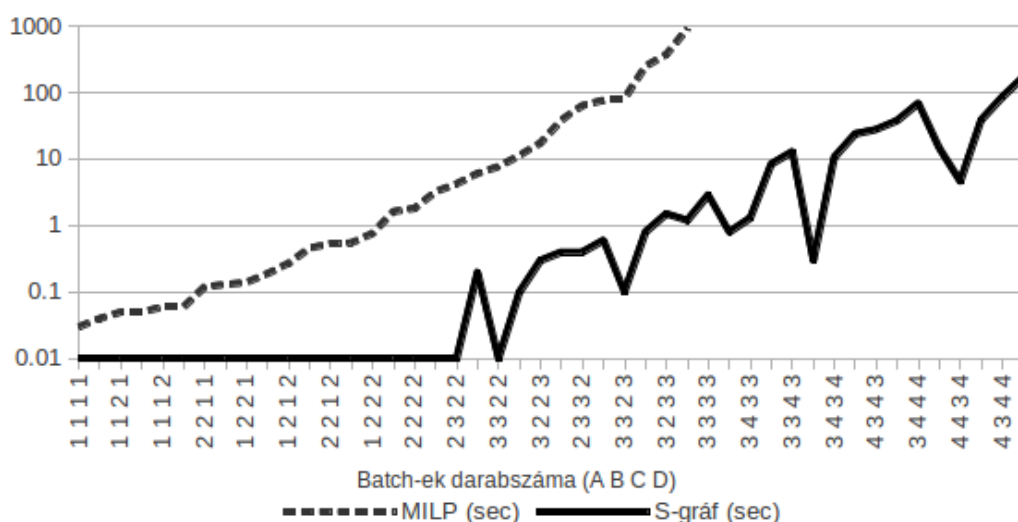
Az S-gráffal történő ciklusidő minimalizálás egy precedencia alapú módszerrel lett összehasonlítva. A MILP modell Méndez és Cerdá[22] makespan minimalizáló formulációjára épül. A célfüggvény a ciklikus ütemezésre lett

módosítva, az S-gráfban is használt módon: a berendezések első taszkjának elejétől az utolsó taszkjának végéig tartó időt minimalizálja.

A ciklusidő minimalizálást egy Intel Core i5-520M (2,4 GHz, 2 fizikai, 4 logikai mag) CPU-val és 4 GB DDR3 (1066 MHz) memóriával rendelkező laptopon teszteltük 1000 ms időlimittel. A MILP modellt az IBM ILOG CPLEX Optimizer 12.3.0.0 programmal oldottuk meg.

A tesztek összehasonlítását a 4.2 ábrán látható diagram tartalmazza. A részletes eredmények a Függelékben találhatóak. A számítási idők közti nagy eltérések miatt logaritmikus skálán vannak ábrázolva az értékek. Az S-gráf módszer nagyságrendekkel gyorsabban találta meg az optimális ciklusidőt. A MILP megoldó a nagyobb esetekben talált a minimális ciklusidőhöz tartozó ütemezést, de az időkorlát alatt nem tudta bizonyítani az optimalitást.

4.2. ábra. Ciklusidő minimalizálása



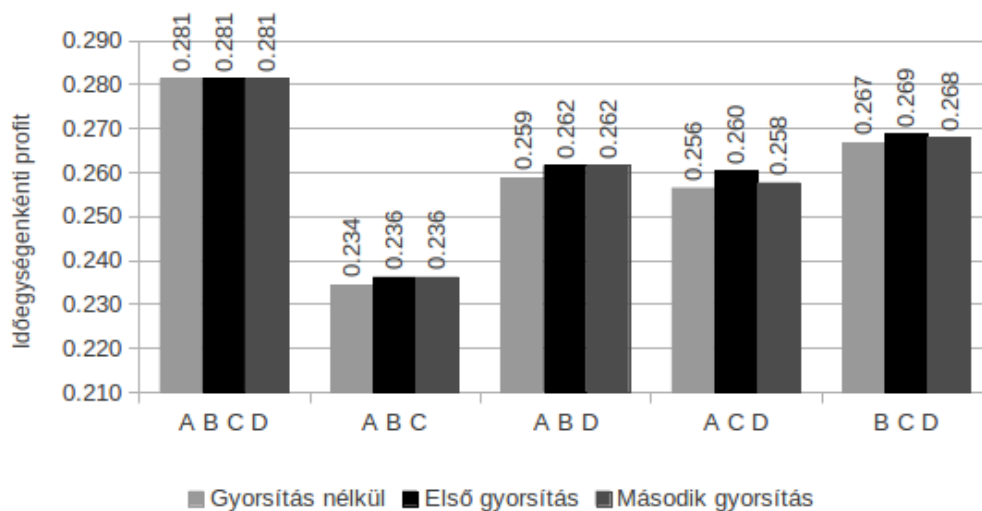
4.2. Optimális profit/ciklusidő meghatározása

Ebben a fejezetben a megoldó és a gyorsító módszerek hatékonysága kerül összehasonlításra.

A megoldó és a gyorsítások teszteléséhez egy Intel Core 2 Duo P8400M (2,26 GHz) kétmagos CPU-val és 4 GB DDR2 (800 MHz) SDRAM-mal felszerelt laptopot használtunk. Az időkorlát 1000 ms volt. Párhuzamosításának köszönhetően, az algoritmus ki tudja használni a többmagos processzor jelentette nagyobb számítási kapacitást.

A tesztek első példájaként a ciklusidő minimalizálásnál is használt receptet alkalmaztuk. A ciklikus megoldó tesztelésénél nincs meg az a lehetőségünk, mint az előző fejezetben, hogy a batch-számok növelésével generáljunk újabb teszteseteket. Ezt úgy oldottuk meg, hogy a feladatokat a termékek különböző részhalmazaira írtuk fel. A részhalmazokba beválasztott termékek kötelezően gyártandóak voltak (legalább egy batch), míg a többi termék gyártása nem volt megengedett. Így négy aktív termék esetén a megoldás biztosan eltér a részhalmazokétól.

4.3. ábra. Az első példa megoldásai



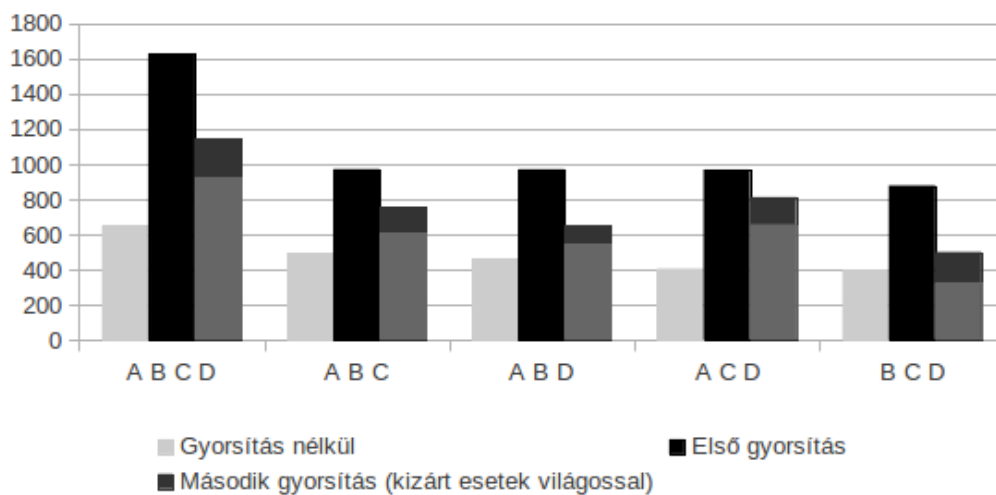
A legjobb megoldást minden esetre az első gyorsítás adta. A 4.4 ábrán látható, hogy a megvizsgált esetek számában is egyértelműen felülmúlta a második gyorsítást. A részproblémák kizárása is hasznosnak bizonyult, nagy mértékben csökkentette a keresési teret, de a módszer még további fejlesztésre szorul.

A megoldót egy másik példán is teszteltük, melyet Ferrer-Nadal és társai[23] mutattak be. Ez a multipurpose feladat kiválóan alkalmas arra, hogy a taszok bonyolultabb hálózatára is megvizsgáljuk az algoritmust.

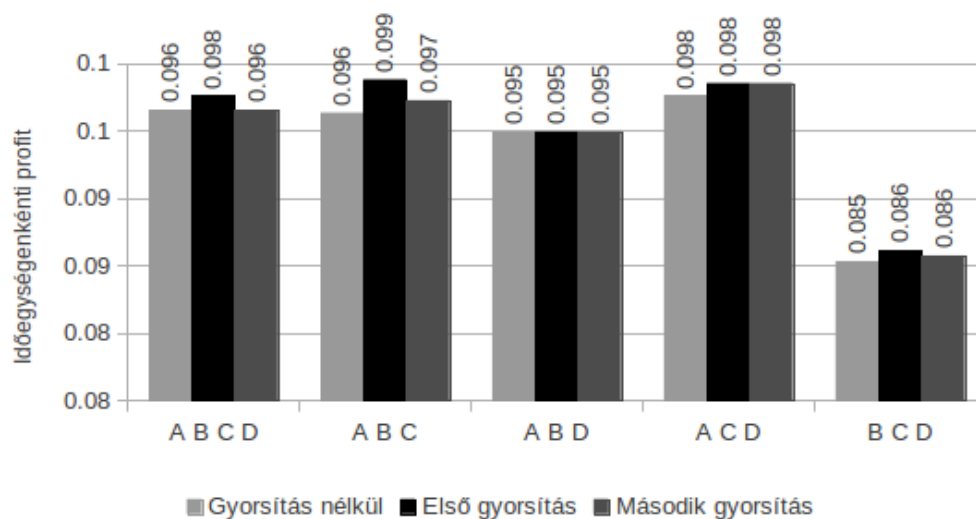
A második példa tesztjei hasonló eredményeket mutatnak, mint az első példában látottak. A különböző termékhalmazokra eltérő mértékű különbség mutatkozik a két gyorsítás között, ami a multipurpose jellegből ered. Ebből is látszik, hogy a multiproduct feladatokat könnyebben oldja meg a módszer.

A teszteredményekből kiderült, hogy az első módszer hatékonyabban gyorsítja fel az algoritmust, több részproblémát vizsgált meg és jobb megoldást

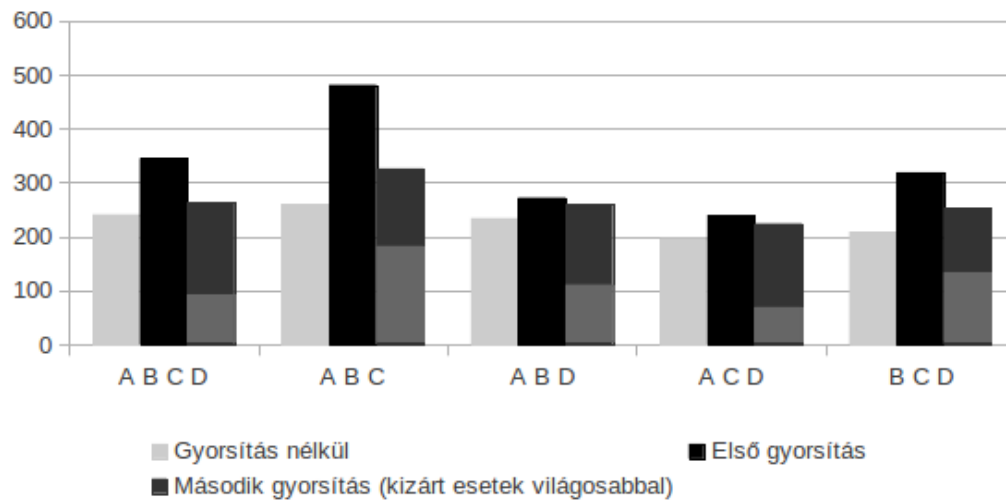
4.4. ábra. Az első példában megvizsgált részproblémák száma



4.5. ábra. A második példa megoldásai



4.6. ábra. A második példában megvizsgált részproblémák száma



adott ugyanakkora időkorláton belül. A második technika a nagyobb részproblémáknál nyújthat segítséget vagy megfelelő stratégiával az első módszerrel együttműködve. További kutatási célunk egy olyan gyorsítás kifejlesztése, ami segít a potenciálisan jobb megoldások megtalálására.

5. fejezet

Összegzés

Dolgozatunkban bemutatunk egy új, kombinatorikus megközelítésen alapuló módszert szakaszos üzemű gyártórendszerek ciklikus ütemezésére. Az implementált megoldó az irodalomban publikált MILP alapú módszereknél nagyságrendekkel gyorsabban képes megadni az optimális ciklusidőt adott mennyiségű termék legyártásához. Rögzített vagy korlátlan ciklusidőre való profit maximalizálás során a részfeladatokat több szálon párhuzamosan képes megoldani. A keresés gyorsítására gyorsítási technikák lettek kifejlesztve, implementálva, valamint hatékonyságuk szakirodalmi példákon futtatott tesztekkel lett alátámasztva.

A publikált módszer több kérdést és lehetőséget is felvet, melyekkel jövőbeni munkánk során kívánunk részletesebben foglalkozni. A meglévő, keresési teret redukáló gyorsítások további fejlesztése mellett tervezzük olyan bejárési stratégiák kidolgozását, melyek az algoritmusokat a célfüggvényt potenciálisan javító részfeladatok irányába terelik. Fontos további irány az alkalmazott módszerek kiterjesztése olyan UIS tárolási stratégiával rendelkező ütemezések esetére, ahol az egyes ciklusok közt megengedettek anyagátmenetek.

Irodalomjegyzék

- [1] N. Shah, C. C. Pantelides, and R. W. H. Sargent. Optimal periodic scheduling of multipurpose batch plants. *Annals of Operations Research*, 42:193–228, 1993.
- [2] C.C. Pantelides. Unified framework for optimal process scheduling. *Computers & Chemical Engineering*, pages 253–274, 1994.
- [3] E. Kondili, C.C. Pantelides, and R.W.H. Sargent. A general algorithm for short-term scheduling of batch operations—i. milp formulation. *Computers & Chemical Engineering*, 17(2):211 – 227, 1993.
- [4] M. Hegyhati, T. Holczinger, A. Szoldatics, and F. Friedler. Combinatorial approach to address batch scheduling problems with limited storage time. *Chemical Engineering Transactions*, 25:495–500, 2011.
- [5] G. Schilling and C.C. Pantelides. A simple continuous-time process scheduling formulation and a novel solution algorithm. *Computers & Chemical Engineering*, 20, Supplement 2(0):S1221 – S1226, 1996. European Symposium on Computer Aided Process Engineering-6.
- [6] X. Zhang and R.W.H. Sargent. The optimal operation of mixed production facilities—a general formulation and some approaches for the solution. *Computers & Chemical Engineering*, 20(6–7):897 – 904, 1996. Fifth International Symposium on Process Systems Engineering.
- [7] Christodoulos A. Floudas and Xiaoxia Lin. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Computers & Chemical Engineering*, 28(11):2109 – 2129, 2004.
- [8] M.G. Ierapetritou and C.A. Floudas. Short-term scheduling: New mathematical models vs algorithmic improvements. *Computers & Chemical Engineering*, 22, Supplement 1:S419 – S426, 1998. European Symposium on Computer Aided Process Engineering-8.

- [9] C.A. Méndez, G.P. Henning, and J. Cerdá. An milp continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilities. *Computers & Chemical Engineering*, 25(4–6):701 – 711, 2001.
- [10] C.A Méndez, G.P Henning, and J Cerdá. Optimal scheduling of batch plants satisfying multiple product orders with different due-dates. *Computers & Chemical Engineering*, 24(9–10):2223 – 2245, 2000.
- [11] M. A. Shaik and C. A. Floudas. Novel unified modeling approach for short-term scheduling. *Industrial & Engineering Chemistry Research*, 48:2947–2964, 2009.
- [12] G. Schilling and C.C. Pantelides. Optimal periodic scheduling of multipurpose plants in the continuous time domain. *Computers & Chemical Engineering*, 21, Supplement(0):S1191 – S1196, 1997.
- [13] G. Schilling and C.C. Pantelides. Optimal periodic scheduling of multipurpose plants. *Computers & Chemical Engineering*, 23(4–5):635 – 655, 1999.
- [14] P.M. Castro, A.P. Barbosa-Póvoa, and H.A. Matos. Optimal periodic scheduling of batch plants using rtn-based discrete and continuous-time formulations: a case study approach. *Industrial & engineering chemistry research*, 42(14):3346–3360, 2003.
- [15] D. Wu and M. Ierapetritou. Cyclic short-term scheduling of multiproduct batch plants using continuous-time representation. *Computers & Chemical Engineering*, 28(11):2271 – 2286, 2004.
- [16] Y. Pochet and F. Warichet. A tighter continuous time formulation for the cyclic scheduling of a mixed plant. *Computers & Chemical Engineering*, 32(11):2723 – 2744, 2008.
- [17] E. Sanmartí, L. Puigjaner, T. Holczinger, and F. Friedler. Combinatorial framework for effective scheduling of multipurpose batch plants. *AIChE J.*, 48(11):2557–2570, 2002.
- [18] T. Holczinger. *Módszer köztes tárolókat nem tartalmazó szakaszos működésű rendszerek ütemezésére*. PhD thesis, Veszprémi Egyetem Informatikai Tudományok Doktori Iskola, 2004.
- [19] T. Majozsi and F. Friedler. Maximization of throughput in a multipurpose batch plant under a fixed time horizon: S-graph approach. *Industrial & engineering chemistry research*, 45:6713–6720, 2006.

- [20] J. Smidla and I. Heckl. S-graph based scheduling method for parallel architecture. *Chemical Engineering Transactions*, 21:937–942, 2010.
- [21] V.T. Voudouris and I.E. Grossmann. Milp model for scheduling and design of a special class of multipurpose batch plants. *Computers & Chemical Engineering*, 20(11):1335–1360, 1996.
- [22] C.A. Mendez and J. Cerda. An milp continuous-time framework for short-term scheduling of multipurpose batch processes under different operation strategies. *Optimization and Engineering*, 4(1-2):7–22, 2003.
- [23] S. Ferrer-Nadal, E. Capón-García, C. A. Méndez, and L. Puigjaner. Material transfer operations in batch scheduling. a critical modeling issue. *Industrial & Engineering Chemistry Research*, 47(20):7721–7732, 2008.

A. függelék

Részletes teszteredmények

A.1. táblázat. Az első példa gyorsítás nélkül

Aktív termékek				Részproblémák		Legjobb megoldás						
A	B	C	D	Generált	Megoldott	A	B	C	D	C.idő	Profit	Pro./C.idő
1	1	1	1	924	650	3	1	1	4	9	32	3.5556
1	1	1	0	605	491	1	8	6	0	15	64	4.2667
1	1	0	1	572	462	6	1	0	8	15	58	3.8667
1	0	1	1	507	406	4	0	1	5	10	39	3.9000
0	1	1	1	499	399	0	4	4	4	12	45	3.7500
1	1	0	0	163	145	1	16	0	0	17	120	7.0588
1	0	1	0	179	160	1	0	1	0	2	14	7.0000
1	0	0	1	210	190	4	0	0	5	9	34	3.7778
0	1	1	0	210	190	0	11	8	0	19	78	4.1053
0	1	0	1	173	154	0	1	0	1	2	10	5.0000
0	0	1	1	187	168	0	0	1	1	2	10	5.0000

A.2. táblázat. Az első példa az első gyorsítással

Aktív termékek				Részproblémák			Legjobb megoldás						
A	B	C	D	Gen.	Mo.	Gyorsított	A	B	C	D	C.idő	Pro.	P/C
1	1	1	1	2157	1626	1622	3	1	1	4	9	32	3.5556
1	1	1	0	1145	971	964	1	9	7	0	17	72	4.2353
1	1	0	1	1143	970	964	7	1	0	9	17	65	3.8235
1	0	1	1	1140	969	961	8	0	1	10	19	73	3.8421
0	1	1	1	1043	878	870	0	6	6	6	18	67	3.7222
1	1	0	0	170	152	136	1	16	0	0	17	120	7.0588
1	0	1	0	183	164	155	1	0	1	0	2	14	7.0000
1	0	0	1	192	172	165	4	0	0	5	9	34	3.7778
0	1	1	0	192	172	165	0	11	8	0	19	78	4.1053
0	1	0	1	182	163	154	0	1	0	1	2	10	5.0000
0	0	1	1	192	172	163	0	0	1	1	2	10	5.0000

A.3. táblázat. Első példa a második gyorsítással

Aktív termékek				Részproblémák			Legjobb megoldás						
A	B	C	D	Gen.	Mo.	Kizárt	A	B	C	D	C.idő	Pro.	Pro./C.idő
1	1	1	1	1563	1146	907	3	1	1	4	9	32	3.5556
1	1	1	0	903	754	598	1	9	7	0	17	72	4.2353
1	1	0	1	783	649	532	7	1	0	9	17	65	3.8235
1	0	1	1	965	812	645	7	0	1	9	17	66	3.8824
0	1	1	1	615	500	312	0	5	5	5	15	56	3.7333
1	1	0	0	165	147	20	1	16	0	0	17	120	7.0588
1	0	1	0	196	176	55	1	0	1	0	2	14	7.0000
1	0	0	1	233	211	152	4	0	0	5	9	34	3.7778
0	1	1	0	323	298	221	0	11	8	0	19	78	4.1053
0	1	0	1	182	163	58	0	1	0	1	2	10	5.0000
0	0	1	1	207	187	95	0	0	1	1	2	10	5.0000

A.4. táblázat. A második példa gyorsítás nélkül

Aktív termékek				Részproblémák		Legjobb megoldás						
A	B	C	D	Gen.	Mo.	A	B	C	D	C.idő	Pro.	Pro./C.idő
1	1	1	1	382	241	7	1	1	2	11	114	10.3636
1	1	1	0	336	260	11	1	1	0	13	135	10.3846
1	1	0	1	306	234	11	1	0	1	13	137	10.5385
1	0	1	1	261	197	9	0	1	2	12	123	10.2500
0	1	1	1	274	209	0	1	1	9	11	129	11.7273

A.5. táblázat. A második példa az első gyorsítással

Aktív termékek				Részproblémák			Legjobb megoldás						
A	B	C	D	Gen.	Mo.	Gyors.	A	B	C	D	C.idő	Pro.	Pro./C.idő
1	1	1	1	524	345	335	8	1	1	2	12	123	10.2500
1	1	1	0	594	481	467	14	1	1	0	16	162	10.1250
1	1	0	1	349	272	251	11	1	0	1	13	137	10.5385
1	0	1	1	313	240	220	10	0	1	2	13	132	10.1538
0	1	1	1	406	319	308	0	1	1	11	13	151	11.6154

A.6. táblázat. Az második példa a második gyorsítással

Aktív termékek				Részproblémák			Legjobb megoldás						
A	B	C	D	Gen.	Mo.	Kiz.	A	B	C	D	C.idő	Pro.	Pro./C.idő
1	1	1	1	411	263	87	7	1	1	2	11	114	10.3636
1	1	1	0	414	326	178	12	1	1	0	14	144	10.2857
1	1	0	1	336	260	106	11	1		1	13	137	10.5385
1	0	1	1	294	224	64	10	0	1	2	13	132	10.1538
0	1	1	1	328	253	129	0	1	1	10	12	140	11.6667

A.7. táblázat. A ciklusidő-minimalizálás futási eredményei

A	B	C	D	Ciklusidő	MILP (sec)	S-gráf (sec)
1	1	1	1	16	0.0*	0.0*
1	1	1	2	18	0.1	0.0*
1	1	2	1	23	0.1	0.0*
1	1	2	2	25	0.2	0.0*
1	2	1	1	22	0.0	0.0*
1	2	1	2	24	0.3	0.0*
1	2	2	1	25	0.1	0.0*
1	2	2	2	27	0.8	0.0*
2	1	1	1	23	0.1	0.0*
2	1	1	2	23	0.1	0.0*
2	1	2	1	29	0.1	0.0*
2	1	2	2	30	0.5	0.0*
2	2	1	1	30	0.1	0.0*
2	2	1	2	30	0.5	0.0*
2	2	2	1	30	0.6	0.0*
2	2	2	2	32	1.8	0.0*
2	2	2	3	34	11.2	0.1
2	2	3	2	39	6.1	0.2
2	2	3	3	41	76.4	0.6
2	3	2	2	37	4.2	0.0*
2	3	2	3	38	38.2	0.4
2	3	3	2	41	63.5	0.4
2	3	3	3	43	907.0	1.2
3	2	2	2	38	1.6	0.0*
3	2	2	3	39	17.4	0.3
3	2	3	2	44	3.2	0.5
3	2	3	3	46	371.1	1.5
3	3	2	2	45	7.7	0.0*
3	3	2	3	45	80.2	0.1
3	3	3	2	46	246.3	0.8
3	3	3	3	48	N/A**	2.9
3	3	3	4	50	N/A**	8.4
3	3	4	3	55	N/A**	12.9
3	3	4	4	57	N/A**	37.8
3	4	3	3	52	N/A**	1.3
3	4	3	4	53	N/A**	10.6
3	4	4	3	57	N/A**	23.9
3	4	4	4	59	N/A**	68.6
4	3	3	3	53	N/A**	0.8
4	3	3	4	55	N/A**	14.6
4	3	4	3	60	N/A**	27.7
4	3	4	4	62	N/A**	84.9
4	4	3	3	60	N/A**	0.3
4	4	3	4	60	N/A**	4.6
4	4	4	3	62	N/A**	38.4
4	4	4	4	64	N/A**	168.9

*: a 0-hoz túl közeli valós érték a kerekítés során elveszett

** : a ciklus optimális ütemezését megtalálta, de az időlimit elérésekor még alacsonyabb volt az alsó korlát, ezért tovább vizsgált volna